Laboratoire Bordelais de Recherche en Informatique
UMR 5800 - Université Bordeaux I, 351, cours de la Libération,
33405 Talence CEDEX, France

Research Report RR-1263-01

Distance Labeling Scheme and Split Decomposition

by Cyril Gavoille and Christophe Paul

November, 2001

Distance Labeling Scheme and Split Decomposition

Cyril Gavoille[*] Christophe Paul[†]

November 12, 2001

Abstract

A distance labeling scheme is a distributed data-structure designed to answer queries about distance between any two vertices of a graph G. The data-structure consists in a label $L(x,G)$ assigned to each vertex x of G such that the distance $d_G(x,y)$ between any two vertices x and y can be estimated as a function $f(L(x,G),L(y,G))$. In this paper we combine several types of distance labeling schemes and split decomposition of graphs. This yields to optimal label length schemes for the family of distance-hereditary graphs and for other families of graphs, allowing distance estimation in constant time once the labels have been constructed.

Keywords: distance labeling scheme, split decomposition, distance-hereditary graphs.

1 Introduction

In this paper we deal with undirected simple and connected graphs. The distance $d_G(x,y)$ between two vertices x,y of a graph G is the minimum length (or the minimum cost in the case of weighted graphs) of a path connecting x and y in G. The problem we are interested in is the distance labeling problem, that is problem of designing of a distributed data-structure that enable us to answer queries about distances between any two vertices of a graph G. More precisely:

> How to label the vertices of a graph G in such a way the distance between any two vertices x and y of G can be computed or approximated by inspecting the labels of x and of y, and without any other information source?

Clearly one can encode all the desired information without any restriction on the label length of the vertices. On the other hand short labels, say of $(\log n)^c$ bits[1] per vertex where c is a constant and n the number of vertices of the graph,

[*]Laboratoire Bordelais de Recherche en Informatique, Université Bordeaux I. *gavoille@labri.fr*

[†]Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier. *paul@lirmm.fr*

[1]The function log denotes the logarithm in base two.

is not doable for all n-vertex graphs since there are more than $\exp(n(\log n)^c)$ graphs with n vertices, and since the list of the labels suffices to entirely rebuild the graph by testing the distances. Our goal is to design labeling schemes that assigns relatively short labels to all n-vertex graphs of a given family of graphs, and such that distance queries can be performed quickly, idealy in constant time. In addition, we are looking for labeling schemes that are polynomially constructible.

This labeling problem has been formally introduced in [27], but early appears informally in [16] about the Squashed Cube Embedding problem. Distance labeling is a natural generalization of the problem of finding *implicit representation* of graphs [4, 19, 28] that consists in assigning labels of the vertices of a graph such that the adjacency between to vertices can be tested using the corresponding labels. Other functions can be computed by suitable labeling schemes: ancestry and small distances in trees [2, 20, 21], near-common ancestor in trees [1], and other functions [22, 26]. A recent overview on compact labeling schemes can be founded in [14].

Definition 1.1 ([13]) *Given a family $\mathcal{F}$ of connected graphs, an (s,r)*-approximate distance labeling scheme *on $\mathcal{F}$, (s,r)-approximate DLS for short, is a pair $\langle L, f\rangle$, where L is call the* labeling function *and f the* distance decoder*, such that for any $G \in \mathcal{F}$ and any pair x,y of distinct vertices of G, $L(x,G) \in \{0,1\}^*$, and*

$$d_G(x,y) \leqslant f(L(x,G),L(y,G)) \leqslant s \cdot d_G(x,y) + r \ .$$

An $(1,r)$-approximate DLS is called *r-additive*, while an $(s,0)$-approximate is called *s-multiplicative*. For convenience, 0-additive and 1-multiplicative DLS are called *exact* DLS.

In this paper we are interesting in constructing DLS for several family of graphs, in particular distance-hereditary graphs. This latter family have bounded *clique-width*, a graph parameter related to a particular graph decomposition (see [9] for more details about this decomposition). The reader interested in that labeling problem has to keep in mind the following deep result due to [10].

Theorem 1.2 ([10]) *The family of n-vertex bounded clique-width graphs enjoys an exact distance labeling scheme using labels of length $O(\log^2 n)$ bits. Moreover the distance can be computed in $O(\log^2 n)$ time.*

In our point of view, this result has to be considered carefully. It shows the existence of exact DLS with relatively short labels and fast distance queries. However the scheme is not explicit (the decomposition must be given) and the hidden constants in the label length and in the query time complexity are huge (a stack of exponentials). Note that computing the clique-width of a graph (and its optimal decomposition) is a wide open problem. For fixed $k > 3$, it is not known whether there exists a polynomial time algorithm that recognizes graphs of clique-width at most k (recently [8] have showed that the case $k = 3$ is polynomial). So finding an explicit and polynomial time constructible distance labeling scheme with constant time query for bounded clique-width graphs is a challenging problem.

In this paper, for the family of n-vertex distance-hereditary graphs, we explicitly construct (in polynomial time) an exact DLS with $O(\log^2 n)$ bit labels, and a $(1+o(1))$-multiplicative DLS with $O(\log n \log\log n)$ bit labels. We show that our schemes are asymptotically optimal w.r.t. the label length and the approximation. Moreover the distance estimate can be computed in constant time under a standard word-RAM model providing standard arithmetic operations on $O(\log n)$ bit words in constant time. We show how the *split decomposition* of a graph [11] can be used to design efficient labeling schemes. Our technique allows to combine different types of schemes. Applying this technique, we show that the optimal schemes for trees obtained by [13, 15] can be generalized to distance-hereditary graphs. Our results are extended to other classical families of graphs.

The paper is organized as follows. Section 2 presents how the split decomposition can be used to compute an tree-like auxiliary graph in which the distances in the original graph can be refound. Then Section 3 shows how different approximate DLS can be arranged together when a graph has a suitable *block-partition*, a graph decomposition in 2-connected components. The technique is applied in the last section to some graph families. First we illustrate our technique on a simple example of graph family, namely the block-graphs. Then we apply the technique to graph families related to distance-hereditary graphs, namely all the graphs G whose the distance between x, y in any connected induced subgraph is at most $s \cdot d_G(x,y) + r$. In particular we show that distances in a distance-hereditary graph ($s = 1$ and $r = 0$) can be reconstructed from distances taken from two block-graphs of linear size.

