Laboratoire Bordelais de Recherche en Informatique
UMR 5800 - Université Bordeaux 1, 351, cours de la Libération,
33405 Talence CEDEX, France

Research Report `RR-1343-05`

Localized and Compact Data-Structure for Comparability Graphs

by Fabrice Bazzaro and Cyril Gavoille

February, 2005

Localized and Compact Data-Structure for Comparability Graphs

Fabrice Bazzaro* Cyril Gavoille*

Abstract

We show that every comparability graph of any two-dimensional poset over n elements (a.k.a. permutation graph) can be preprocessed in $O(n)$ time, if two linear extensions of the poset are given, to produce an $O(n)$ space data-structure supporting distance queries in constant time. The data-structure is localized and given as a distance labeling, that is each vertex receives a label of $O(\log n)$ bits so that distance queries between any two vertices are answered by inspecting on their labels only. This result improves the previous scheme due to Katz, Katz and Peleg [KKP00] in the STACS '00 by a $\log n$ factor.

As a byproduct, our data-structure supports all-pair shortest-path queries in $O(d)$ time for distance-d pairs, and so identifies in constant time the first edge along a shortest path between any source and destination.

More fundamentally, we show that this optimal space and time data-structure cannot be extended for higher dimension posets. More precisely, we prove that for comparability graphs of three-dimensional posets, every distance labeling scheme requires $\Omega(n^{1/3})$ bit labels.

Keywords: algorithms, data-structure, distributed algorithms, distance queries, distance labeling scheme, partially ordered set, permutation graphs

1 Introduction

The dimension of a partially ordered set (or poset for short) $P = (V, <)$ is a fundamental invariant. It is the minimum number d of totally ordered sets $(V, <_1), \ldots, (V, <_d)$ whose intersection is P, i.e., $x < y$ if and only if $x <_i y$ for all $i \in \{1, \ldots, d\}$. Each total order $<_i$ is called linear extension of P, and a k-dimensional poset is a poset of dimension at most k.

*Laboratoire Bordelais de Recherche en Informatique, Université Bordeaux 1. {*bazzaro,gavoille*}@*labri.fr*. The two authors are supported by the project "PairAPair" of the ACI Masses de Données.

The comparability graph of a poset $P = (V, <)$ is the graph $G = (V, E)$ such that $\{x, y\} \in E$ if and only if $x < y$ or $y < x$. The important point is that all posets with the same comparability graph have the same dimension [BLS99, §7.6]. So the comparability graph is definitively a fundamental tool for the study of posets.

Of special interest are the two-dimensional posets because they can be characterized in term of a single ordering [DM41]. It is NP-complete to recognize posets of dimension three [Yan82] whereas linear time (linear in the size of the relation) algorithms exist for two-dimensional posets [MS97]. Actually, the comparability graphs of two-dimensional posets are exactly the permutation graphs, namely the intersection graphs of straight segments between two parallel lines [BFR71]. Intersection graphs are graphs in which vertices are mapped to objects, with the vertices defined to be adjacent if and only if the corresponding objects have nonempty intersection. See [MM99] for a comprehensive introduction to the intersection graphs.

This paper deals with the problem of the distance computation and distributed abilities of comparability of graphs. Commonly, when we make a query concerning a set of nodes in a graph (adjacency, distance, connectivity, etc.), we need to make a global access to the structure. In our approach, the compromise is to store the maximum of information in a label associated with a vertex to have directly what we need with a local access. Motivation of localized data-structures in distributed computing is survey and widely discussed in [GP03c].

We are especially interested in the distance labeling problem, introduced in [Pel00]. The problem consists in labeling the vertices of a graph to compute the distance between any two of its vertices x and y using only the information stored in the labels of x and y, without any other source of information. The main parameters taken into account when designing a solution are: (1) length (in bits) of the labels; (2) time complexity to decode the distance from the labels; and (3) time complexity to preprocess the graph and to compute all the labels.

1.1 Related works

Distance computation in graphs is one of the most fundamental graph algorithmic problem. Computing the distance matrix of a general graph is strongly related to Boolean matrix multiplication [DHZ00], and achieving this task as quickly as possible is a widely open problem.

However, the time complexity of this problem is known, and can be reduced significantly from the naive $O(n^3)$ upper bound, for many families of graphs: planar graphs [Dji96, KRRS94, Tho01], bounded tree-width graphs [CZ00], interval graphs [ACL95, CLSS98, GP03b], etc.

