Optimal Distance Labeling for Interval and Circular-arc Graphs

Cyril Gavoille[1] and Christophe Paul[2]

[1] LaBRI, Université Bordeaux I, gavoille@labri.fr
[2] LIRMM, CRNS, paul@lirmm.fr

Abstract. In this paper we design a distance labeling scheme with $\mathcal{O}(\log n)$ bit labels for interval graphs and circular-arc graphs with n vertices. The set of all the labels is constructible in $\mathcal{O}(n)$ time if the interval representation of the graph is given and sorted. As a byproduct we give a new and simpler $\mathcal{O}(n)$ space data-structure computable after $\mathcal{O}(n)$ preprocessing time, and supporting constant worst-case time distance queries for interval and circular-arc graphs. These optimal bounds improve the previous scheme of Katz, Katz, and Peleg (STACS '00) by a $\log n$ factor. To the best of our knowledge, the interval graph family is the first hereditary family having $2^{\Omega(n \log n)}$ unlabeled n-vertex graphs and supporting a $o(\log^2 n)$ bit distance labeling scheme.

keywords: data-structure, distance queries, labeling scheme, interval graphs, circular-arc graphs

1 Introduction

Network representation plays an extensive role in the areas of distributed computing and communication networks (including peer-to-peer protocols). The main objective is to store useful information about the network (adjacency, distances, connectivity, etc.) and make it conveniently accessible.

This paper deals with distance representation based on assigning vertex labels [25]. Formally, a *distance labeling scheme* for a graph family $\mathcal{F}$ is a pair $\langle L, f \rangle$ of functions such that $L(v, G)$ is a binary label associated to the vertex v in the graph G, and such that $f(L(x, G), L(y, G))$ returns the distance between the vertices x and y in the graph G, for all x, y of G and every $G \in \mathcal{F}$. The labeling scheme is said an $\ell(n)$-distance labeling if for every n-vertex graph $G \in \mathcal{F}$, the length of the labels are no more than $\ell(n)$ bits.

A labeling scheme using short labels is clearly a desirable property for a graph, especially in the framework of distributing computing where individual processor element of a network want to communicate with its neighbors but has not enough local memory resources to store all the underlying topology of the network. Schemes providing compact labels play an important role for localized distributed data-structures (see [14] for a survey).

1.1 Related works

In this framework, Peleg [24] introduced *informative labeling schemes* for graphs and thereby captured a whole set of already known results. Among them implicit representation or adjacency labeling [4,18] whose objective is only to decide adjacency between two vertex labels, compact routing [11,29] whose objective is to provide the first edge on a near-shortest path between the source and the destination, nearest common ancestor and other related functions for trees [2,3,1,19] (used for instance in XML file engine), flow and connectivity [20].

A first set of results about distance labeling can be found in [15]. It is shown, for instance, that general n-vertex graphs enjoy an optimal $\Theta(n)$-distance labeling, and trees an optimal $\Theta(\log^2 n)$-distance labeling. It is also proved a lower bound of $\Omega(\sqrt{n})$ on the label length for bounded degree graphs, and of $\Omega(n^{1/3})$ for planar graphs. There are still intriguing gap between upper and lower bounds for these two latter results. The upper bound for planar graphs is $\mathcal{O}(\sqrt{n}\log n)$, coming from a more general result about graphs having small separators. Related works concern distance labeling schemes in dynamic tree networks [22], and approximate distance labeling schemes [12,27,28].

Several efficient schemes have been designed for specific graph families: interval and permutation graphs [21], distance hereditary graphs [13], bounded tree-width graphs (or graphs with bounded vertex-separator), and more generally bounded clique-width graphs [9]. All support an $\mathcal{O}(\log^2 n)$-distance labeling scheme. Except for the two first families (interval and permutation graphs), these schemes are known to be optimal as theses families include trees.

1.2 Our results

An interval graph is a graph whose vertices are intervals of the real line and whose edges are defined by the intersecting intervals. The interval graph family is hereditary, i.e., a family closed under vertex deletion, and supports a straightforward adjacency labeling with $\mathcal{O}(\log n)$ bit labels. Finding the complexity of distance label length for interval graphs is a difficult problem. The best scheme up to date is due to Katz, Katz, and Peleg [21]. It is based on the particular shape of the separators (that are cliques) and uses $\mathcal{O}(\log^2 n)$ bit labels. The $\Omega(\log^2 n)$ lower bound of [15] does not apply because interval graphs do not contain trees. As shown in [18], information-theoretic lower bound coming from the number of n-vertex graph in the family play an important role for the label length. Kannan, Naor, and Rudich [18] conjecture that any hereditary family containing no more than $2^{k(n)n}$ graphs of n vertices enjoys a $\mathcal{O}(k(n))$-adjacency distance labeling, $\Omega(k(n))$ being clearly a lower bound. Interval graphs, like trees and several other families cited above (including bounded tree-width, bounded clique-width, bounded genus graphs, etc.), are hereditary and support $\mathcal{O}(\log n)$ bit adjacency labels. However, there are much more interval graphs than all the above mentioned graphs. Indeed, trees, bounded tree-width, bounded clique-width, and bounded genus graphs possess only $2^{\mathcal{O}(n)}$ unlabeled n-vertex graphs. As we will see later there are $2^{\Omega(n \log n)}$ unlabeled interval graphs. Moreover, up to now, no hereditary graph family is known to support an $o(\log^2 n)$-distance labeling scheme[3]. All these remarks seem to plead in Katz-Katz-Peleg's favor with their $\mathcal{O}(\log^2 n)$-distance labeling scheme [21].

