Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Theoretical Computer Science 340 (2005) 577–593

Theoretical Computer Science

www.elsevier.com/locate/tcs

Informative labeling schemes for graphs

David Peleg[*,1]

Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot, 76100 Israel

Abstract

This paper introduces the notion of *informative labeling schemes* for arbitrary graphs. Let $f(W)$ be a function on subsets of vertices W. An *f labeling scheme* labels the vertices of a weighted graph G in such a way that $f(W)$ can be inferred (or at least approximated) efficiently for any vertex subset W of G by merely inspecting the labels of the vertices of W, without having to use any additional information sources.

A number of results illustrating this notion are presented in the paper. We begin by developing f labeling schemes for three functions f over the class of n-vertex trees. The first function, `SepLevel`, gives the *separation level* of any two vertices in the tree, namely, the depth of their least common ancestor. The second, `LCA`, provides the *least common ancestor* of any two vertices. The third, `Center`, yields the center of any three given vertices v_1, v_2, v_3 in the tree, namely, the unique vertex z connected to them by three edge-disjoint paths. All of these three labeling schemes use $O(\log^2 n)$-bit labels, which is shown to be asymptotically optimal.

Our main results concern the function $\mathtt{Steiner}(W)$, defined for weighted graphs. For any vertex subset W in the weighted graph G, $\mathtt{Steiner}(W)$ represents the weight of the Steiner tree spanning the vertices of W in G. Considering the class of n-vertex trees with M-bit edge weights, it is shown that for this class there exists a `Steiner` labeling scheme using $O((M + \log n)\log n)$ bit labels, which is asymptotically optimal. It is then shown that for the class of arbitrary n-vertex graphs with M-bit edge weights, there exists an *approximate*-`Steiner` labeling scheme, providing an estimate (up to a factor of $O(\log n)$) for the Steiner weight $\mathtt{Steiner}(W)$ of a given set of vertices W, using $O((M + \log n)\log^2 n)$ bit labels.

© 2005 Elsevier B.V. All rights reserved.

[*] Tel.: +972 8934 3478.
E-mail address: david.peleg@weizmann.ac.il.

[1] Supported in part by grants from the Israel Science Foundation and the Israel Ministry of Science and Art.

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.03.015

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 and combinatorial optimization to databases, distributed computing and communication networks. Research on network representations concerns the development of various methods and structures for cheaply storing useful information about the network and making 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 is dedicated to a somewhat neglected component of network representations, namely, the *labels* (or *names*, or *identifiers*) assigned to the vertices of the network. The issue of precisely how are vertex identifiers to be selected is often viewed as minor or inconsequential. For instance, most traditional *centralized* approaches to the problem of network representation are based on storing adjacency information using some kind of a data structure, e.g., an adjacency matrix. Such representation enables one to decide, given the indices of two vertices, whether or not they are adjacent in the network, simply by looking at the appropriate entry in the table. However, note that (a) this decision cannot be made in the absence of the table, and (b) the indices themselves contain no useful information, and they serve only as "place holders," or pointers to entries in the table, which forms a *global* representation of the network.

In contrast, the notion of *adjacency labeling schemes*, introduced by Breuer and Folkman in [2,1], involves using more *informative* and *localized* labeling schemes for networks. The idea is to associate with each vertex a label selected in a such way, that will allow us to infer the adjacency of two vertices *directly* from their labels, without using *any* additional information sources. Hence in essence, this rather extreme approach to the network representation problem *discards* all other components, and bases the entire representation on the set of labels *alone*.

Obviously, labels of unrestricted size can be used to encode any desired information. Specifically, it is possible to encode the entire row i in the adjacency matrix of the graph in the label chosen for vertex i. It is clear, however, that for such a labeling scheme to be useful, it should strive to use relatively *short* labels (say, of length polylogarithmic in n), and yet allow us to deduce adjacencies efficiently (say, within polylogarithmic time). The feasibility of such *efficient* adjacency labeling schemes was explored over a decade ago by Kannan et al. in [4].

Interest in this natural idea was recently revived by the observation that in addition to *adjacency* labeling schemes, it may be possible to devise similar schemes for capturing *distance* information. This has led to the notion of *distance labeling schemes*, which are schemes possessing the ability to determine the distance between two vertices efficiently (say, in polylogarithmic time again) given their labels. This notion was introduced in [9], and studied further in [3,6].

The current paper is motivated by the naturally ensuing observation that the ability to decide adjacency and distance are but *two of a number* of basic properties a representation may be required to possess, and that many other interesting properties may possibly be representable via an appropriate labeling scheme.

In its broadest sense, this observation leads to the general question of developing label-based network representations that will allow retrieving useful information about *arbitrary* functions or substructures in a graph in a *localized* manner, i.e., using only the local pieces of information available to, or associated with, the vertices under inspection, and not having to search for additional *global* information. We term such representations *informative labeling schemes*.

To illustrate this idea, let us consider the class of *rooted trees*. In addition to finding out whether two given vertices v and w are adjacent, or what is the distance between them, one may be interested in many other pieces of information concerning these vertices. For example, in some cases it may be useful to know if v is an *ancestor* (or a descendant) of w. It turns out that it is rather easy to encode the ancestry (or descendance) relation in a tree using interval-based schemes (cf. [10]). Another example for a useful piece of non-numeric information is the *least common ancestor* of v and w. Moreover, the types of localized information to be encoded by an informative labeling scheme are not limited to *binary* relations. An example for information involving *three* vertices v_1, v_2 and v_3 is finding their *center*, namely, the unique vertex z connected to them by edge-disjoint paths. More generally, for any subset of vertices W in the tree, one may be interested in inferring $S(W)$, the weight of their *Steiner tree* (namely, the lightest tree spanning them), based on their labels. The current paper demonstrates the feasibility of informative labeling schemes by providing such schemes for all of the above types of information over the class of rooted trees.