Beyond the classical all-pair distance problem, whose goal is to preprocess (possibly linearly) a graph and to produce a data-structure supporting distance or shortest-path queries in the minimum time complexity, the distance labeling problem is a variant in which the queries must be answered *locally*, by looking at the information related to the concerned vertices only. Introduced in [Pel00], it generalizes adjacency labeling [ABR03, KNR92] whose goal is only to decide whether the distance is 1 or not between any two vertices. At this point, it is worth to mention that any distance labeling scheme on a family $\mathcal{F}$ of graphs with $\ell(n)$ bit labels converts trivially into a non-distributed data-structure for $\mathcal{F}$ of $O(\ell(n) \cdot n/\log n)$ space supporting distance queries within the same time complexity, being assumed that a cell of space can store $\Omega(\log n)$ bits of data.

The main results on the field are that general graphs support a distance labeling scheme with labels of $O(n)$ bits [GPPR04], and that trees [ABR03, Pel00], bounded tree-width graphs [GPPR04], distance-hereditary graphs [GP03a], bounded clique-width graphs [CV03], some non-positively curved plane graphs [CDV04], interval and permutation graphs, all support distance labeling schemes with $O(\log^2 n)$ bit labels. There are also deep results concerning approximated distance labeling schemes we will not discuss here, e.g., see [GKK$^+$01, Tal04, Tho01, TZ01].

The $O(n)$ bit upper bound is tight for general graphs, and a lower bound of $\Omega(\log^2 n)$ bit on the label length is known for trees [GPPR04], implying that all the results mentioned above are tight as well, excepted for interval and permutation graphs that does not contain trees. Recently, [GP03b] showed an optimal bound of $O(\log n)$ bits for interval graphs and circular-arc graphs.

Concerning permutation graphs, we should mention the related work of [RLDL94]. It is shown how to compute one fixed BFS tree and DFS tree in $O(n)$ and $O(n \log\log n)$ time respectively when the permutation of the graph is given. Although original, the technique is limited; it does not allow us to choose arbitrarily the root of the trees.

1.2 Our results

In this paper, we are only interesting in comparability graphs, and in particular the permutation graphs, the comparability graphs of two-dimensional posets. This latter family is well-known and have a lot of valuable properties due to its characterizations as a partially ordered sets [DM41], as an intersection graph of segments between two parallel lines [Gol80] or as an Asteroidal Triple-free graph [COS97].

We show that permutation graphs with n vertices enjoy a distance labeling with labels of length $O(\log n)$ bits, more precisely $9\lceil \log n \rceil + 6$ bits. The distance can be computed in constant time, and all the labels are computed in $O(n)$ time if a realizer (the two linear extensions of the poset) is given. (Such realizer can be obtained be in $O(n + m)$ time [MS97] otherwise, m the number of edges). This result has optimal complexities (label length, distance decoder time complexity, and preprocessing time complexity). It improves within a $\log n$ factor on the label length the previous result of Katz, Katz and Peleg [KKP00] in the STACS '00.

We also show that $3 \log n$ bits for the label length of *any* labeling distance scheme is inevitably in the worst-case. As intermediate combinatorial result to prove this latter Information-Theoretic lower bound, and of independent interest, we prove that there are $2^{\Omega(n \log n)}$ unlabeled permutation graphs with n vertices.

As remarked previously, this leads to an optimal $O(n)$ space data-structure supporting distance queries in constant time, and computable in $O(n)$ time. We also show how to adapt our data-structure such that a length-d shortest path can be extracted in $O(d)$ time for any distance-d pair of vertices. One can also use our localized data-structure to identify the first shortest-path edge, and we therefore present a new shortest-path compact routing scheme for permutation graphs, improving the result of [DL02]. All these results can be extended to circular permutation graphs, a natural generalization of permutation graphs.

Looking for a generalized scheme with $O(f(d) \log n)$ bit labels for all comparability graphs of d-dimensional posets, we have proved that unfortunately no such function $f(d)$ can exist. More precisely, for every distance labeling scheme, there are comparability graphs of posets of dimension three that requires $\Omega(n^{1/3})$ bit labels. This makes a difference between comparability graphs of two- and three-dimensional posets for distance computation. This could not be observed when only adjacency is required, since $O(\log n)$ bit labels suffices for comparability graphs of bounded dimension posets.