Surprisingly, we show that the interval graph family enjoys a $5\log n$-distance labeling scheme[4], and that any $\ell(n)$-distance labeling on this family requires $\ell(n) \geqslant 3\log n - \mathcal{O}(\log\log n)$. The lower bound derives from an new estimate of the number of interval graphs. We also show that proper interval graphs, a sub-family of interval graphs, enjoy a $2\log n$-distance labeling, and we prove an optimal lower bound of $2\log n - \mathcal{O}(\log\log n)$.

Moreover, once the labels have been assigned, the distance computation from the labels takes a constant number of additions and comparisons on $O(\log n)$ bit integers. Even more interesting, the preprocessing time to set all the labels runs optimally in $\mathcal{O}(n)$ time, once the sorted list of intervals of the graph is done. Our scheme extends to circular-arc graphs, a natural generalization of interval graphs, where the same bounds apply.

At this step, it is worth to remark that any $\ell(n)$-distance labeling scheme on a family $\mathcal{F}$ converts trivially into a non-distributed data-structure for $\mathcal{F}$ of $\mathcal{O}(\ell(n)\,n/\log n)$ space supporting distance queries within the same time complexity, being assumed that a cell of space can store $\Omega(\log n)$ bits of data. Therefore, our result implies that interval graphs (and circular-arc graphs) have a $\mathcal{O}(n)$ space data-structure, constructible in $\mathcal{O}(n)$ time, supporting constant time distance queries.

This latter formulation implies the result of Chen et al. [6]. However, we highlight that both approaches differ in essence. Their technique consists in building a one-to-one mapping from the vertices of the input graph to the nodes of a rooted tree, say T. Then, distances are computed as follows. Let $l(v)$ be the level of v in T (i.e., the distance from the root), and let $A(i,v)$ be the i-th ancestor of v (i.e., the i-th node on the path from v to the root). The distance d between x and y is computed by: if $l(x) > l(y)+1$ then $d = l(x)-l(y)-1+d_1(z,x)$ where $z = A(l(x)-l(y)-1, x)$, and where $d_1(z,x)$ is the distance between two nodes whose levels differ by at most 1. The distance d_1 is $1, 2$ or 3 and is computed by a case analysis with the interval representation of the involved vertices. Answering query is mainly based on the efficient implementation of *level ancestor queries* on trees (to compute z) given by Berkman and Vishkin [5]. However, this clever scheme cannot be converted into a distributed data-structure as ours for the following reason. As the tree has to support level ancestor queries, it implies that any node, if represented with a local label, can extract any of its ancestors with its level. In particular, x and y can extract from their label their nearest common ancestor and its level, so x and y can compute their distance in T. By

---

[3] It is not difficult to construct a family of diameter two graphs whose adjacency can be decided with $\mathcal{O}(\log n)$ bit labels (some bipartite graphs for instance), so supporting an $\mathcal{O}(\log n)$ distance labeling scheme. However, "diameter two" is not a hereditary property.

[4] In this paper all the logarithms are in based two.

the lower bound of [15], this cannot be done in less than $\Omega(\log^2 n)$ bit labels. So, access to a global data-structure is inherent to the Chen et al. [6] approach.

1.3 Outline of the paper

Let us sketch our technique (due to space limitation proofs have been removed from this extended abstract). The key of our scheme is a reduction to proper interval graphs, namely the family of graphs having an interval representation with only proper intervals, i.e., with no strictly included intervals. We carefully add edges to the input graph (by extending the length of non-proper intervals) to obtain a proper interval graph. Distances in the original graph can be retrieved from the label of two vertices of the proper interval graph and from the original interval (see Section 3).

Therefore, the heart of our scheme is based on an efficient labeling of proper interval graphs, and it is presented in Section 2. First, an integer $\lambda(x)$ is associated to every vertex x with the property that the distance d between x and y is: $d = \lambda(y) - \lambda(x) + \delta(x,y)$, if $\lambda(y) \geqslant \lambda(x)$, and where $\delta(x,y) = 0$ or 1. Another key property of λ is that the binary relation δ has the structure of a 2-dimensional poset (thus adjacency is feasible within two linear extensions). Actually, we show how to assign in $\mathcal{O}(n)$ time a number $\pi(x)$ to every x such that $\delta(x,y) = 1$ iff $\pi(x) > \pi(y)$. It gives a $2\log n$-distance labeling for proper interval graphs (with constant time decoding), and this is optimal from the lower bound presented in Section 5.

For general interval graphs we construct a $5\log n$-distance labeling scheme (see Section 3). In Section 5, we also prove a lower bound of $3\log n - \mathcal{O}(\log\log n)$. A byproduct of our counting argument used for the lower bound, is a new asymptotic of $2^{2n\log n - o(n\log n)}$ on the number of labeled n-vertex interval graphs, solving an open problem of the 1980's [16].