A natural question to ask at this point is whether efficient *exact* informative labeling schemes can be developed for *any* graph family (including, in particular, the family of *all* graphs). Unfortunately, the answer is negative. In [4] it is pointed out that for a family of $\Omega(\exp(n^{1+\varepsilon}))$ non-isomorphic n-vertex graphs, for $\varepsilon > 0$, any adjacency labeling scheme must use labels whose total combined length is $\Omega(n^{1+\varepsilon})$, hence at least one label must be of $\Omega(n^{\varepsilon})$ bits. In particular, any adjacency labeling scheme for the class of all n-vertex graphs requires labels of size $\Omega(n)$. The same observation carries over to other types of labeling schemes.

This raises the next natural question, namely, could more efficient labeling schemes be constructed if we abandon the requirement of capturing *exact* information, and settle for the less ambitious goal of obtaining *approximate* estimates. The last result presented in this paper is an approximate scheme for the Steiner weight function $S(W)$ over general weighted graphs.

The relevance of distance labeling schemes in the context of communication networks has been pointed out in [9], and illustrated by presenting an application of such labeling schemes to distributed connection setup procedures in circuit-switched networks. Some other problems where distance labeling schemes may be useful include memory-free routing schemes, bounded ("time-to-live") broadcast protocols, topology update mechanisms, etc.

It is plausible that other types of informative labeling schemes may also prove useful for other applications. In particular, Steiner labeling schemes may be utilized as a basic tool for optimizing multicast schedules and within mechanisms for the selection of subtrees for group communication via communication subtrees, and potentially even for certain information representation problems on the web.

1.2. Related work

Adjacency labeling systems of general graphs based on Hamming distances were studied by Breuer and Folkman in [2,1]. Specifically, in [2] it is shown that it is possible to label the vertices of every n-vertex graph with $2n\Delta$-bit labels, such that two vertices are adjacent iff their labels are at Hamming distance $4\Delta - 4$ or less of each other, where Δ is the maximum vertex degree in the graph.

An elegant labeling scheme is proposed in [4] for the class of trees using $2\log n$-bit labels. It is also shown in [4] how to extend that scheme, and construct $O(\log n)$ adjacency labeling schemes for a number of other graph families, such as bounded arboricity graphs (including, in particular, graphs of bounded degree or bounded genus, e.g., planar graphs), various intersection-based graphs (including interval graphs), and c-decomposable graphs.

It is clear that distance labeling schemes with short labels are easily derivable for highly regular graph classes, such as rings, meshes, tori, hypercubes, and the like. Whether more general graph classes can be labeled in this fashion is not as clear. It is shown in [9] that the family of n-vertex weighted trees with M-bit edge weights enjoys an $O(M\log n + \log^2 n)$ distance labeling scheme. This scheme is complemented by a matching lower bound given in [3], showing that $\Omega(M\log n + \log^2 n)$ bit labels are necessary for this class. The approach of [9] extends to handle also the class of c-decomposable graphs for constant c, which includes the classes of series–parallel graphs and k-outerplanar graphs, with $c = 2k$. Also, an approximate distance labeling scheme is given in [9] for the class of general weighted graphs.

In [3] it is shown also that n-vertex graphs with a k-separator support a distance labeling with labels of size $O(k\log n + \log^2 n)$. This implies, in particular, that the family of n-vertex planar graphs enjoys such a labeling scheme with $O(\sqrt{n}\log n)$-bit labels, and the family of n-vertex graphs with bounded treewidth has a distance labeling scheme with labels of size $O(\log^2 n)$. For n-vertex planar graphs, there exists also a lower bound of $\Omega(n^{1/3})$ on the label size required for distance labeling, leaving an intriguing (polynomial) gap. More recently, $O(\log^2 n)$ distance labeling schemes for n-vertex interval and permutation graphs were presented in [6].

1.3. Framework

Let us now formalize the notion of informative labeling schemes.

Definition 1.1. 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.

Definition 1.2. 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 $W = \{v_1, \dots, v_k\}$ in *G*, the value returned by the decoder $\mathcal{D}_f$ on the set of labels $\hat{L}(W) = \{L(v) \mid v \in W\}$ satisfies $\mathcal{D}_f(\hat{L}(W)) = f(W)$.

It is important to note that the decoder $\mathcal{D}_f$, responsible for 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 pre-computed by considering the whole graph structure.

Clearly, an *f*-decoder always exists for any graph family if arbitrarily large labels are allowed. Our focus here is on the existence of f labeling schemes which assign labelings with short labels.

For a labeling L for the graph $G = (V, E)$, let $|L(u)|$ denote the number of bits in the (binary) string $L(u)$.

Definition 1.3. 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 finite graph family $\mathcal{G}$, let

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

Labeling schemes providing *approximate* information are defined in an analogous way to Definition 1.2.

Definition 1.4. Let f be a function from sets of vertices in a graph to the integers. Given a family $\mathcal{G}$ of weighted graphs, *R-approximate 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 $W = \{v_1, \dots, v_k\}$ in *G*, the value returned by the decoder $\mathcal{D}_f$ on the set of labels $\hat{L}(W) = \{L(v) \mid v \in W\}$ satisfies

$$\mathcal{D}_f(\hat{L}(W)) \leqslant f(W) \leqslant R \cdot \mathcal{D}_f(\hat{L}(W)).$$

1.4. Our results

This paper starts by introducing and studying f-labeling schemes for three basic functions on the class $\mathcal{T}$ of unweighted trees. For a graph family $\mathcal{G}$, let $\mathcal{G}_n$ denote the subfamily containing the n-vertex graphs of $\mathcal{G}$.

First, we consider the *separation level* function $\texttt{SepLevel}$. The separation level of two vertices in a rooted tree is defined as the depth of their least common ancestor (i.e., its distance from the root of the tree). We show that this function is equivalent to the distance function on the class $\mathcal{T}$ of unweighted trees in terms of its labelability on trees, i.e., it requires labels of size $\Theta(\log^2 n)$, or formally, $\mathcal{L}(\texttt{SepLevel}, \mathcal{T}_n) = \Theta(\log^2 n)$.