2 On the Split Decomposition

By the use of the split decomposition on a graph G, we show in this section how to produce a new graph T and a tree $\widetilde{T}$ from G such that the distances in G can be deduced from the distances in T and in $\widetilde{T}$, expecting that efficient DLS on T exists. The split decomposition has been originally introduced by [11].

In a graph G, the *neighborhood* of a vertex $x \in V(G)$ is denoted by $N(x)$. The neighborhood of a set of vertices $S \subseteq V(G)$ is the set $N(S) = \bigcup_{s \in S} N(s) \backslash S$.

Definition 2.1 ([11]) *A* split *in a graph $G = (V, E)$ is a partition of V into two sets V_1 and V_2 such that $|V_1| \geqslant 2$, $|V_2| \geqslant 2$, and such that for all $x_1 \in \widetilde{V}_1 = V_1 \cap N(V_2)$ and $x_2 \in \widetilde{V}_2 = V_2 \cap N(V_1)$, $\{x_1, x_2\} \in E$.*

A graph without split is called *prime*, in particular a graph with 3 vertices or less is prime. When a graph G has a split (V_1, V_2), it can be decomposed into two graphs G_1 and G_2 as follows: G_1 (resp. G_2) is induced by $V_1 \cup \{v_1\}$ (resp. $V_2 \cup \{v_2\}$) where v_1 (resp. v_2) is a virtual vertex adjacent to all the vertices of $\widetilde{V}_1$ (resp. $\widetilde{V}_2$). One can then decide to decompose recursively G_1 and/or G_2. The graphs obtained at the end of a split decomposition are the *split-components*.

It is important to note that each split-component is isomorphic to an induced subgraph of G. In particular split-components that are prime graphs are induced subgraphs of G. A split decomposition is not unique, however, it has been shown in [11] that every connected graph have a unique split decomposition into a minimal number of split-components that are cliques, stars (i.e., a tree of depth 1), or prime graphs. This unique split decomposition is obtained if during the decomposition maximal splits are chosen (maximal w.r.t. the inclusion of $\widetilde{V_1} \cup \widetilde{V_2}$).

Let G be an unweighted n-vertex graph. From G and from any split decomposition applied to G we construct a graph T, called *tree-like graph* of G, obtained recursively as follows (cf. Fig. 1 and 2): If G is a split-component (e.g. G is prime), then T is G. Otherwise, let G_1, G_2 be the two graphs into which G is decomposed. Then T consists in joining the graphs T_1 and T_2, the tree-like graphs of G_1 and G_2, by connecting the vertices v_1 with v_2 by an edge called *virtual edge.*

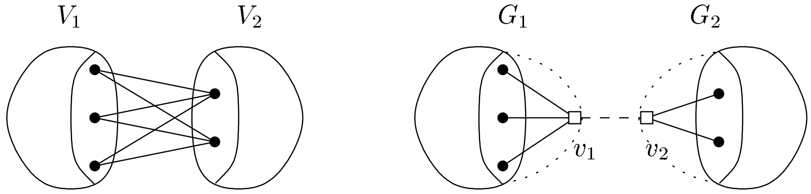

Figure 1: A step in a split decomposition and the tree-like graph construction.

A similar construction of T appears in [24]. The split-components of G are exactly the connected components – called hereafter simply *component of* T – obtained by removing all the virtual edges of T. Fig. 2 shows three tree-like graphs of a graph G, the last one denoted T has 5 components.

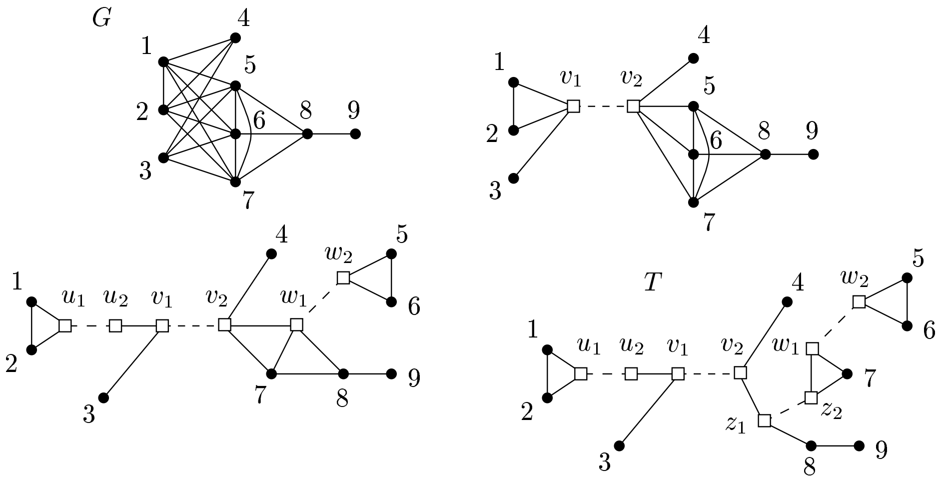

Figure 2: A tree-like graph T obtained from G with the 4 splits: $(V_1, V_2) = (\{1,2,3\}, \{4,5,6,7,8,9\})$, then $(U_1, U_2) = (\{1,2\}, \{v_1, 3\})$ and $(W_1, W_2) = (\{5,6\}, \{4, v_2, 7, 8, 9\})$, and finally $(Z_1, Z_2) = (\{w_1, 7\}, \{4, v_2, 8, 9\})$. The component $\{4, v_2, z_1, 8, 9\}$ can still be decomposed.

For every tree-like graph T of G, we define $\widetilde{T}$ be the graph obtained from T by contracting each component into a single vertex. So the vertices of $\widetilde{T}$

are seen as the components of T, and its edges are the virtual edges of T. By construction of T, $\widetilde{T}$ is a tree. In the example depicted on Fig. 2, $\widetilde{T}$ is a path of length 4.

Lemma 2.1 *Let T be a tree-like graph of G, let x and y be two vertices of G, and let X (resp. Y) be the component of T containing x (resp. y). Then, $d_G(x,y) = d_T(x,y) - 2d_{\widetilde{T}}(X,Y)$.*

Proof: The property holds if $T = G$, in particular whenever $n \leqslant 3$. For an induction on n, assume that $n > 3$, and that the property holds for every tree-like graph of a graph of at most $n-1$ vertices. Finally, assume that T consists in joining the tree-like graphs T_1 and T_2 for a split (V_1, V_2) of G.