In Section 4 we show how to reduce distance labeling scheme on circular-arc graphs to interval graphs by "unrolling" twice the input circular-arc graph. Distances can be recovered by doubling the label length.

2 A Scheme for Proper Interval Graphs

For all vertices x, y of a graph G, we denote by $\mathrm{dist}_G(x,y)$ distance between x and y in G.

Proper interval graphs form a sub-family of interval graphs. A layout of an interval graph is *proper* if there is no intervals strictly contained in another one, i.e., there are no intervals $[a,b]$ and $[c,d]$ with $a < c < d < b$. A proper interval graph is an interval graph having a proper layout.

There are several well known characterizations of this sub-family: G is a proper interval graph iff G is an interval graph without any induced $K_{1,3}$ [30]; G is a proper interval graph iff it has an interval representation using unit length interval [26]. The layout of an interval graph, and a proper layout of a proper interval graph can be computed in linear time [8,10,17].

From now on we consider that the input graph $G = (V, E)$, a connected, unweighted and n-vertex proper interval graph, is given by a proper layout $\mathcal{I}$. If the layout is not proper, we use the general scheme described in Section 3. For every vertex x, we denote by $\text{L}(x)$ and $\text{R}(x)$ respectively the left and the right boundary of the interval $\mathcal{I}(x)$. As done in [6], the intervals are assumed to be sorted according to the left boundary, breaking the tie with increasing right boundary. We will also assume that the $2n$ boundaries are pairwise distinct. If not, it is not difficult to see that in $\mathcal{O}(n)$ time one can scan all the boundaries from $\min_x \text{L}(x)$ to $\max_x \text{R}(x)$ and compute another layout of G that is still proper, and with sorted and distinct boundaries[5].

Let x_0 be the vertex with minimum right boundary. The *base-path* $[x_0, \ldots, x_k]$ is the sequence of vertices such that $\forall i > 0$, x_i is the neighbor of x_{i-1} whose $\text{R}(x_i)$ is maximum. The *layer partition* $V_1, \ldots, V_k$ is defined by:

$$V_i = \{v \mid \text{L}(v) < \text{R}(x_{i-1})\} \setminus \bigcup_{0 \leqslant j < i} V_j \quad \text{with } V_0 = \varnothing.$$

Given the sorted list of intervals, the base-path and the layer partition can be

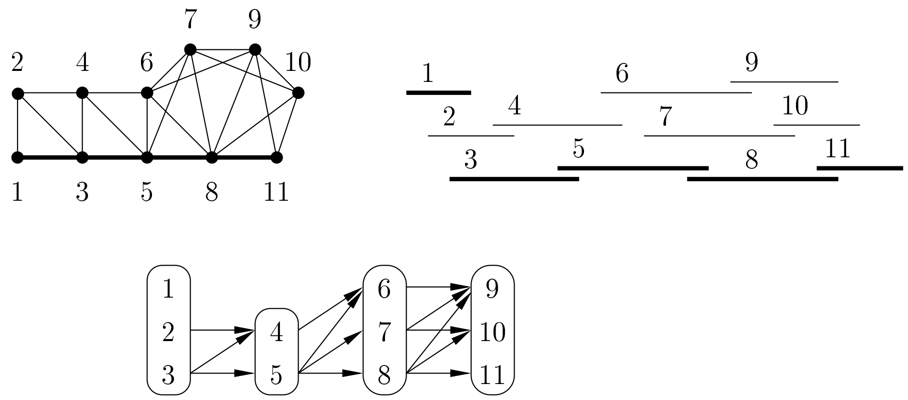

Fig. 1. A proper interval graph with an interval representation and the associated layer partition. The base-path is in bold.

computed in $\mathcal{O}(n)$ time. Let $\lambda(x)$ denote the unique index i such that $x \in V_i$. Let H be the digraph on the vertex set V composed of all the arcs xy such that $\lambda(x) < \lambda(y)$ and $(x, y) \in E$. Note that H is a directed acyclic graph. Let H^t denote the transitive closure of H (since H is acyclic, H^t defines a poset), and let $\text{adj}_{H^t}(x, y)$ be the boolean such that $\text{adj}_{H^t}(x, y) = 1$ iff xy is an arc of H^t.

The scheme we propose is based on the following theorem:

[5] Another technical assumption it that all the boundaries of the layout are positive integers bounded by some polynomial of n, so that boundaries can be manipulated in constant time on RAM-word computers. Note that recognizing linear time algorithms produce such layout.

Theorem 1. *For all distinct vertices x and y such that $\lambda(x) \leqslant \lambda(y)$,*

$$\mathrm{dist}_G(x,y) = \lambda(y) - \lambda(x) + 1 - \mathrm{adj}_{H^t}(x,y) \tag{1}$$

Therefore to compute the distance between any pair of vertices we have to test whether $\mathrm{adj}_{H^t}(x,y) = 1$. For this reason H^t can be considered as the *graph of errors* associated to the layer partition.