Next, we consider an $\texttt{LCA}$ labeling scheme for trees, where $z = \texttt{LCA}(v, w)$ is the *least common ancestor* of any two vertices v, w. Formally, we assume that each vertex u has a unique *identifier*, denoted $I(u)$, typically of size $O(\log n)$, and the function $\texttt{LCA}$ maps the vertex pair (v, w) to the identifier $I(z)$. It is shown that for the class of n-vertex trees, there exists such a labeling scheme using $O(\log^2 n)$ bit labels, and this is asymptotically optimal, i.e., $\mathcal{L}(\texttt{LCA}, \mathcal{T}_n) = \Theta(\log^2 n)$.

Next, we turn to vertex triples, and consider the $\texttt{Center}$ function. The center of three vertices v_1, v_2, v_3 in a tree T is the unique vertex z such that the three paths connecting z to v_1, v_2 and v_3 are edge-disjoint. Here, too, we show the existence of an (asymptotically optimal) $\texttt{Center}$ labeling scheme using $O(\log^2 n)$ bit labels, i.e., $\mathcal{L}(\texttt{Center}, \mathcal{T}_n) = \Theta(\log^2 n)$ as well.

We then turn to *weighted* graphs. For a graph family $\mathcal{G}$, let $\mathcal{G}_{n,M}$ denote the subfamily containing the n-vertex graphs of $\mathcal{G}$ with M-bit edge weights. We consider $\texttt{Steiner}$ labeling schemes for graphs. Given a subset W of vertices in G, a *Steiner tree* $T_S(W)$ for W is a minimum weight tree spanning all the vertices of W (and perhaps some other vertices as well) in G. The *Steiner weight* of W, denoted $\texttt{Steiner}(W)$, is the weight of the Steiner tree $T_S(W)$. Using the $\texttt{LCA}$ labeling scheme, we show that the $\texttt{Steiner}$ weight function has an $O((M + \log n)\log n)$ size labeling scheme on the class $\mathcal{T}_{n,M}$ of weighted n-vertex trees with M-bit edge weights, and this is asymptotically optimal, i.e., $\mathcal{L}(\texttt{Steiner}, \mathcal{T}_{n,M}) = \Theta((M + \log n)\log n)$.

Finally, we consider the class of *arbitrary* weighted graphs $\mathcal{G}$. Note that $\text{dist}(v_1, v_2, G) = \texttt{Steiner}(W)$ for any pair of vertices $W = \{v_1, v_2\}$. Hence any $\texttt{Steiner}$ labeling scheme can be used also as a distance labeling scheme. Subsequently, given the lower bound of $\mathcal{L}(\text{dist}, \mathcal{G}_n) = \Theta(n)$ established in [3] for the class of *unweighted* n-vertex graphs $\mathcal{G}_n$, an exact $\texttt{Steiner}$ labeling scheme for the class of arbitrary weighted graphs $\mathcal{G}_{n,M}$ clearly requires at least $\Omega(M + n)$-bit labels.

We therefore turn to labeling schemes providing approximate information, and show that for the class of arbitrary n-vertex graphs with M-bit edge weights, there exists an $O(\log n)$-approximate $\texttt{Steiner}$ labeling scheme using $O((M + \log n)\log^2 n)$ bit labels.

This paper introduces the concept of informative labeling schemes, illustrates it through a number of simple examples and presents a rather preliminary study of the properties of such schemes. Many questions are left for further research. Informative labeling schemes for the functions of flow and connectivity in graphs were subsequently studied in [5]. A cardinal direction for future study is handling dynamically changing networks. This direction is pursued in [7], where some initial results are established for restricted dynamic network models.

2. `SepLevel` labeling schemes

We start with a `SepLevel` labeling scheme for trees. Consider a rooted tree T with root r_0. The depth of a vertex $v \in T$, denoted $\text{depth}(v)$, is its distance $\text{dist}(v, r_0)$ from the root r_0. Two vertices $v, w \in T$ are said to have *separation level* $\texttt{SepLevel}(v, w) = \ell$ if their least common ancestor z has depth $\text{depth}(z) = \ell$. We now claim that for the class $\mathcal{T}$ of unweighted trees, distance labeling and `SepLevel` labeling require the same label size up to an additive logarithmic[2] term.

Lemma 2.1. (1) $\mathcal{L}(\texttt{SepLevel}, \mathcal{T}_n) \leqslant \mathcal{L}(dist, \mathcal{T}_n) + \log n$.
(2) $\mathcal{L}(dist, \mathcal{T}_n) \leqslant (\texttt{Seplevel}, \mathcal{T}_n) + \log n$.

Proof. Suppose that we are given a distance labeling scheme $\langle \mathcal{M}_{\texttt{dist}}, \mathcal{D}_{\texttt{dist}} \rangle$ for $\mathcal{T}_n$. Define a `SepLevel` labeling scheme $\langle \mathcal{M}_{\texttt{SepLevel}}, \mathcal{D}_{\texttt{SepLevel}} \rangle$ for $\mathcal{T}_n$ as follows. Given a tree T, let L be the labeling assigned by $\mathcal{M}_{\texttt{dist}}$ for T. The `SepLevel`-marker $\mathcal{M}_{\texttt{SepLevel}}$ augments each label $L(v)$ into a label $L'(v)$ with an additional $\log n$ bit field containing v's depth, $\text{depth}(v)$.

Consider two vertices v, w with $z = \texttt{LCA}(v, w)$. Let $\ell_v = \text{dist}(z, v)$, $\ell_w = \text{dist}(z, w)$ and $\ell_{r_0} = \text{dist}(z, r_0) = \text{depth}(z)$. Given the labels $L'(v) = \langle L(v), \text{depth}(v) \rangle$ and $L'(w) = \langle L(w), \text{depth}(w) \rangle$, the fields $L(v)$ and $L(w)$ allow the `SepLevel`-decoder $\mathcal{D}_{\texttt{sepLevel}}$ to deduce the distance $\text{dist}(v, w) = \ell_v + \ell_w$, and the two additional fields provide it with $\text{depth}(v) = \text{dist}(v, r_0) = \ell_v + \ell_{r_0}$ and $\text{depth}(w) = \text{dist}(w, r_0) = \ell_w + \ell_{r_0}$. Combined, these three equations allow $\mathcal{D}_{\texttt{SepLevel}}$ to deduce $\text{depth}(z) = \ell_{r_0}$. Thus $\langle \mathcal{M}_{\texttt{SepLevel}}, \mathcal{D}_{\texttt{Seplevel}} \rangle$ is a `SepLevel` labeling scheme, and the labels it uses are larger than those used by $\langle \mathcal{M}_{\texttt{dist}}, \mathcal{D}_{\texttt{dist}} \rangle$ by $\log n$.