Let $n_1 = |V_1|$ and $n_2 = |V_2|$. Let v_1 (resp. v_2) be the vertex of T_1 (resp. T_2) connected to all the vertices of $\widetilde{V_1}$ (resp. $\widetilde{V_2}$). Note that T_1 (resp. T_2) is a tree-like graph of a graph G_1 (resp. G_2) with $n_1 + 1$ vertices (resp. with $n_2 + 1$ vertices). Since $n = n_1 + n_2$ and $n_1, n_2 \geqslant 2$, it follows that $n_1+1, n_2+1 \leqslant n-1$, thus that Lemma 2.1 holds for G_1 and G_2. Let P be a shortest path from x to y in G. We consider two cases.

Case 1: $x, y \in V_1$ (or both in V_2).
Let us first show that P is isomorphic to a path P' wholly contained in G_1. If $P \subseteq V_1$, then one can set $P' = P$ and we are done. If $P \cap V_2 \neq \varnothing$, then P contains one (and only one) vertex of V_2 and P has the form $P = (x, A, u, w, v, B, y)$ where A and B are sequences of vertices of V_1, $u, v \in \widetilde{V_1}$, and $w \in \widetilde{V_2}$. Since v_1 is adjacent to u and v in G_1, P is isomorphic to $P' = (x, A, u, v_1, v, B, y)$, a path wholly contained in G_1.

Since P is isomorphic to a path P' wholly contained in G_1, we have $d_{G_1}(x,y) \leqslant d_G(x,y)$. However, since G_1 is isomorphic to an induced subgraph of G, we have $d_{G_1}(x,y) = d_G(x,y)$. By induction, $d_{G_1}(x,y) = d_{T_1}(x,y) - 2d_{\widetilde{T_1}}(X,Y)$. Since the edge joining T_1 to T_2 is a bridge, $d_{T_1}(x,y) = d_T(x,y)$ and $d_{\widetilde{T_1}}(X,Y) = d_{\widetilde{T}}(X,Y)$. It follows that $d_{G_1}(x,y) = d_T(x,y) - 2d_{\widetilde{T}}(X,Y)$, hence that $d_G(x,y) = d_T(x,y) - 2d_{\widetilde{T}}(X,Y)$, completing the proof of Case 1.

Case 2: $x \in V_1$ and $y \in V_2$ (or the reverse).
Note that removing the edges between $\widetilde{V_1}$ and $\widetilde{V_2}$ disconnects G. Thus P has the form $P = (x, A, u, w, B, y)$ where $A \subseteq V_1$, $B \subseteq V_2$, $u \in \widetilde{V_1}$, and $w \in \widetilde{V_2}$. It follows that

$$d_G(x,y) \;=\; d_{G_1}(x,u) + d_{G_2}(w,y) + 1\ . \tag{1}$$

By Case 1, $d_{G_1}(x,u) = d_T(x,u) - 2d_{\widetilde{T}}(X,U)$ and $d_{G_2}(w,y) = d_T(w,y) - 2d_{\widetilde{T}}(W,Y)$, where U (resp. W) is the component of T containing u (resp. w). $\{U, W\}$ is an edge of $\widetilde{T}$ that disconnects the component X from Y, thus $d_{\widetilde{T}}(X,Y) = d_{\widetilde{T}}(X,U) + 1 + d_{\widetilde{T}}(W,Y)$. Hence

$$d_{\widetilde{T}}(X,U) + d_{\widetilde{T}}(W,Y) \;=\; d_{\widetilde{T}}(X,Y) - 1\ .$$

The vertex v_1 is connected to u in T_1, v_2 is connected to w in T_2, and $\{v_1, v_2\}$ is a bridge of T, thus (u, v_1, v_2, w) is a shortest path in T. Hence $d_T(x,y) = d_T(x,u) + d_T(u,w) + d_T(w,y)$, that implies

$$d_T(x,u) + d_T(w,y) \;=\; d_T(x,y) - 3\ .$$

Plugging in Eq. 1, we have:

$$\begin{aligned}
d_G(x,y) &= \big(d_T(x,u) - 2d_{\widetilde{T}}(X,U)\big) + \big(d_T(w,y) - 2d_{\widetilde{T}}(W,Y)\big) + 1 \\
&= d_T(x,u) + d_T(w,y) - 2\left(d_{\widetilde{T}}(X,U) + d_{\widetilde{T}}(W,Y)\right) + 1 \\
&= d_T(x,y) - 3 - 2\left(d_{\widetilde{T}}(X,Y) - 1\right) + 1 \\
&= d_T(x,y) - 2d_{\widetilde{T}}(X,Y),
\end{aligned}$$

completing the proof of Case 2 and of Lemma 2.1. □

Lemma 2.2 *Let T be a tree-like graph of G. Then, T contains no more than* $\max\{n-3,0\} < n$ *virtual edges.*

Proof: The property holds if $T = G$, in particular whenever $n \leqslant 3$. For an induction on n, assume that $n > 4$, and that the property holds for every tree-like graph of a graph of at most $n-1$ vertices. Finally, assume that T consists in joining the tree-like graphs T_1 and T_2 for a split (V_1, V_2) of G. Let $n_1 = |V_1|$ and $n_2 = |V_2|$. As already said in the proof of Lemma 2.1, T_1, T_2 are tree-like graphs of graphs with $n_1+1, n_2+1 \leqslant n-1$ vertices. So by induction T_1 and T_2 have respectively at most n_1+1-3 and n_2+1-3 virtual edges. It follows that T has at most $1+(n_1+1-3)+(n_2+1-3) = n-3$ virtual edges, completing the proof. □

It follows that T has less than $2n$ virtual vertices, and less than $3n$ vertices.

3 Distance Labeling Schemes Combination

It is known that there exist some n-vertex graphs that need exact distance labels of $\Theta(n)$ bits [15]. Therefore we do not expect the design of short distance labels by the use of split decomposition in the general case, despite the uniqueness decomposition result of [11]. However, there is hope for families of graphs whose prime components have suitable properties, like bounded tree-width, bounded chordality, planarity, etc. Widely based on the schemes of trees [13], we show in this section how to combine DLS of prime components in order to derive DLS for the whole graph. For that purpose we need to investigate labeling schemes on graphs having a suitable *block-partition*, notion we introduce below. Recall that a *cut-vertex* of a graph G is a vertex whose deletion disconnects G in two or more non-empty connected components.