Theorem 2. *There exists a linear ordering π of the vertices, constructible in $\mathcal{O}(n)$ time, such that:*

$$\mathrm{adj}_{H^t}(x,y) = 1 \textit{ iff } \lambda(x) < \lambda(y) \textit{ and } \pi(y) < \pi(x) \tag{2}$$

Proof. The ordering π is defined by the following simple algorithm:

π is the pop ordering of a DFS on H using $\mathrm{L}(x)$ as a priority rule.

This algorithm can be seen as an elimination ordering of the vertices such that at each step, the sink of the subgraph of H induced by the non-eliminated vertices, that has a minimum left bound, is removed. On the example of Fig. 1, we obtain the following ordering:

vertex x	1	9	6	4	2	10	7	11	8	5	3
$\pi(x)$	1	2	3	4	5	6	7	8	9	10	11

This algorithm can be implemented to run in $\mathcal{O}(n)$ time. First the digraph H can be stored in $\mathcal{O}(n)$ space. The set of vertices are sorted in an array with respect to the left boundaries of their interval. Notice that the layers and also the neighborhood in H of any vertex appear consecutively in this array. So each vertex is associated to its first and last vertex in the array.

The priority rule implies that when a vertex v is popped by the algorithm, any vertex u of the same layer such that $\mathrm{L}(u) < \mathrm{L}(v)$ has already been popped. It means that when a vertex is at the top of the stack during the DFS, we can find in $\mathcal{O}(1)$ the next vertex to be pushed if the layer of each vertex is also stored and for each layer, the index of the last popped vertex is maintained.

Let us now prove the correctness of Eq. (2).

- $\Rightarrow$ It is easy to check that if there is an arc from x to y in H^t, then $\lambda(x) < \lambda(y)$ and $\pi(y) < \pi(x)$.
- $\Leftarrow$ So let x and y be two non-adjacent vertices in H^t such that $\lambda(x) < \lambda(y)$ (it implies that $\mathrm{L}(x) < \mathrm{L}(y)$).
  We first show that $\forall z$ such that $xz \in E(H^t)$, then $yz \in E(H^t)$. If $\lambda(x) = \lambda(y)$, yz exists since $\mathrm{L}(x) < \mathrm{L}(y)$. So assume that $\lambda(x) < \lambda(y)$. If xz exists but not yz, then we can enlighten a $K_{1,3}$ containing x, y and z with a vertex $v \in V_{\lambda(z)}$ that belongs to a shortest x,z-path in G: contradiction, G is a proper interval graph.
  By the way at the step y becomes a sink, there remains no vertex z such that $\lambda(y) < \lambda(z)$ and $xz \in E(H^t)$. If x is also a sink, then by the priority rule $\pi(x) < \pi(y)$. If it is not the case, then $\lambda(x) < \lambda(y)$ and there exists a

sink x' such that $xx' \in E(H^t)$ and $\lambda(x') \leqslant \lambda(y)$. Notice that $\text{L}(x') < \text{L}(y)$. Otherwise we would have $\lambda(x') = \lambda(y)$ and the existence of a shortest x,x'-path of length $\lambda(x') - \lambda(x)$ (see Theorem 1), would implies the existence of a x,y-path of same length. Therefore xy would be an arc of H^t: contradiction. Since $\text{L}(x') < \text{L}(y)$, the priority rule implies that x' is popped before y. Recursively applying this argument, we can prove that x will be popped before y by the algorithm and so $\pi(y) < \pi(x)$. $\square$

The *dimension* of a poset P is the minimum number d of linear orderings $\rho_1, \ldots, \rho_d$ such that $x <_P y$ iff $x <_{\rho_i} y$ for every i. The next corollary follows immediately from Theorem 2.

Corollary 1. *The graph of errors H^t is a poset of dimension two.*

To each vertex x of G, we assign the label $L(x, G) = \langle \lambda(x), \pi(x) \rangle$. The distance decoder is given by Eq. (1) and (2). Clearly, $\lambda(x), \pi(x) \in \{1, \ldots, n\}$, so:

Theorem 3. *The family of n-vertex proper interval graphs enjoys a distance labeling scheme using labels of length $2\lceil \log n \rceil$, and the distance decoder has constant time complexity. Moreover, given the sorted list of intervals, all the labels can be computed in $\mathcal{O}(n)$ time.*

3 A Scheme for Interval Graphs

In this section, we assume that $G = (V, E)$ is an interval graph, connected and unweighted, and that a layout $\mathcal{I}$ of G is given by a sorted list of intervals. Up to an $\mathcal{O}(n)$ preprocessing time, we assume that interval boundaries are pairwise distinct and taken from the set $\{1, \ldots, 2n\}$.

From $\mathcal{I}$ we will build a family of intervals $\mathcal{I}'$ that is the layout of a proper interval graph $G' = (V, E')$ so that the distance in G can be retrieved from the distances in G' plus some additional information. From now $\text{L}(x)$ and $\text{R}(x)$ will denote the boundaries of $\mathcal{I}(x)$, and $\text{L}'(x)$ and $\text{R}'(x)$ the boundaries of $\mathcal{I}'(x)$.