For the opposite direction, suppose that we are given a `SepLevel` labeling scheme $\langle \mathcal{M}_{\texttt{SepLevel}}, \mathcal{D}_{\texttt{SepLevel}} \rangle$ for $\mathcal{T}_n$. Define a distance labeling scheme $\langle \mathcal{M}_{\text{dist}}, \mathcal{D}_{\text{dist}} \rangle$ for $\mathcal{T}_n$ as follows. Given a tree T, let L be the labeling assigned by $\mathcal{M}_{\texttt{SepLevel}}$ for T. The dist-marker $\mathcal{M}_{\text{dist}}$ augments each label $L(v)$ into a label $L'(v)$ in the same way. The proof now follows along similar lines to the first part. $\square$

Based on the upper and lower bounds of [3,9] for distance labeling schemes for trees, we get

Corollary 2.2. *There exists a* `SepLevel` *labeling scheme for the class of n-vertex trees* $\mathcal{T}_n$ *using labels of* $O(\log^2 n)$ *bits, and any* `SepLevel` *labeling scheme for* $\mathcal{T}_n$ *requires some labels of* $\Omega(\log^2 n)$ *bits, i.e.,*

$$\mathcal{L}(\texttt{SepLevel}, \mathcal{T}_n) = \Theta(\log^2 n).$$

[2] For clarity of presentation we ignore rounding issues in stating our claims. For instance, here and in several other places, $\log n$ stands for $\lfloor \log n \rfloor$.

3. LCA labeling schemes

We now turn to developing an LCA labeling scheme for trees, where $z = \mathsf{LCA}(v, w)$ is the *least common ancestor* of any two vertices v, w. As mentioned earlier, this requires us to assume that each vertex u has a unique *identifier*, denoted $I(u)$, of size $O(\log n)$, and the function LCA maps the vertex pair (v, w) to the identifier $I(z)$.

3.1. Definitions

For every vertex v in the tree, let $T(v)$ denote the subtree of T rooted at v. For $0 \leqslant i \leqslant \mathrm{depth}(v)$, denote v's ancestor at level i of the tree by $\gamma_i(v)$. In particular, $\gamma_0(v) = r_0$ and $\gamma_{\mathrm{depth}(v)}(v) = v$.

Definition 3.1. A nonroot vertex v with parent w is called *small* if its subtree, $T(v)$, contains at most half the number of vertices contained in its parents' subtree, $T(w)$. Otherwise, v is *large*. (The root is defined to be small.)

For every vertex v, the "*small ancestor*" levels of v are the levels above it in which its ancestor is small,

$$SAL(v) = \{i \mid 1 \leqslant i \leqslant \mathrm{depth}(v), \quad \gamma_i(v) \text{ is small}\},$$

the *small ancestors* of v are

$$SA(v) = \{\gamma_i(v) \mid i \in SAL(v)\}$$

and their identifiers are

$$SAI(v) = \{I(\gamma_i(v)) \mid i \in SAL(v)\}.$$

Fig. 1 depicts a vertex v and its small ancestors.

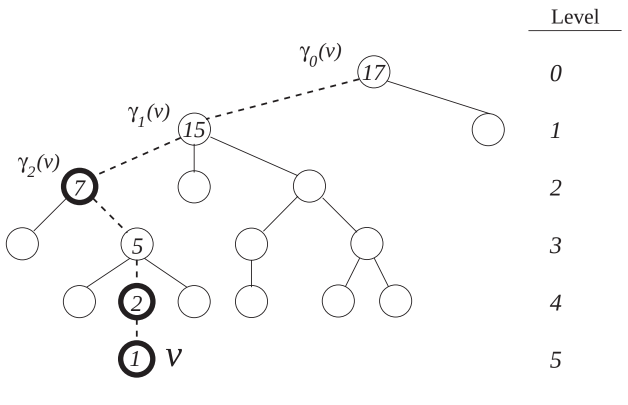

Fig. 1. Bold circles mark the small ancestors of v. Here $SAL(v) = \{2, 4, 5\}$. The number of vertices in $T(w)$ is displayed for every ancestor $w = \gamma_i(v)$ of v.

3.2. The LCA*-marker*

The labels are constructed as follows. As a preprocessing step, assign each v an *interval* $Int(v)$ as in the interval labeling scheme of [10], in addition to its identifier $I(v)$. This scheme is based on the following two steps. First, construct a *depth-first* numbering of the tree T, starting at the root, and assign each vertex $u \in T$ a depth-first number $DFS(u)$. Then, label a vertex u by the interval $Int(u) = [DFS(u), DFS(w)]$, where w is the last descendent of u visited by the DFS tour. The resulting interval labels are of size $O(\log n)$. What makes these interval labels useful for our purposes is the fact that they enjoy the following important property:

For every two vertices u and v of the tree T, $Int(v) \subseteq Int(u)$ iff v is a descendent of u in T.

Definition 3.2. For a vertex v and $1 \leqslant i < \text{depth}(v)$, the *i-triple* of v consists of the identifiers of its ancestors on levels $i-1$, i and $i+1$,

$$Q_i(v) = \langle\langle i-1, I(\gamma_{i-1}(v))\rangle,\ \langle i, I(\gamma_i(v))\rangle,\ \langle i+1, I(\gamma_{i+1}(v))\rangle\rangle.$$

In the second and main stage we do the following. For each vertex v, assign the label

$$L(v) = (I(v),\ Int(v),\ \{Q_i(v) \mid 1 \leqslant i < \text{depth}(v),\ i \in SAL(v)\}).$$

3.3. The LCA*-decoder*

Let us now describe the LCA-decoder $\mathcal{D}_{\mathtt{LCA}}$ which, given two vertex labels $L(v)$ and $L(w)$, infers the identifier $I(z)$ of their least common ancestor $z = \mathsf{LCA}(v, w)$.