1.3 Outline of the paper

Let us sketch our distance labeling scheme. We use a 2D geometric representation of the permutation graph, each vertex being associated with a point of $\mathbb{N}^2$, and u is adjacent to v if and only if v is in the South-West or North-East quadrant around u (cf. Fig. 2). For our purpose, two sets of points (or vertices) are of special interests: those with no neighbors in their North-East quadrant (call A), and those with no neighbors in their South-West quadrant (call B). Intuitively, one can always construct a shortest path between non-adjacent vertices by using only edges alternating between A and B. The main difficult resides in deciding whether the first edge must be incident to a vertex of A or of B.

The first trick is to treat differently short and long distances. We entirely characterize distances $\leqslant 3$ by comparing the coordinates of six specific vertices spread around each vertex (three belongs to A and three to B). Then, for long distances, i.e., distances $\geqslant 4$, we show that they reduce to the distance computation between vertices of two intermediate graphs, G_A and G_B, defined on respectively the vertices of A and of B. The edges of G_A and G_B are entirely determined by an asymmetric relation between the points of A and of B, and we show that a distance labeling for these graphs with $O(\log n)$ bit labels is possible. The distance is then computed by evaluating the distance in the graphs G_A and G_B between the six points associated with the source and the six points associated with the destination.

Finally, after several optimizations, the resulting data-structure is extremely simple. It is composed of 9 integers of $\{0, \ldots, n-1\}$ plus 6 bits. The distance decoder simply consists of a constant number of additions and comparisons on these integers.

Background and preliminaries are presented in Section 2, and the implementation of the scheme is presented in Section 3. An extension of the data-structure for all-pair shortest-path queries, compact routing, and to circular permutation graphs is discussed in Section 4. Several lower bounds are presented in Section 5 before a conclusion and a discussion in Section 6.

2 Preliminaries

We consider simple undirected graphs. Moreover, along this paper, we will assume that graphs are connected. We denote by $d_G(u,v)$ the distance between u and v in G, the minimum number of edges of a path connecting u to v. For a vertex u of graph G, we denote by $N(u)$ the set of neighbors of u, and $N[u] := N(u) \cup \{u\}$.

A *permutation graph* is an intersection graph of straight segments between two parallel lines [Gol80]. On Fig. 1, Segment 1 intersects the segments 2,5,6,7, so in the permutation graph vertex 1 is adjacent to the vertices 2,5,6,7. More formally, a permutation graph is isomorphic to a graph $G = (V,E)$ with $V = \{1,\ldots,n\}$ such that there exists a permutation π of V, called *realizer* of G, satisfying: u is adjacent to v if and only if $u < v$ and $\pi^{-1}(u) > \pi^{-1}(v)$.

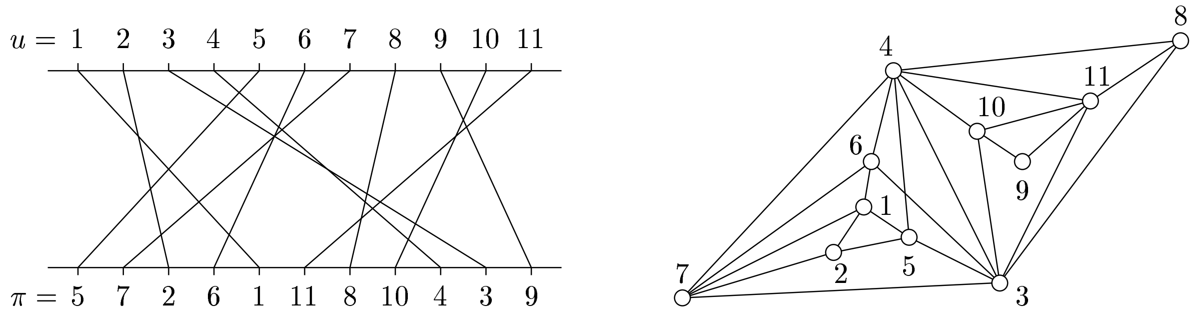

Figure 1: A permutation graph G with the realizer $\pi = (5,7,2,6,1,11,8,10,4,3,9)$.

It is not difficult to see that the isomorphism and the permutation π can be combined to form two linear extensions of a 2-dimensional poset. Actually, permutation graphs are exactly the comparability graphs of 2-dimensional posets [BFR71].