Let $x \in V$ such that $\mathcal{I}(x) \subset \mathcal{I}(y)$ for some vertex y, then the *enclosed neighbor* of x, denoted by $\text{N}_\text{e}(x)$, is the neighbor of x such that $\mathcal{I}(x) \subset \mathcal{I}(\text{N}_\text{e}(x))$ and such that $\text{R}(\text{N}_\text{e}(x))$ is maximum. As the boundaries of $\mathcal{I}$ are pairwise disjoint, the enclosed neighbor, if it exists, is unique for every vertex. The layout $\mathcal{I}'$ is obtained from $\mathcal{I}$ as follows: if x has an enclosed neighbor then we set $\mathcal{I}'(x) = [\text{L}(x), \text{R}(\text{N}_\text{e}(x))]$, and we set $\mathcal{I}'(x) = \mathcal{I}(x)$ otherwise. Let us remark:

1. $\text{L}'(x) = \text{L}(x)$ and $\text{R}'(x) \geqslant \text{R}(x)$;
2. by maximality of its right boundary, $\mathcal{I}(\text{N}_\text{e}(x)) = \mathcal{I}'(\text{N}_\text{e}(x))$; and
3. G is a subgraph of G', and thus $E \subseteq E'$.

Lemma 1. *The layout $\mathcal{I}'$ of G' is proper.*

Note that in the above transformation, the proper layout obtained may have $\mathcal{I}'(x) \subset \mathcal{I}'(y)$ and $\text{R}'(x) = \text{R}'(y)$. However $\mathcal{I}'$ is a proper layout, so the scheme of Section 2 applies. We say that a vertex x has been *extended* in G' if $\text{R}'(x) > \text{R}(x)$.

Lemma 2. *If x and y are two vertices such that* $\mathrm{R}(x) = \mathrm{R}'(x) \leqslant \mathrm{R}(y)$*. Then* $\mathrm{dist}_G(x, y) = \mathrm{dist}_{G'}(x, y)$.

For any vertex x, define the *maximum neighbor* of x, denoted by $\mathrm{N_m}(x)$, as the neighbor of x such that $\mathrm{R}(\mathrm{N_m}(x))$ is maximum. Like enclosed neighbors, maximum neighbors are not extended in G'. Otherwise we would have $\mathcal{I}(\mathrm{N_m}(x)) \subset \mathcal{I}(v)$ for some vertex v and $\mathrm{R}(\mathrm{N_m}(x)) < \mathrm{R}(v)$. Since $\mathcal{I}(x) \cap \mathcal{I}(\mathrm{N_m}(x)) \neq \varnothing$, we also have $\mathcal{I}(x) \cap \mathcal{I}(v) \neq \varnothing$. Since $\mathrm{R}(\mathrm{N_m}(x)) < \mathrm{R}(v)$, by definition, we should have $v = \mathrm{N_m}(x)$: contradiction.

Theorem 4. *Let x and y be two vertices such that* $\mathrm{R}(x) \leqslant \mathrm{R}(y)$*. Then,*

$$\mathrm{dist}_G(x, y) = \mathrm{dist}_{G'}(\mathrm{N_m}(x), y) + 1 - \mathrm{adj}_G(x, y) \ . \tag{3}$$

Eq. (3) directly gives a distance labeling scheme for the family of interval graphs. To each vertex, we have to associate $L(x, G)$:

- $\mathrm{R}(x)$ and $\mathrm{L}(x)$ the boundaries of $\mathcal{I}(x)$;
- $L(x, G')$ and $L(\mathrm{N_m}(x), G')$, the labels of x and $\mathrm{N_m}(x)$ in the proper interval graph G'.

The distance decoder is the function described in Eq. (3). It uses the distance decoder of proper interval graph given in the previous section. These labels, composed a priori of 6 integers, can be compacted. Indeed, there is no possible edge in G between any pair of vertices that do not belong to the same or consecutive layers of the layer partition. Thus, we have either $\lambda(\mathrm{N_m}(x)) = \lambda(x)$ or $\lambda(\mathrm{N_m}(x)) = \lambda(x) + 1$. We do not need to store the pair $\langle \lambda(x), \lambda(\mathrm{N_m}(x)) \rangle$. A single bit, say $b(x)$, suffices to recover the second part of this information. Therefore the label of a vertex x is:

$$L(x, G) = \langle \mathrm{L}(x), \mathrm{R}(x), \lambda(x), \pi(x), b(x), \pi(\mathrm{N_m}(x)) \rangle$$

As $1 \leqslant \mathrm{L}(x) < \mathrm{R}(x) \leqslant 2n$, $2 \lceil \log n \rceil + 2$ bits suffice two store the two first integers. The four next fields can be stored with $3 \lceil \log n \rceil + 1$ bits, summing up to $5 \lceil \log n \rceil + 3$. Since all the labels in proper interval graphs can be computed in $\mathcal{O}(n)$, and since G' can be obtained within the same complexity, all the labels of G can be computed in $\mathcal{O}(n)$ time.