Decoder $\mathcal{D}_{\mathtt{LCA}}$

```
1. If $Int(w) \subseteq Int(v)$   /* $v$ is an ancestor of $w$ */
   then return $I(v)$.
2. If $Int(v) \subseteq Int(w)$   /* $w$ is an ancestor of $v$ */
   then return $I(w)$.
3. /* $w$ and $v$ are unrelated */
   Extract from $L(v)$ and $L(w)$ the sets $SAL(v)$, $SAL(w)$, $SAI(v)$ and $SAI(w)$.
4. Let $\alpha$ be the highest level vertex in $SA(v) \cap SA(w)$.
         /* $\alpha$ is the least common small ancestor of $v$ and $w$ */
   Let $K$ be its level, i.e., $\alpha = \gamma_K(v) = \gamma_K(w)$.
5. If $I(\gamma_{K+1}(v)) \neq I(\gamma_{K+1}(w))$ then return $I(\alpha)$.
6. /* $\gamma_{K+1}(v) = \gamma_{K+1}(w)$ is also a common (yet large) ancestor of $v$ and $w$ */
         Let $i_v = \min\{i \in SAL(v) \mid i > K\}$,
             $i_w = \min\{i \in SAL(w) \mid i > K\}$,
7. If $i_v \leqslant i_w$ then extract $I(\gamma_{i_v-1}(v))$ from the $i_v$-triple $Q_{i_v}(v)$.
   Else extract $I(\gamma_{i_w-1}(w))$ from the $i_w$-triple $Q_{i_w}(w)$.
8. Return the extracted identifier.
```

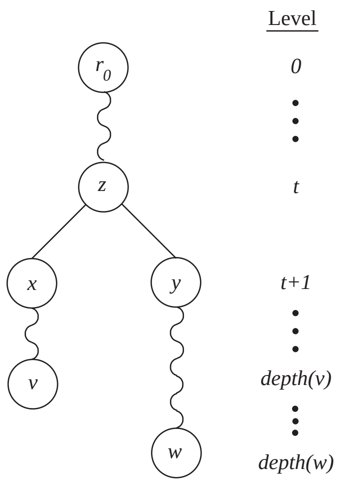

Fig. 2. The least common ancestor $z = \mathtt{LCA}(v, w)$ and its children x and y. Straight lines represent edges, and winding lines represent paths.

3.4. Correctness proof

Let us now prove the correctness of the labeling scheme. It is immediate to observe that if v is an ancestor of w or vice versa, then Steps 1, 2 of the decoder $\mathcal{D}_{\mathtt{LCA}}$ correctly find $\mathtt{LCA}(v, w)$. Hence hereafter we assume that neither of the above holds, i.e., $\mathtt{LCA}(v, w)$ is neither v nor w.

For the remainder of this section, denote $z = \mathtt{LCA}(v, w)$, and let its level be $t = \mathrm{depth}(z)$. Let x be the child of z on the path to v, and let y be the child of z on the path to w (see Fig. 2).

Lemma 3.3. $SA(v) \cap SA(w) = SA(z)$.

Let K and $\alpha = \gamma_K(v) = \gamma_K(w)$ be the level number and vertex selected in Step 4 of the algorithm. By the previous lemma, $\alpha \in SA(z)$, so $K \leqslant t$ and α is small, and hence $K \in SAL(z)$.

Now observe that if $K = t$ then we are done, since in this case the test done in Step 5 will necessarily succeed, and subsequently the algorithm will return $I(\alpha)$, which is the correct answer. Hence it remains to handle the case when $K < t$. In this case, the test of Step 5 will fail, and the execution will reach Steps 6 and 7. Our analysis of this case is based on showing that in this case the situation is that depicted in Fig. 3, namely, all the vertices on the path from α to the $\mathtt{LCA}$ z (including z itself) are large, and that necessarily either x or y (or both) must be small, hence justifying the choice made by the algorithm.

The following is obvious from the definitions.

Lemma 3.4. *Each vertex has at most one large child.*

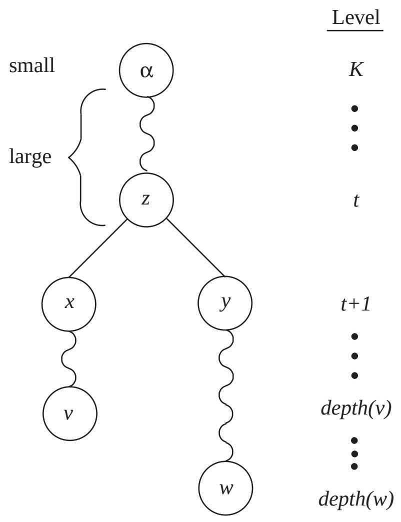

Fig. 3. The situation in case $K < t$.

Consequently, as $x \neq y$ and both are the children of the same parent z, at least one of them is small, hence we have:

Lemma 3.5. *$x \in SA(v)$ or $y \in SA(w)$.*

Lemma 3.6. *$i_v, i_w \geqslant t + 1$.*

Lemma 3.7. (1) *If $x \in SA(v)$ then $i_v = t + 1$,*
(2) *If $y \in SA(w)$ then $i_w = t + 1$.*

Combining the last three lemmas yields

Corollary 3.8. $\min\{i_v, i_w\} = t + 1$.

Hence the output returned by the algorithm in Step 7 is the correct one, $z = \gamma_t(v) = \gamma_t(w)$.

Lemma 3.9. *For every two vertices v and w, the decoder $\mathcal{D}_{\mathtt{LCA}}$ correctly deduces $\mathtt{LCA}(v, w)$ given $L(v)$ and $L(w)$.*

3.5. Analysis of the resulting label size

The following is obvious from the definitions.