Definition 3.1 *A* block-partition *of a graph G is a set of subgraphs of G, called* blocks *of G, such that: 1) every vertex of G is contained in a block; 2) every edge of G is contained in exactly one block; 3) the blocks intersect at some cut-vertices only.*

Note that if two blocks intersect they have in common exactly one cut-vertex. Given a block-partition P we associate a tree T defined as follows:

$$\begin{aligned}
V(T) &= P \cup C, \text{ where } C = \{V(B) \cap V(B') \mid B \neq B' \in P\} \\
E(T) &= \{\{B, X\} \mid B \in P, X \in C \cap V(B)\}.
\end{aligned}$$

Roughly speaking T is obtained from P by replacing every block by a star whose leaves are the intersections with all its intersecting blocks (see Fig. 3). Note that we made no restrictions on the blocks. They may contain, for instance, some cut-vertices of the graph that are not in the set C.

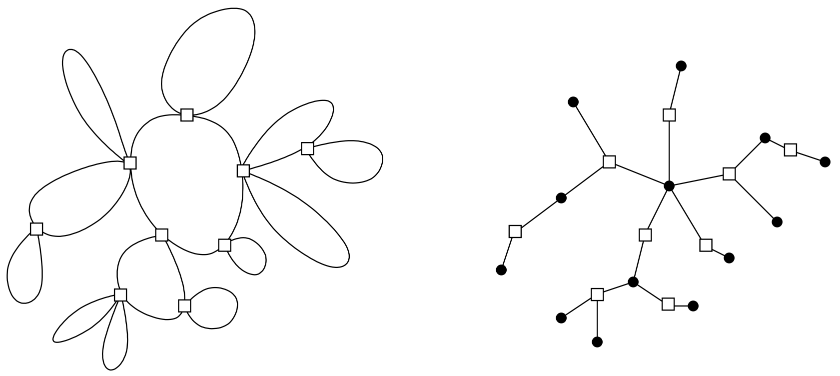

Figure 3: A block-partition of an arbitrary graph, and its associated tree.

Lemma 3.1 $|V(T)| \leqslant 2|P| - 1$ *and* $|P| \leqslant n - 1$, *where* $n = |V(G)|$.

Proof: Let us root T at a vertex of P. Consider $c \in C$. Because c is the intersection of two blocks, c is not a leaf in T. Thus, in T, every non-root block of P has an unique father of C, proving that $|P| - 1 \geqslant |C|$. Thus, $|V(T)| \leqslant 2|P| - 1$. Consider now any spanning tree S of G (G is connected). By Rule 2 of Definition 3.1, every edge of S belongs to at most one block, thus $|P| \leqslant n - 1$. $\square$

In this section we deal with weighted or unweighted graphs with n vertices. Given an integer $w > 0$, let us denoted by w*-diameter graph* any weighted graph whose edge cost is a non-null integer, and such that the weighted diameter is at most w. Unweighted n-vertex graphs are n-diameter graphs. Hereafter, we assume a word-RAM model of computation providing constant time standard arithmetic operations on $O(\log n)$ bit words.

Given a real $s \geqslant 1$ and an integer $w > 0$, a pair of functions $\langle \lambda, \delta \rangle$ is an (s, w)*-estimator* if $\lambda : \{1, \ldots, w\} \to \{0,1\}^\alpha$, $\delta : \{0,1\}^\alpha \to \mathbb{N}$, and for every $x \in \{1, \ldots, w\}$, $x \leqslant \delta(\lambda(x)) \leqslant s\,x$. Intuitively, we think of λ as a function "compacting" x, (typically $\alpha \ll \log w$) and of δ as a decoding function attempting to reconstruct x from $\lambda(x)$. The *size* of the estimator is α, and its *time complexity decoder* is the worst-case time complexity of the function δ. For every w, there is a trivial $(1, w)$-estimator of size $\lceil \log w \rceil$ consisting in representing in binary each integer of $\{1, \ldots, w\}$. More sophisticated estimators can be constructed. For instance, in [13] it is built the following range of estimators:

Lemma 3.2 ([13]) *For all* k *and* $m \in \{0, \ldots, k\}$, *there exists a (polynomially constructible)* $(1 + 2^{-m}, 2^k)$*-estimator of size* $\alpha = m + \lceil \log(k - m + 1) \rceil$ *and of constant time decoder if* $k = O(\log n)$.

Choosing $k = \lceil \log w \rceil$ and $m = \lceil \log k \rceil$, Lemma 3.2 gives an $(1 + 1/\log w, w)$-estimator of size $2 \log\log w + O(1)$, and of constant time decoder if $\log w =$

$O(\log n)$.

The main result of this section is the following theorem:

Theorem 3.2 *Let α be the size of an (s,w)-estimator with constant time decoder. Let $\mathcal{F}$ and $\mathcal{B}$ be two families of w-diameter graphs with at most n-vertices such that every $G \in \mathcal{F}$ has a block-partition $P \subset \mathcal{B}$ such that $|P| \leqslant p$. Assume that $\mathcal{B}$ has an (s,r)-approximate DLS with labels of length at most β and with a constant time distance decoder. Then, $\mathcal{F}$ has an (s,r)-approximate DLS using labels of length at most $(\alpha+\beta)\log(4p) + O(\log n)$ bits and with a constant time distance decoder.*

Intuitively, Theorem 3.2 means that distances in G can be approximated by the combination of $O(\log p)$ DLS, each one being defined inside a block of G. It is important to note that despite the fact that each block can be of unbounded size, β can be of constant size. In Definition 1.1, it is just required that distances can be computed from labels of *distinct* vertices, two vertices can have the same label. For instance, if each block is an unweighted clique with $\Omega(\sqrt{n})$ vertices, then $\Omega(\log n)$ is not a lower bound for β: constant length labels suffice to compute the distances in cliques of arbitrary size! To prove Theorem 3.2 we need some preliminaries.

The *separator* of a rooted tree T is the vertex s obtained by performing a traversal of T from its root, and defined as follows: Let u be the current vertex (initially u is the root). If all the connected components of $T \setminus \{u\}$ have $n/2$ vertices or less, then $s = u$. Otherwise update u to the unique child of u having more than $n/2$ descendants. It is not difficult to see that the above procedure ends on a vertex s such that $T \setminus \{s\}$ is composed of connected components of at most $n/2$ vertices. Moreover, in the case of a rooted tree, s is unique and can be founded in linear time.