Theorem 5. *The family of n-vertex interval graphs enjoys a distance labeling scheme using labels of length $5 \lceil \log n \rceil + 3$, and the distance decoder has constant time complexity. Moreover, given the sorted list of intervals, all the labels can be computed in $\mathcal{O}(n)$ time.*

4 A Scheme for Circular-Arc Graphs

Circular-arc graphs are a natural generalization of interval graphs, in which vertices correspond to arcs of a circle rather than intervals on the real line.

Circular-arc graphs can be recognized in $\mathcal{O}(n^2)$ time [23]. Consider an circular-arc graph G. We can associate to every vertex x of G a range $I(x) = [\theta_r(x), \theta_l(x)]$ of angles where $\theta_r(x)$ and $\theta_l(x)$ are taken clockwise in $[0, 2\pi)$. The range $I(x)$ can be seen as a "cyclic" interval in which $[\theta_r(x), \theta_l(x)]$ for $\theta_r(x) > \theta_l(x)$ stands for $[\theta_r(x), 2\pi) \cup [0, \theta_l(x)]$. The vertices x and y are adjacent if $I(x) \cap I(y) \neq \varnothing$.

Let $x_1, \ldots, x_n$ the vertices of G ordered such that $\theta_r(x_i) \leqslant \theta_r(x_{i+1})$ for every $i \in \{1, \ldots, n-1\}$. We associate to G a new intersection graph $\widetilde{G}$ with vertex set $V(\widetilde{G}) = \bigcup_{1 \leqslant i \leqslant n} \left\{I(x_i^1), I(x_i^2)\right\}$ where, for every $1 \leqslant i \leqslant n$:

$$I(x_i^1) = \begin{cases} [\theta_r(x_i), \theta_l(x_i)] & \text{if } \theta_r(x_i) \leqslant \theta_l(x_i) \\ [\theta_r(x_i), \theta_l(x_i) + 2\pi] & \text{if } \theta_r(x_i) > \theta_l(x_i) \end{cases}$$

$$I(x_i^2) = \begin{cases} [\theta_r(x_i) + 2\pi, \theta_l(x_i) + 2\pi] & \text{if } \theta_r(x_i) \leqslant \theta_l(x_i) \\ [\theta_r(x_i) + 2\pi, \theta_l(x_i) + 4\pi] & \text{if } \theta_r(x_i) > \theta_l(x_i) \end{cases}$$

Intuitively, $\widetilde{G}$ is obtained by unrolling twice the graph G. To form $\widetilde{G}$, we list all the intervals of G according to increasing angle θ_r, starting from the arc of x_1, and clockwise turning two rounds. So, each vertex x_i in G appears twice in $\widetilde{G}$ as x_i^1 and x_i^2. We check that $\widetilde{G}$ is an interval graph.

Lemma 3. *For every $i < j$, $\operatorname{dist}_G(x_i, x_j) = \min\left\{\operatorname{dist}_{\widetilde{G}}(x_i^1, x_j^1), \operatorname{dist}_{\widetilde{G}}(x_j^1, x_i^2)\right\}$.*

Therefore, a distance labeling scheme for interval graphs can be transformed into a scheme for circular-arc graph family by doubling the number of vertices, and the label length.

Theorem 6. *The family of n-vertex circular-arc graphs enjoys a distance labeling scheme using labels of length $\mathcal{O}(\log n)$, and the distance decoder has constant time complexity. Moreover, given the sorted list of intervals, all the labels can be computed in $\mathcal{O}(n)$ time.*

5 Lower Bounds

For any graph family $\mathcal{F}$, let $\mathcal{F}_n$ denote the set of graphs of $\mathcal{F}$ having at most n vertices. Before proving the main results of this section, we need some preliminaries.

An α-graph, for integer $\alpha \geqslant 1$, is a graph G having a pair of vertices (l, r), possibly with $l = r$, such that l and r are of eccentricity at most α. Let $H = (G_0, G_1, \ldots, G_k)$ be a sequence of α-graphs, and let (l_i, r_i) denote the pair of vertices that defines the α-graph G_i, for $i \in \{0, \ldots, k\}$. For each non-null integer sequence $W = (w_1, \ldots, w_k)$, we denote by H^W the graph obtained by attaching a path of length w_i between the vertices r_{i-1} and l_i, for every $i \in \{1, \ldots, k\}$ (see Fig. 2).

A sub-family $\mathcal{H} \subset \mathcal{F}$ of graphs is α-*linkable* if $\mathcal{H}$ consists only of α-graphs and if $H^W \in \mathcal{F}$ for every graph sequence H of $\mathcal{H}$ and every non-null integer sequence W.

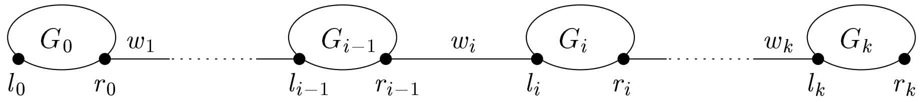

Fig. 2. Linking a sequence of α-graphs.

The following lemma shows a lower bound on the length of the labels used by a distance labeling scheme on any graph family having an α-linkable sub-family. The bound is related to the number of labeled graphs contained in the sub-family. As we will see later, the interval graph family supports a large 1-linkable sub-family (we mean large w.r.t. n), and the proper interval graph family supports large 2-linkable sub-family.