Lemma 3.10. *In an n-vertex tree, every vertex v has at most $\log n$ small ancestors, i.e., $|SA(v)| \leqslant \log n$.*

It follows that each vertex v has at most $\log n$ i-triples $Q_i(v)$. The size of the resulting labels thus depends on the size of the identifiers used by the scheme. In particular, let $g(n)$ denote the maximum size of an identifier assigned to any vertex in any n-vertex tree. Clearly $g(n) = \Omega(\log n)$, hence each triple requires $\mathrm{O}(g(n))$ bits, and the entire label is of size $\mathrm{O}(g(n)\log n)$.

Theorem 3.11. *$\langle \mathcal{M}_{\mathtt{LCA}}, \mathcal{D}_{\mathtt{LCA}} \rangle$ is an LCA labeling scheme with labels of size $\mathrm{O}(g(n)\log n)$ for the class $\mathcal{T}_n$ of n-vertex trees with identifiers of size $g(n)$.*

Since log n-bit identifiers can always be chosen, we get the following corollary.

Corollary 3.12. $\mathcal{L}(\mathsf{LCA}, \mathcal{T}_n) = \mathrm{O}(\log^2 n)$.

Note that this is optimal, in the following sense.

Lemma 3.13. *If $\mathcal{T}_n$ has an LCA labeling scheme with $l(n) \cdot g(n)$-bit labels over $g(n)$-bit identifiers, then it has a $\mathtt{SepLevel}$ labeling scheme with $l(n) \cdot (g(n) + \log n)$-bit labels.*

Proof. Suppose that we are given an LCA labeling scheme $\langle \mathcal{M}_{\mathtt{LCA}}, \mathcal{D}_{\mathtt{LCA}} \rangle$ with $l(n) \cdot g(n)$-bit labels over $g(n)$-bit identifiers for $\mathcal{T}_n$. Define a $\mathtt{SepLevel}$ labeling scheme $\langle \mathcal{M}_{\mathtt{SepLevel}}, \mathcal{D}_{\mathtt{SepLevel}} \rangle$ for $\mathcal{T}_n$ as follows. Given a tree T, the $\mathtt{SepLevel}$-marker $\mathcal{M}_{\mathtt{SepLevel}}$ augments the identifier $I(v)$ of each vertex v into $I'(v)$ with an additional $\log n$ bit field containing v's depth, $\mathrm{depth}(v)$, and then invokes the LCA-marker $\mathcal{M}_{\mathtt{LCA}}$ to generate a labeling L for T. As the new identifiers are of size $g(n) + \log n$, the labeling L uses labels of size $l(n) \cdot (g(n) + \log n)$.

Consider two vertices v, w with $z = \mathsf{LCA}(v, w)$. The labels $L(v)$ and $L(w)$ allow the $\mathtt{SepLevel}$-decoder $\mathcal{D}_{\mathtt{SepLevel}}$ to deduce the identifier $I'(z)$ of z, and hence its depth $\mathrm{depth}(z)$, which is the separation level $\mathtt{SepLevel}(v, w)$. It follows that $\langle \mathcal{M}_{\mathtt{SepLevel}}, \mathcal{D}_{\mathtt{SepLevel}} \rangle$ is indeed a $\mathtt{SepLevel}$ labeling scheme. $\square$

Since $g(n) = \Omega(\log n)$, Corollary 2.2 implies

Corollary 3.14. *Any LCA labeling scheme for $\mathcal{T}_n$ requires some labels of $\Omega(\log^2 n)$ bits. Hence*

$$\mathcal{L}(\mathsf{LCA}, \mathcal{T}_n) = \Theta(\log^2 n).$$

4. `Center` labeling schemes

For every three vertices v_1, v_2, v_3 in a tree T, let $\mathtt{Center}(v_1, v_2, v_3)$ denote their *center*, namely, the unique vertex z such that the three paths connecting z to v_1, v_2 and v_3 are edge-disjoint (in fact, also vertex-disjoint except at z). See Fig. 4.

We now show that an LCA-marker can serve also as a $\mathtt{Center}$-marker, provided that the identifiers it uses are themselves ancestry and depth labelings, namely, the identifier $I(v)$ contains v's level $\mathrm{depth}(v)$ and any two identifiers $I(v)$ and $I(w)$ allow us to deduce whether

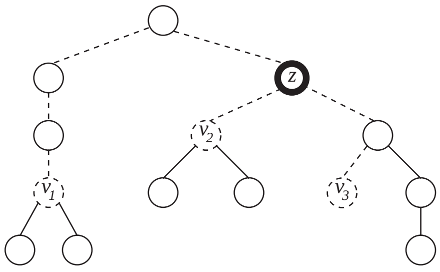

Fig. 4. Three vertices v_1, v_2, v_3 and their center z.

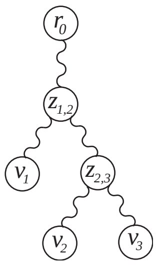

Fig. 5. The least common ancestors $z_{1,2} = z_{1,3}$ and $z_{2,3}$ in the case handled by Step 3 of the algorithm.

one of the two vertices is an ancestor of the other. As mentioned earlier, both requirements are achievable using identifiers of size $O(\log n)$. Hence the $\texttt{Center}$-marker $\mathcal{M}_{\texttt{Center}}$ will first pick such identifiers for the vertices, and then invoke the $\texttt{LCA}$-marker $\mathcal{M}_{\texttt{LCA}}$ described in the previous section for generating the labels.

Proving the claim requires us to present an algorithm for computing $\texttt{Center}(v_1, v_2, v_3)$ given the labels $L(v_1)$, $L(v_2)$ and $L(v_3)$.

4.1. *The* Center*-decoder*

For $1 \leqslant i \leqslant j \leqslant 3$, denote $z_{i,j} = \texttt{LCA}(v_i, v_j)$.

Decoder $\mathcal{D}_{\texttt{Center}}$

1. Compute $I(z_{1,2})$, $I(z_{1,3})$ and $I(z_{2,3})$.
2. If the three LCA's coincide then return $I(z_{1,2})$.
3. If exactly two LCA's coincide, say, $z_{1,3} = z_{1,2}$, then return the third, $I(z_{2,3})$.