Given a vertex x of T (rooted and with n vertices), the *separator-sequence of* x, is the sequence of vertices $(s_1,\ldots,s_h)$ such that: 1) s_1 is the separator of T; 2) either $x = s_1$ and $h = 1$, or $(s_2,\ldots,s_h)$ is the separator-sequence of the tree $T' = T \setminus \{s_1\}$ containing x and rooted at the neighbor of s_1 contained in T'. The separator-sequence of all the vertices of T can be constructed in $O(n \log n)$ time, and we have $h \leqslant 1 + \log n$.

By construction, every path from x to y in T (with $x \neq y$) must traverse a vertex s common to the separator-sequence of x and of y. More precisely, if the separator-sequence of x is $(s_1 = x_1, x_2, \ldots, x_{h_x} = x)$, and the separator-sequence of y is $(s_1 = y_1, y_2, \ldots, y_{h_y} = y)$, then we have: $d_T(x,y) = d_T(x,x_{i_0}) + d_T(x_{i_0},y)$, where i_0 is the largest index i such that $x_i = y_i$. Actually, i_0 is the length of the longest common prefix of the separator-sequence of x and of y, that we denote hereafter by $\mathrm{lcp}(x,y)$.

Assume that T is a w-diameter tree, and assume that some (s,w)-estimator of size α is given. A simple s-multiplicative DLS on T can be described. The vertex x has a label S_x representing its separator-sequence and a table D_x of h_x binary strings such that $D_x[i] = \lambda(d_T(x,x_i))$ for every $i \in \{1,\ldots,h_x - 1\}$ (since $d_T(x,x_{h_x}) = 0$, we can save the last entry of D_x). Similarly, y has a label S_y and a table $D_y[j] = \lambda(d_T(y,y_j))$ for every $j \in \{1,\ldots,h_x - 1\}$. From the above discussion, $d_T(x,y)$ can be estimated as follows:

1. compute $i_0 = \text{lcp}(x,y)$ from S_x and S_y;
2. compute $\tilde{d}(x,y) = \delta(D_x[i_0]) + \delta(D_y[i_0])$.

We check that $d_T(x,y) \leqslant \tilde{d}(x,y) \leqslant s \cdot d_T(x,y)$. As shown in [13], the labels S_x and S_y can be coded (in polynomial time) on $O(\log n)$ bits if we are interested only in extracting $\text{lcp}(x,y)$ in constant time[2]. Therefore, provided that the (s,w)-estimator has constant time decoder, one can estimate the distances in constant time in n-vertex trees using labels of $\alpha \log n + O(\log n)$ bits. This is mainly the scheme used for trees in [13].

One need to extend this construction to complete the proof of Theorem 3.2.

Proof: (of Theorem 3.2) Let $\langle \lambda, \delta \rangle$ be an (s,w)-estimator of size α and with constant time decoder. Let $\langle L_{\mathcal{B}}, f_{\mathcal{B}} \rangle$ be an (s,r)-approximate DLS on $\mathcal{B}$ with labels of length at most β and with a constant time distance decoder. Consider an n-vertex w-diameter graph $G \in \mathcal{F}$ having a block-partition $P \subset \mathcal{B}$. Let $C = \{V(B) \cap V(B') \mid B \neq B' \in P\}$. Let T be the associated tree of P, and let us root T in an arbitrary vertex.

We construct an approximate DLS $\langle L, f \rangle$ on $\mathcal{F}$ as follows. For every $x \in V(G)$, we set:

$$L(x,G) \;=\; (h_X, S_X, M_X, D_x, P_X)$$

where $X \in P$ is a block containing x (if several blocks contains x, choose one arbitrarily). h_X is the length of the separator-sequence of X in T. S_X is the label assigned to the separator-sequence of X that allows to efficiently compute $\text{lcp}(X,Y)$ from the corresponding label of any vertex Y of T. Assume that the separator-sequence of X is $(X_1, \ldots, X_{h_X})$. From this sequence we construct a new sequence of vertices of G, $(x_1, \ldots, x_{h_X})$, such that, for every $i \in \{1, \ldots, h_X\}$: if $X_i \in C$, then $x_i = X_i$; $x_{h_X} = x$; and if $X_i \in P$ and $i \neq h_X$, then x_i is the unique vertex of C that neighbors X_i on the path from X to X_i in T (x_i exists since $X, X_i \in P$ and $X \neq X_i$). M_X is a table such that, for every $i \in \{1, \ldots, h_X - 1\}$: $M_X[i] = 1$ if $x_i = X_i$ (i.e., if $X_i \in C$), and $M_X[i] = 0$ otherwise. Finally, D_x and P_X are tables such that: for every $i \in \{1, \ldots, h_X - 1\}$, $D_x[i] = \lambda(d_G(x, x_i))$, and for every $i \in \{1, \ldots, h_X\}$, $P_X[i] = L_{\mathcal{B}}(x_i, X_i)$ if $X_i \in P$ (if $X_i \in C$, $P_X[i]$ is not defined – the entry is empty).

Let x, y be any two distinct vertices of G with labels:

$$\begin{aligned} L(x,G) &= (h_X, S_X, M_X, D_x, P_X) \\ L(y,G) &= (h_Y, S_Y, M_Y, D_y, P_Y) \end{aligned}$$

The distance decoder f is computed as follows (for short, we let $\tilde{d}(x,y) = f(L(x,G), L(y,G))$):

1. compute $i_0 = \text{lcp}(X,Y)$ from S_X and S_Y;
2. If $i_0 = h_X = h_Y$, then return $\tilde{d}(x,y) = f_{\mathcal{B}}(P_X[i_0], P_Y[i_0])$;
3. If $M_X[i_0] = 1$, then return $\tilde{d}(x,y) = \delta(D_x[i_0]) + \delta(D_y[i_0])$;

[2] The idea is the following. Instead of storing $(s_1, \ldots, s_h)$ in extenso with $h \log n = O(\log^2 n)$ bits, we associate to each s_i a number $c_i \geqslant 1$ that identifies the subtree containing x whenever s_i is removed from the current tree component. Sorting the subtrees by size one can show that $\prod_i c_i \leqslant n$, and thus one can encode $(c_1, \ldots, c_h)$ on $O(\log n)$ bits.