Lemma 4. *Let $\mathcal{F}$ be any graph family, and let $F(N)$ be the number of labeled N-vertex graphs of an α-linkable sub-family of $\mathcal{F}$. Then, every distance labeling scheme on $\mathcal{F}_n$ requires a label of length at least $\frac{1}{N}\log F(N) + \log N - 9$, where $N = \lfloor n/(\alpha \log n) \rfloor$.*

Let us sketch the proof of this lemma. We use a sequence H of $\Theta(\log n)$ α-graphs G_i taken from an arbitrary α-linkable sub-family, each with $N = \Theta(n/\log n)$ vertices, and spaced with paths of length $\Theta(n/\log n)$. Intuitively, the term $\frac{1}{N}\log F(N)$ measures the minimum label length required to decide whether the distance between any two vertices of a same G_i's is one or two. And the term $\log N$ denotes the minimum label length required to compute the distance between two vertices of consecutive G_i's. The difficulty is to show that some vertices require both information, observing that one can distribute information on the vertices in a non trivial way. For instance, the two extremity vertices of a path of length w_i does not require $\log w_i$ bit labels, but only $\frac{1}{2}\log w_i$ bits: each extremity can store one half of the binary word representing w_i, and merge their label for a distance query.

Let $I(n)$ be the number of labeled interval graphs with n vertices. At every labeled interval graph G with $n-1$ vertices, one can associated a new labeled graph G' obtained by attaching a vertex r, labeled n, to all the vertices of G. As the extra vertex is labeled n, all the associated labeled graphs are distinct, and thus their number is at least $I(n-1)$. The graph G' is an interval 1-graph, choosing $l = r$. Clearly, such interval 1-graphs can be linked to form an interval graph. It turns out that interval graphs have a 1-linkable sub-family with at least $I(n-1)$ graphs of n vertices.

To get a lower bound on the label length for distance labeling on interval graphs, we can apply Lemma 4 with $I(n)$. However, computing $I(n)$ is an unsolved graph-enumeration problem. Cohen, Komlós and Muller gave in [7] the probability $p(n,m)$ that a labeled n-vertex m-edge random graph is an interval graph under conditions on m. They have computed $p(n,m)$ for $m < 4$, and showed that $p(n,m) = \exp\left(-32c^6/3\right)$, where $\lim_{n\to+\infty} m/n^{5/6} = c$. As the total number of labeled n-vertex m-edge graphs is $\binom{\binom{n}{2}}{m}$, it follows a formula of

$p(n,m) \cdot \binom{\binom{n}{2}}{m}$ for the number of labeled interval graphs with $m = \Theta(n^{5/6})$ edges. Unfortunately, using this formula it turns out that $I(n) \geqslant 2^{\Omega(n^{5/6} \log n)} = 2^{o(n)}$, a too weak lower bound for our needs. The exact number of interval graphs is given up to 30 vertices in [16]. Actually, the generating functions for interval and proper interval graphs (labeled and unlabeled) are known [16], but only an asymptotic of $2^{2n+o(n)}$ for unlabeled proper interval graphs can be estimated from these equations. In conclusion Hanlon [16] left open to know whether the asymptotic on the number of unlabeled interval graphs is $2^{\mathcal{O}(n)}$ or $2^{\Omega(n \log n)}$.

As the number of labeled interval graphs is clearly at least $n! = 2^{(1-o(1))n \log n}$ (just consider a labeled path), the open question of Hanlon is to know whether $I(n) = 2^{(c-o(1))n \log n}$ for some constant $c > 1$. Hereafter we show that $c = 2$, which is optimal.

Theorem 7. *The number $I(n)$ of labeled n-vertex connected interval graphs satisfies $\frac{1}{n} \log I(n) \geqslant 2 \log n - \log \log n - \mathcal{O}(1)$. It follows that there are $2^{\Omega(n \log n)}$ unlabeled n-vertex interval graphs.*

We have seen that the interval graph family has a 1-linkable sub-family with at least $I(n-1)$ graphs of n vertices. By Theorem 7 and Lemma 4, we have:

Theorem 8. *Any distance labeling scheme on the family of n-vertex interval graphs requires a label of length at least $3 \log n - 4 \log \log n$.*

Using a construction of a 2-linkable sub-family of proper interval graphs with at least $(n-2)!$ graphs of n vertices, one can also show:

Theorem 9. *Any distance labeling scheme on the family of n-vertex proper interval graphs requires a label of length at least $2 \log n - 2 \log \log n - \mathcal{O}(1)$.*