Fig. 5 depicts the case handled by Step 3 of the algorithm.

4.2. *Correctness proof*

We rely on the following easy to verify facts.

Fact 4.1. (1) *For every three vertices v_1, v_2, v_3 in rooted tree, at least two of the three* LCA*'s $z_{1,2}$, $z_{1,3}$ and $z_{2,3}$ must coincide.*

(2) *If $z_{1,3} = z_{1,2} \neq z_{2,3}$, then $z_{2,3}$ is a descendent of $z_{1,3}$.*

As an easy corollary we get

Corollary 4.2. *For every three vertices v_1, v_2 and v_3, the* `Center`*-decoder $\mathcal{D}_{\texttt{Center}}$ correctly deduces* $\texttt{Center}(v_1, v_2, v_3)$ *given $L(v_i)$ for $i = 1, 2, 3$.*

Theorem 4.3. $\mathcal{L}(\texttt{Center}, \mathcal{T}_n) = \mathrm{O}(\log^2 n)$.

Finally, let us record the following fact for later use.

Lemma 4.4. *The* `Center` *labeling scheme allows us also to deduce the distance between* $z = \texttt{Center}(v_1, v_2, v_3)$ *and each v_i, $1 \leqslant i \leqslant 3$.*

5. `Steiner` labeling schemes

Our final two sections concern *weighted* trees and graphs. For a set W of vertices in a weighted graph G, their *Steiner tree*, denoted $T_S(W)$, is the minimum-weight subtree of G spanning the vertices of W, and its weight is denoted $\texttt{Steiner}(W)$. A `Steiner` labeling scheme can deduce $\texttt{Steiner}(W)$ given the labels $L(v)$ for every $v \in W$.

We now show that the `Center`-marker $\mathcal{M}_{\texttt{Center}}$ presented in the previous section can serve also as a `Steiner`-marker within a `Steiner` labeling scheme for the class of weighted trees. In particular, we rely also on the fact that in the labelings produced by the `Center`-marker $\mathcal{M}_{\texttt{Center}}$, the identifiers $I(v)$ of every vertex v provide depth(v).

Dealing with weighted graphs requires us, in particular, to use weighted measures of distance and depth. This means that when employing the `Center` labeling scheme of the previous section, which in turn makes use of our other schemes, the distance and depth functions used by the schemes must be the weighted ones. While this does not require any other change in the schemes, it does have some immediate implications on the size of the resulting labels, as explained later on.

5.1. *The* `Steiner` *decoder*

Given a `Center`-marker $\mathcal{M}_{\texttt{Center}}$ as in the previous section, and taking the `Steiner`-marker to be $\mathcal{M}_{\texttt{Steiner}} = \mathcal{M}_{\texttt{Center}}$, we now present a `Steiner`-decoder $\mathcal{D}_{\texttt{Steiner}}$ for computing the weight $\texttt{Steiner}(W)$ of the Steiner tree $T_S(W)$ for any vertex set W in T, given as input the labels $L(v)$ for every $v \in W$.

Let us first consider the case when $|W| = 3$, or $W = \{v_1, v_2, v_3\}$. In this case, the `Steiner`-decoder $\mathcal{D}_{\texttt{Steiner}}$ can simply deduce the center $z = \texttt{Center}(v_1, v_2, v_3)$, calculate the distances $d_i = \mathrm{dist}(v_i, z)$ for $1 \leqslant i \leqslant 3$ as in Lemma 4.4, and return $\omega(W) = d_1 + d_2 + d_3$.

Now suppose that W contains more than three vertices, $W = \{v_1, \ldots, v_q\}$ for $q > 3$. For every $3 \leqslant k \leqslant q$, let $W_k = \{v_1, \ldots, v_k\}$. Given the set W, the `Steiner`-decoder $\mathcal{D}_{\texttt{Steiner}}$ works iteratively, starting by computing $\omega(W_3)$ and adding the remaining vertices one at a time, computing $\omega(W_k)$ for $k = 4, \ldots, q$.

```
Decoder $\mathcal{D}_{\texttt{Steiner}}$
1. Deduce the center $z = \texttt{Center}(v_1, v_2, v_3)$.
2. Calculate the distances $d_i = \mathrm{dist}(v_i, z)$ for $1 \leqslant i \leqslant 3$ (as in Lemma 4.4).
3. Let $\omega(W_3) = d_1 + d_2 + d_3$.
4. For $k = 4$ to $q$ do:
   (a) For every $1 \leqslant i \leqslant j \leqslant k$, compute $z_{i,j} = \texttt{Center}(v_i, v_j, v_{k+1})$.
   (b) For every $1 \leqslant i < j \leqslant k$, compute $d_{i,j} = \mathrm{dist}(z_{i,j}, v_{k+1})$ (again, as in Lemma 4.4).
   (c) Let $1 \leqslant i' \leqslant j' \leqslant k$ be the pair minimizing $d_{i,j}$.
   (d) Let $\omega(W_{k+1}) = \omega(W_k) + d_{i',j'}$.
5. Return $\omega(W_q)$.
```

5.2. Correctness proof

Definition 5.1. For a subtree T' and a vertex v in T, let $p(v, T')$ denote the (unique) shortest path connecting v to some vertex of T'.