4. return $\tilde{d}(x,y) = \delta(D_x[i_0]) + f_{\mathcal{B}}(P_X[i_0], P_Y[i_0]) + \delta(D_y[i_0])$.

Let us show that $\langle L, f\rangle$ is an (s,r)-approximate DLS, that is $d_G(x,y) \leqslant \tilde{d}(x,y) \leqslant s \cdot d_G(x,y) + r$.

1. If $i_0 = h_X = h_Y$, then $X = Y$ and x and y belongs to the same block B of P. Therefore the local (s,r)-approximate DLS on $\mathcal{B}$, namely $\langle L_{\mathcal{B}}, f_{\mathcal{B}}\rangle$, can be used and the approximation follows.
2. If $M_X[i_0] = 1$, then $X_{i_0} = Y_{i_0} \in C$ is a cut-vertex that separates x from y. In other words, $d_G(x,y) = d_G(x, X_{i_0}) + d_G(y, Y_{i_0})$. The first term is stored in the table D_x and the second in the table D_y. Therefore the result follows by using $\langle \lambda, \delta\rangle$.
3. If $M_X[i_0] = 0$, then $X_{i_0} = Y_{i_0} \in P$ is a block B that separates x from y. Therefore any shortest path from x to y has to cross x_{i_0} and y_{i_0}, both in B. In other words, $d_G(x,y) = d_G(x, x_{i_0}) + d_B(x_{i_0}, y_{i_0}) + d_G(y_{i_0}, y)$. Note $x_{i_0} \neq y_{i_0}$ so $d_B(x_{i_0}, y_{i_0})$ can be estimated thank to the labels $L_{\mathcal{B}}(x_{i_0}, B)$ and $L_{\mathcal{B}}(y_{i_0}, B)$. Since $D_x[i_0] = \lambda(d_G(x, x_{i_0}))$, $D_y[i_0] = \lambda(d_G(y, y_{i_0}))$ and since the local label of x_{i_0} and y_{i_0} are stored respectively in $P_X[i_0]$ and $P_Y[i_0]$, the approximation follows.

Let us upper bound the length of $L(x,G)$. Note that $h_X \leqslant 1 + \log|V(T)|$. By Lemma 3.1, $|V(T)| \leqslant 2|P| - 1 < 2p$, thus $h_X < \log(4p)$. The label length of x is therefore bounded by: $\lceil \log h_X \rceil = O(\log\log p)$ bits for h_X, plus $O(\log p)$ bits for S_X, plus $h_X - 1 = O(\log p)$ bits for M_X, plus $\alpha(h_X - 1) < \alpha\log(4p)$ bits for D_x, and plus at most $\beta \cdot h_X < \beta\log(4p)$ bits for P_X. By Lemma 3.1, $p < n$, thus h_X, S_X and M_X cost $O(\log n)$ bits. All together, we obtain $(\alpha+\beta)\log(4p) + O(\log n)$ bit for $L(x,G)$, and this completes the proof of Theorem 3.2. □

Theorem 3.2 can be completed by remarking that the scheme constructed for $\mathcal{F}$ is polynomial time constructible if the block-partitions, the estimator, and the DLS for $\mathcal{B}$ are polynomial time constructible.

4 Application to Some Graph Families

Let us now apply the technique presented in the previous section to some graph families. All the graphs we consider in this section are unweighted.

4.1 Block-Graphs

Theorem 3.2 can be applied in a trivial case: the *block-graph* family. A graph is a *block-graph* if all its maximal 2-connected components are cliques (see [18, 23]). Block-graphs can be seen as graphs constructed from trees by replacing each edge by a clique of arbitrary size. Every block-graph is *chordal* i.e., a graph without induced cycles of length larger than 3. Block-graphs are also the intersection graphs induced by the block-partition of a graph: the vertex set is the set of blocks, and the edge set is the blocks that intersect.

Theorem 4.1 *The family of unweighted n-vertex block-graphs has an exact DLS (resp. $(1+1/\log n)$-multiplicative) with $O(\log^2 n)$ (resp. $O(\log n \log\log n)$)*

bit labels. Moreover the scheme is polynomial time constructible and the distance decoder has constant time complexity.

Proof: We combine Lemma 3.1 ($p < n$), Lemma 3.2 (choosing $\alpha = O(\log n)$ or $O(\log\log n)$ depending on the willing approximation), and Theorem 3.2 remarking that $\beta = 0$ since exact DLS on the family of cliques can be done without any label. $\square$

Since block-graphs contain trees, Theorem 4.1 is optimal according to the following lower bound due to [13, 15].

Lemma 4.1 ([13, 15]) *Let s such that $1 \leqslant s \leqslant 1 + 6/\log n$, and let $r \geqslant 0$. Let $\mathcal{L}_s$ (resp. $\mathcal{L}_r$) be any s-multiplicative (resp. r-additive) DLS on the family of unweighted n-vertex trees. Then, $\mathcal{L}_s$ (resp. $\mathcal{L}_r$) assigns on a vertex of a tree a label of length $\Omega(\log n \log\log n)$ bits (resp. $\Omega(\log^2(n/(r+1)))$ bits).*

4.2 Generalized Distance-Hereditary Graphs

A graph G is *distance-hereditary* if the distance between any two vertices of any connected induced subgraph H of G is the same in H and in G. Distance-hereditary graphs have been investigated to design interconnection network topologies for their distance property in the case of faulty nodes (see [3] for references about this family).

We introduce the family of (s,r)-*distance-hereditary graphs*, with $s \geqslant 1$ and $r \geqslant 0$, a natural generalization of distance-hereditary graphs.

Definition 4.2 *A graph G is a (s,r)-*distance-hereditary graph *if for every connected induced subgraph H of G, $d_H(x,y) \leqslant s \cdot d_G(x,y) + r$ for any two vertices x,y of H. We denote by $\mathrm{DH}(s,r)$ the family of all (s,r)-distance-hereditary graphs.*

The sub-family $\mathrm{DH}(1,0)$ is exactly the family of distance-hereditary graphs, and $\mathrm{DH}(s,0)$ has been introduced and investigated in [6, 7]. The sub-family $\mathrm{DH}(1,r)$ have been in studied in [5]. Observe that the family $\mathrm{DH}(s,r)$ is close under taking induced subgraphs.