References

1. S. Abiteboul, H. Kaplan, and T. Milo, *Compact labeling schemes for ancestor queries*, in 12^{th} Symp. on Discrete Algorithms (SODA), 2001, pp. 547–556.
2. S. Alstrup, P. Bille, and T. Rauhe, *Labeling schemes for small distances in trees*, in 15^{th} Symp. on Discrete Algorithms (SODA), 2003.
3. S. Alstrup, C. Gavoille, H. Kaplan, and T. Rauhe, *Nearest common ancestors: A survey and a new distributed algorithm*, in 14^{th} ACM Symp. on Parallel Algorithms and Architecture (SPAA), Aug. 2002, pp. 258–264.
4. S. Alstrup and T. Rauhe, *Small induced-universal graphs and compact implicit graph representations*, in 43^{rd} IEEE Symp. on Foundations of Computer Science (FOCS), 2002, pp. 53–62.
5. O. Berkman and U. Vishkin, *Finding level-ancestors in trees*, J. of Computer and System Sciences, 48 (1994), pp. 214–230.
6. D. Z. Chen, D. Lee, R. Sridhar, and C. N. Sekharan, *Solving the all-pair shortest path query problem on interval and circular-arc graphs*, Networks, 31 (1998), pp. 249–257.
7. J. E. Cohen, J. Komlós, and T. Mueller, *The probability of an interval graph, and why it matters*, Proc. of Symposia in Pure Mathematics, 34 (1979), pp. 97–115.

8. D. Corneil, H. Kim, S. Natarajan, S. Olariu, and A. Sprague, *Simple linear time algorithm of unit interval graphs*, Info. Proces. Letters, 55 (1995), pp. 99–104.
9. B. Courcelle and R. Vanicat, *Query efficient implementation of graphs of bounded clique width*, Discrete Applied Mathematics, (2001). To appear.
10. C. de Figueiredo Herrera, J. Meidanis, and C. Picinin de Mello, *A linear-time algorithm for proper interval recognition*, Information Processing Letters, 56 (1995), pp. 179–184.
11. C. Gavoille, *Routing in distributed networks: Overview and open problems*, ACM SIGACT News - Distributed Computing Column, 32 (2001), pp. 36–52.
12. C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg, *Approximate distance labeling schemes*, in 9^{th} European Symp. on Algorithms (ESA), vol. 2161 of LNCS, Springer, 2001, pp. 476–488.
13. C. Gavoille and C. Paul, *Distance labeling scheme and split decomposition*, Discrete Mathematics, (2003). To appear.
14. C. Gavoille and D. Peleg, *Compact and localized distributed data structures*, Research Report RR-1261-01, LaBRI, University of Bordeaux, Aug. 2001. To appear in J. of Distributed Computing for the PODC 20-Year Special Issue.
15. C. Gavoille, D. Peleg, S. Pérennes, and R. Raz, *Distance labeling in graphs*, in 12^{th} Symp. on Discrete Algorithms (SODA), 2001, pp. 210–219.
16. P. Hanlon, *Counting interval graphs*, Transactions of the American Mathematical Society, 272 (1982), pp. 383–426.
17. P. Hell, J. Bang-Jensen, and J. Huang, *Local tournaments and proper circular arc graphs*, in Algorithms, Int. Symp. SIGAL, vol. 450 of LNCS, 1990, pp. 101–108.
18. S. Kannan, M. Naor, and S. Rudich, *Implicit representation of graphs*, SIAM J. on Discrete Mathematics, 5 (1992), pp. 596–603.
19. H. Kaplan, T. Milo, and R. Shabo, *A comparison of labeling schemes for ancestor queries*, in 14^{th} Symp. on Discrete Algorithms (SODA), 2002.
20. M. Katz, N. A. Katz, A. Korman, and D. Peleg, *Labeling schemes for flow and connectivity*, in 13^{th} Symp. on Discrete Algorithms (SODA), 2002, pp. 927–936.
21. M. Katz, N. A. Katz, and D. Peleg, *Distance labeling schemes for well-separated graph classes*, in 17^{th} Symp. on Theoretical Aspects of Computer Science (STACS), vol. 1770 of LNCS, Springer Verlag, 2000, pp. 516–528.
22. A. Korman, D. Peleg, and Y. Rodeh, *Labeling schemes for dynamic tree networks*, in 19^{th} Symp. on Theoretical Aspects of Computer Science (STACS), vol. 2285 of LNCS, Springer, 2002, pp. 76–87.
23. R. M. McConnell, *Linear-time recognition of circular-arc graphs*, in 42^{th} IEEE Symp. on Foundations of Computer Science (FOCS), 2001.
24. D. Peleg, *Informative labeling schemes for graphs*, in 25^{th} Int. Symp. on Mathematical Foundations of Computer Science (MFCS), vol. 1893 of LNCS, Springer, 2000, pp. 579–588.
25. ———, *Proximity-preserving labeling schemes*, J. of Graph Theory, 33 (2000).
26. F. Roberts, *Indifference graphs*, in Proof Techniques in Graph Theory, Academic Press, 1969, pp. 139–146.
27. M. Thorup, *Compact oracles for reachability and approximate distances in planar digraphs*, in 42^{th} IEEE Symp. on Foundations of Computer Science (FOCS), 2001.
28. M. Thorup and U. Zwick, *Approximate distance oracles*, in 33^{rd} ACM Symp. on Theory of Computing (STOC), 2001, pp. 183–192.
29. ———, *Compact routing schemes*, in 13^{th} ACM Symp. on Parallel Algorithms and Architectures (SPAA), 2001, pp. 1–10.
30. G. Wegner, *Eigenschaften der Neven homologish-einfacher Familien im R^n*, PhD thesis, University of Göttingen, 1967.