Lemma 5.2. *For every set of vertices $W = \{v_1, \ldots, v_k\}$ and vertex $v \notin W$, there exists a pair of vertices $v_i, v_j \in W$, connected by a path $P_{i,j}$ in T, such that $p(v, T_S(W)) = p(v, P_{i,j})$.*

Proof. Let $P' = p(v, T_S(W))$ and let z be the vertex of $T_S(W)$ that meets P'. (In case v itself occurs on $T_S(W)$, the path P' is of length 0, i.e., it consists of the single vertex $z = v$.) As $z \in T_S(W)$, z occurs on the path $P_{i,j}$ connecting *some* two leaves v_i and v_j of $T_S(W)$. Observing that the leaf set of $T_S(W)$ is a subset of W, the claim follows. □

Lemma 5.3. *For every set of vertices W in T, the* `Steiner`*-decoder $\mathcal{D}_{\texttt{Steiner}}$ correctly deduces $\omega(W)$ given $L(v)$ for every $v \in W$.*

As mentioned earlier, label sizes may be somewhat larger in the weighted case. Specifically, if *M*-bit edge weights are used, then the depth(v) field in the identifier $I(v)$ may require $\Theta(M + \log n)$ bits in the worst case. On the other hand, as $\mathrm{dist}(v_1, v_2, T) = \texttt{Steiner}(W)$ for any pair of vertices $W = \{v_1, v_2\}$, the lower bound of $\mathcal{L}(\mathrm{dist}, \mathcal{T}_{n,M}) = \Theta(M \log n + \log^2 n)$ established in [3] extends to the `Steiner` function as well. This yields the following result.

Theorem 5.4. $\mathcal{L}(\texttt{Steiner}, \mathcal{T}_{n,m}) = \Theta(M \log n + \log^2 n)$.

6. Approximate `Steiner` labeling schemes for general graphs

Our last result concerns approximate `Steiner` labeling schemes for the class $\mathcal{G}_{n,M}$ of arbitrary *n*-vertex graphs with *M*-bit edge weights. The presented scheme relies on the following relation between Steiner trees and minimum weight spanning trees, established in [5]. Consider a weighted graph $G = (V, E, \omega)$ and a set of vertices *W* in *G*. Let $G' = (W, E', \omega')$ denote the complete weighted graph defined on the vertex set *W* by setting

$\omega'(x, y) = \text{dist}(x, y, G)$ for every $x, y \in W$. Let $\text{MST}(W)$ denote the minimum weight of a spanning tree for G'. Then the following claim is established in [8] (in the proof of Theorem 1 therein).

Lemma 6.1 (*Kou et al. [8]*). $\texttt{Steiner}(W) \leqslant MST(W) \leqslant 2 \cdot \texttt{Steiner}(W)$.

Now consider an R approximate-distance labeling scheme $\langle \mathcal{M}'_{\text{dist}}, \mathcal{D}'_{\text{dist}} \rangle$ for $\mathcal{G}_{n,M}$. The same marker algorithm $\mathcal{M}'_{\text{dist}}$ can also be employed as part of a $2R$-approximate `Steiner` labeling scheme for $\mathcal{G}_{n,M}$, using the following decoding procedure.

Approximate Decoder $\mathcal{D}'_{\texttt{Steiner}}$

Given the label $L(v)$ of every vertex $v \in W$, the decoder does the following.

1. Using the distance decoder $\mathcal{D}'_{\text{dist}}$, calculate a distance estimate $\tilde{\omega}(x, y)$ for every $x, y \in W$.
2. Construct a minimum-weight spanning tree T' for the complete graph $G' = (W, E', \tilde{\omega})$.
3. Calculate its weight $\text{MST}(W) = \tilde{\omega}(T')$.
4. Return $\text{MST}(W)$.

The following is immediate from Lemma 6.1.

Lemma 6.2. *The decoder* $\mathcal{D}'_{\texttt{Steiner}}$ *yields a* $2R$*-approximation for* $\texttt{Steiner}(W)$.

Corollary 6.3. *If* $\mathcal{G}_{n,M}$ *enjoys an R-approximate distance labeling scheme, then it also enjoys a* $2R$*-approximate* `Steiner` *labeling scheme with labels of the same size.*

We now rely on the approximate-distance labeling scheme for the class $\mathcal{G}_{n,M}$, due to [9].

Lemma 6.4 (*Peleg [9]*). *There exists an* $8 \log n$*-approximate distance labeling scheme for the class* $\mathcal{G}_{n,M}$ *with labels of size* $O((M + \log n) \log^2 n)$.

Corollary 6.5. *The class* $\mathcal{G}_{n,M}$ *enjoys a* $16 \log n$*-approximate* `Steiner` *labeling scheme with labels of size* $O((M + \log n) \log^2 n)$.

Acknowledgements

I am grateful to Michal Katz and Nir Katz for their helpful comments and suggestions.

References

[1] M.A. Breuer, Coding the vertexes of a graph, IEEE Trans. Inform. Theory IT-12 (1966) 148–153.
[2] M.A. Breuer, J. Folkman, An unexpected result on coding the vertices of a graph, J. Math. Anal. Appl. 20 (1967) 583–600.
[3] C. Gavoille, D. Peleg, S. Pérennes, R. Raz, Distance labeling in graphs, in: Proc. 12th ACM-SIAM Symp. on Discrete Algorithms, ACM-SIAM, January 2001, pp. 210–219.

[4] S. Kannan, M. Naor, S. Rudich, Implicit representation of graphs, in: Proc. 20th ACM Symp. on Theory of Computing, May 1988, pp. 334–343.
[5] M. Katz, N.A. Katz, A. Korman, D. Peleg, Labeling schemes for flow and connectivity, in: Proc. 13th ACM-SIAM Symp. on Discrete Algorithms, ACM-SIAM, January 2002, pp. 927–936.
[6] M. Katz, N.A. Katz, D. Peleg, Distance labeling schemes for well-separated graph classes, in: Proc. 17th Symp. on Theoretical Aspects of Computer Science, February 2000, pp. 516–528.
[7] A. Korman, D. Peleg, Y. Rodeh, Labeling schemes for dynamic tree networks, in: Proc. 19th Symp. on Theoretical Aspects of Computer Science, March 2002, pp. 76–87.
[8] L. Kou, G. Markowsky, L. Berman, A fast algorithm for Steiner trees, Acta Inform. 15 (1984) 141–145.
[9] D. Peleg, Proximity-preserving labeling schemes and their applications, in: Proc. 25th Internat. Workshop on Graph-Theoretic Concepts in Computer Science, June 1999, pp. 30–41.
[10] N. Santoro, R. Khatib, Labelling and implicit routing in networks, Comput. J. 28 (1985) 5–8.