Recall that a graph is k-*chordal* if it does not contain any induced chordless cycle of length larger than k. Chordal graphs, are exactly 3-chordal graphs.

Lemma 4.2 *Every (s,r)-distance-hereditary graph is $(2s+r+2)$-chordal.*

Proof: Let C be a cycle of length ℓ. Observe that if $\ell > 2s + r + 2$, then $C \notin \mathrm{DH}(s,r)$. Indeed, the deletion of a vertex of C induces a subgraph P of C (a path) with two vertices x,y such that $d_P(x,y) = \ell-2 > 2s+r = s \cdot d_C(x,y)+r$, contradicting $C \in \mathrm{DH}(s,r)$. Since $\mathrm{DH}(s,r)$ is close under taking induced subgraph, every $G \in \mathrm{DH}(s,r)$ has no induced cycle of length $\ell > 2s+r+2$, implying that G is $(2s+r+2)$-chordal. $\square$

For k-chordal graphs we have the following result:

Lemma 4.3 ([13]) *The family of n-vertex k-chordal graphs has a $\lfloor k/2 \rfloor$-additive (resp. $(1+1/\log n, \lfloor k/2 \rfloor)$-approximate) DLS with $O(\log^2 n)$ (resp. $O(\log n \log\log n)$) bit labels. Moreover the scheme is polynomial time constructible and the distance decoder has constant time complexity.*

Lemma 4.2 and Lemma 4.3 give us an immediate DLS for the family $\mathrm{DH}(s,r)$.

Theorem 4.3 *The family of n-vertex (s,r)-distance-hereditary graphs has an $(s+\lfloor r/2 \rfloor+1)$-additive (resp. $(1+1/\log n, s+\lfloor r/2 \rfloor+1)$-approximate) DLS with $O(\log^2 n)$ (resp. $O(\log n \log\log n)$) bit labels. Moreover the scheme is polynomial time constructible and the distance decoder has constant time complexity.*

The scheme of [13] for k-chordal graphs is optimal in term of label length for every $k \leqslant n^\epsilon$, for arbitrary constant $\epsilon < 1$. Indeed, trees are k-chordal, and there is no r-additive DLS on trees with labels shorter than $O(\log^2(n/(r+1)))$ bits (cf. Lemma 4.1). However, for the family $\mathrm{DH}(s,r)$ there is a little hope to do better because Lemma 4.2 is not a complete characterization[3].

It is not difficult to check that, for all $s \geqslant 1$ and $r \geqslant 0$, the graph $C(s,r)$ composed of two cycles sharing an edge of length respectively $\ell_1 = s + \lfloor r/2 \rfloor + 3$ and $\ell_2 = s + \lceil r/2 \rceil + 2$ satisfies: $C(s,r)$ is $(s+\lfloor r/2 \rfloor+3)$-chordal (thus is also $(2s+r+2)$-chordal), but $C(s,r) \notin \mathrm{DH}(s,r)$. For that it suffices to delete a vertex of degree 3 in $C(s,r)$ to get a path P with two vertices x,y at distance two in $C(s,r)$ and such that: $d_P(x,y) = \ell_1 - 2 + \ell_2 - 2 = 2s+r+1 > s \cdot d_{C(s,r)}(x,y) + r$.

Let us investigate the sub-family $\mathrm{DH}(1,0)$ in more details, and let us try to improve Theorem 4.3 for $s=1$ and $r=0$. By Theorem 4.3, distance-hereditary graphs have a 2-additive DLS with $O(\log^2 n)$ bit labels. It is known [13] that 3-chordal graphs (and 4-chordal graphs as well) need $\Theta(n)$ bit labels for any exact DLS. However, the counter-example $C(1,0)$ taken from the above discussion – it is named *domino* in [17] – shows that many 4-chordal graphs are not distance-hereditary. So, there is hope to do better, at least from the approximation side. Effectively, we will see that an exact DLS can be designed with $O(\log^2 n)$ bit labels. We use the following result:

Lemma 4.4 ([12, 17]) *Distance-hereditary graphs has a split decomposition such that its split-components are stars and cliques [17]. Moreover, this split decomposition can be performed in linear time [12].*

Consider an n-vertex distance-hereditary graph G. Lemma 4.4 implies that in linear time one can construct a tree-like graph T from G (as defined in section 2) whose components are stars and cliques. Therefore, T is a block-graph. The associated tree of T, namely $\widetilde{T}$, is also a block-graph (it is a tree). By Lemma 2.2, T and $\widetilde{T}$ have $O(n)$ vertices. So, by Theorem 4.1, T and $\widetilde{T}$ have an efficient DLS. By Lemma 2.1, one can rebuilt the distances in G from the distances in T and $\widetilde{T}$, therefore:

[3]To our best knowledge, only the families $\mathrm{DH}(1,0)$ [17], $\mathrm{DH}(1,1)$ [5], and $\mathrm{DH}(3/2,0)$ [6] have been characterized in terms of forbidden induced subgraphs and have a polynomial time recognition algorithm. The recognition problem for $\mathrm{DH}(1,r)$ (r not fixed) has been shown Co-NP-complete [5].

Theorem 4.4 *The family of n-vertex distance-hereditary graphs enjoys an exact (resp. $(1+1/\log n)$-multiplicative) DLS (polynomially constructible) using labels of length $O(\log^2 n)$ (resp. $O(\log n \log\log n)$) and constant time decoder.*

Again since distance-hereditary graphs contain trees, this result is optimal by Lemma 4.1.

4.3 Weak-Bipolarizable Graphs

Since we are looking for graphs whose split components have a nice local (s, r)-approximate DLS, one could think about graphs whose split-components are k-chordal. But it is easy to see that for any $k > 4$, if G is k-chordal then its split-components are also k-chordal. So there is some hope only for 4-chordal graphs.

Let us consider the family of *weak-bipolarizable* or HHDA-free graphs [25]. This is the family consists of graphs without House (a square and a triangle sharing an edge), Hole (a cycle of length larger than 4), Domino (two squares sharing an edge), and A (a cycle of 4 vertices with two pendant vertices attached to two consecutive vertices on the cycle) as induced subgraphs. Even though HHDA-free graphs are not necessarily 3-chordal, [25] has showed that this family has a split decomposition (linear time constructible) such that the split-components are 3-chordal graphs.

Combining Theorem 3.2 and Lemma 4.3, we have that HHDA-free graphs enjoy 1-additive DLS with $O(\log^3 n)$ bit labels. However, one can reduce the label length to $O(\log^2 n)$ bits, observing that the tree-like graph obtained by the split decomposition of [25] is a 3-chordal graphs. So the scheme for a weak-bipolarizable graph G can be obtained from a simple application of Lemma 4.3 and Lemma 2.1.

Theorem 4.5 *The family of weak-bipolarizable n-vertex graphs enjoys an 1-additive (resp. $(1+1/\log n, 1)$-approximate) DLS (polynomially constructible) using labels of length $O(\log^2 n)$ (resp. $O(\log n \log\log n)$) and constant time decoder.*

5 Conclusion

This paper presents a way to combine several approximate DLS by the use of the decomposition techniques (block-partition and split decomposition). The final results are generalization of known results to larger graphs families. The natural question is about the use of more general graph decomposition techniques. For example, one could think about a block-partition based on k-vertices separators. On the other hand the split decomposition, which uses cutset (set of edges that splits the graphs into several components) that are complete bipartite graphs, may be generalized if we consider cutsets that are not complete bipartite graphs. What kind of bipartite graphs can be interesting to define a generalized split decomposition? This problem is of wide interest in algorithmic graph theory.

References

[1] S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe, *Identifying nearest common ancestors in a distributed environment*, Tech. Rep. IT-C Series 2001-6, ISSN 1600-6100, The IT University of Copenhagen, Aug. 2001.

[2] S. Alstrup and T. Rauhe, *Improved labeling scheme for ancestor queries*, Tech. Rep. IT-C Series 2001-5, ISSN 1600-6100, The IT University of Copenhagen, Aug. 2001.

[3] A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes – A survey*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1999.

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

[5] S. Cicerone, G. D'Ermiliis, and G. Di Stefano, *$(k,+)$-distance-hereditary graphs*, in 27^{th} International Workshop, Graph - Theoretic Concepts in Computer Science (WG), vol. 2204 of Lecture Notes in Computer Science, Springer, June 2001, pp. 66–77.

[6] S. Cicerone and G. Di Stefano, *Networks with small stretch number*, in 26^{th} International Workshop, Graph - Theoretic Concepts in Computer Science (WG), vol. 1928 of Lecture Notes in Computer Science, Springer, June 2000, pp. 95–106.

[7] ——, *Graphs with bounded induced distance*, Discrete Applied Mathematics, 108 (2001), pp. 3–21.

[8] D. G. Corneil, M. Habib, J.-M. Lanlignel, B. Reed, and U. Rotics, *Polynomial time recongition algorithm of clique-width $\leqslant 3$ graphs*, in Latin American Symposium on Theoretical Informatics (LATIN), vol. 1776 of Lecture Notes in Computer Science, 2000, pp. 126–134.

[9] B. Courcelle, J. Engelfriet, and G. Rozenberg, *Handle-rewriting hypergraph grammars*, Journal of Computer and System Science, 46 (1993), pp. 218–270.

[10] B. Courcelle and R. Vanicat, *Query efficient implementation of graphs of bounded clique width*, Discrete Applied Mathematics, (2001). To appear.

[11] W. Cunninghan, *Decomposition of directed graphs*, SIAM Journal on Algebraic and Discrete Methods, 3 (1982), pp. 214–228.

[12] E. Dahlhaus, *Efficient parallel and linear time sequential split decomposition*, in 14^{th} Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS), vol. 880 of Lectures Notes in Computer Science, Springer, Dec. 1994, pp. 171–180.

[13] C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg, *Approximate distance labeling schemes*, in 9^{th} Annual European Symposium on Algorithms (ESA), vol. 2161 of Lecture Notes in Computer Science, Springer, Aug. 2001, pp. 476–488.

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

[15] C. Gavoille, D. Peleg, S. Pérennes, and R. Raz, *Distance labeling in graphs*, in 12^{th} Symposium on Discrete Algorithms (SODA), ACM-SIAM, Jan. 2001, pp. 210–219.

[16] R. L. Graham and H. O. Pollak, *On embedding graphs in squashed cubes*, Lecture Notes in Mathematics, 303 (1972), pp. 99–110.

[17] P. Hammer and F. Maffray, *Completely separable graphs*, Discrete Applied Mathematics, 27 (1990), pp. 85–99.

[18] E. Howorka, *On metric properties of certain clique graphs*, Journal of Combinatoric Theory B, 27 (1979), pp. 67–74.

[19] S. Kannan, M. Naor, and S. Rudich, *Implicit representation of graphs*, SIAM Journal on Discrete Mathematics, 5 (1992), pp. 596–603.

[20] H. Kaplan and T. Milo, *Parent and ancestor queries using a compact index*, in 20^{th} ACM Symposium on Principles of Database Systems (PODS), ACM-SIAM, May 2001.

[21] ——, *Short and simple labels for small distances and other functions*, in 7^{th} International Workshop on Algorithms and Data Structures (WADS), vol. 2125 of Lecture Notes in Computer Science, Springer, Aug. 2001, pp. 32–40.

[22] M. Katz, N. A. Katz, A. Korman, and D. Peleg, *Labeling schemes for flow and connectivity*, Tech. Rep. MCS01-01, The Weizmann Institute of Science, Aug. 2001.

[23] K. C. Kay and C. Chartrand, *A characterization of certain ptolemaic graphs*, Canadian Journal of Mathematics, 17 (1965), pp. 342–346.

[24] J.-M. Lanlignel, *Autour de la décomposition en coupes*, PhD thesis, Université Montpellier II – Sciences et Techniques du Languedoc, June 2001.

[25] S. Olariu, *Weak-bipolarizable graphs*, Discrete Mathematics, 74 (1989), pp. 159–171.

[26] D. Peleg, *Informative labeling schemes for graphs*, in 25^{th} International Symposium on Mathematical Foundations of Computer Science (MFCS), vol. 1893 of Lecture Notes in Computer Science, Springer, Aug. 2000, pp. 579–588.

[27] ——, *Proximity-preserving labeling schemes*, Journal of Graph Theory, 33 (2000), pp. 167–176.

[28] J. P. Spinrad, *Efficient Graph Representations*, 2000. In preparation.