We can draw in the plane a permutation graph G with the realizer π, by associating with each vertex $u \in \{1,\ldots,n\}$ the point of coordinates $(u, \pi^{-1}(u))$. Within this graphic representation, the neighbors of u are the points located in the North-West and South-East quadrants around u (see Fig. 2 for an example).

Hereafter, we assume that $G = (V,E)$ is a given connected permutation graph, and π a realizer of G. Since $V = \{1,\ldots,n\}$, we use the total ordering on natural numbers as total ordering of the vertices of G. We partition the neighbors of u in the subsets

$$N^+(u) \ := \ \{v \in N(u) \mid v > u\} \quad \text{and} \quad N^-(u) \ := \ \{v \in N(u) \mid v < u\} \ .$$

(see left side of Fig. 2 for a graphical interpretation).

The following lemma is used in many places in the paper. The proof is obvious using the graphic representation.

Lemma 2.1 *For all $u, v, w \in V$ such that $u \leqslant v \leqslant w$, if $\{u, w\} \in E$, then $\{u, v\} \in E$ or $\{v, w\} \in E$, and if $\{u, w\} \notin E$, then $\{u, v\} \notin E$ or $\{v, w\} \notin E$.*

Proof. The statement is trivially true if $u = v$ or $v = w$. So let assume $u < v < w$.

Assume $\{u, w\} \in E$. Then $w \in N^+(u)$. If $\pi^{-1}(v) < \pi^{-1}(u)$, then $v \in N^+(u)$ so that $\{u, v\} \in E$. And, if $\pi^{-1}(v) > \pi^{-1}(u)$, then $w \in N^+(v)$ so that $\{v, w\} \in E$. Assume $\{u, w\} \notin E$. If $\pi^{-1}(v) > \pi^{-1}(u)$, then $\{u, v\} \notin E$. And, if $\pi^{-1}(v) < \pi^{-1}(u)$, then $\{v, w\} \notin E$. □

We distinguish two particular subsets of vertices, A and B depicted in black and gray on Fig. 2, defined by:

$$A \ := \ \left\{u \in V \mid N^-(u) = \varnothing\right\} \quad \text{and} \quad B \ := \ \left\{u \in V \mid N^+(u) = \varnothing\right\} \ .$$

Note that A and B are nonempty stables of G. If G is connected and has at least one edge, then $A \cap B = \varnothing$. Moreover, we check that G is bipartite if and only if $V = A \cup B$.

Lemma 2.2 *For every $u \in V$, there exist $a^-(u), a^+(u) \in A$ such that $a^-(u) \leqslant a^+(u)$ and $N[u] \cap A = [a^-(u), a^+(u)] \cap A$. Similarly, there exist $b^-(u), b^+(u) \in B$ such that $b^-(u) \leqslant b^+(u)$ and $N[u] \cap B = [b^-(u), b^+(u)] \cap B$.*

Proof. We only prove Lemma 2.2 for the first statement, about the set A. The second statement, about set B, can be proved similarly.

Let $x := \min N[u]$, which exists since $N[u]$ is nonempty. Let us show that $x \in A$, and thus that $N[u] \cap A \neq \varnothing$. If $x = u$, then $N^-(u) = \varnothing$, and thus $u \in A$ and $u = \min N[u]$. So, $x = u \in A$. So assume $x \neq u$. It follows that $x \in N^-(u)$. If $x \notin A$, then $N^-(x) \neq \varnothing$ and contains at least a vertex, say y. We have $y < x < u$, and $y \in N^-(x)$ and $x \in N^-(u)$. So, $y \in N[u]$: a contradiction with the definition of x.

Therefore $N[u] \cap A$ is nonempty, and in particular $N[u] \cap A \subseteq [x, y] \cap A$, where $x = \min \{N[u] \cap A\}$ and $y = \max \{N[u] \cap A\}$. Let us show that actually $N[u] \cap A = [x, y] \cap A$.

So let $w \in [x, y] \cap A$, and assume that $w \notin N[u] \cap A$. It follows that $w \in A$, and that $\{w, u\} \notin E$. Since A is a stable, w is adjacent to neither x nor y. If $w \leqslant u$, then we have $x \leqslant w \leqslant u$ and since $\{x, u\} \in E$ and $\{w, u\} \notin E$, by Lemma 2.1, w must be adjacent to

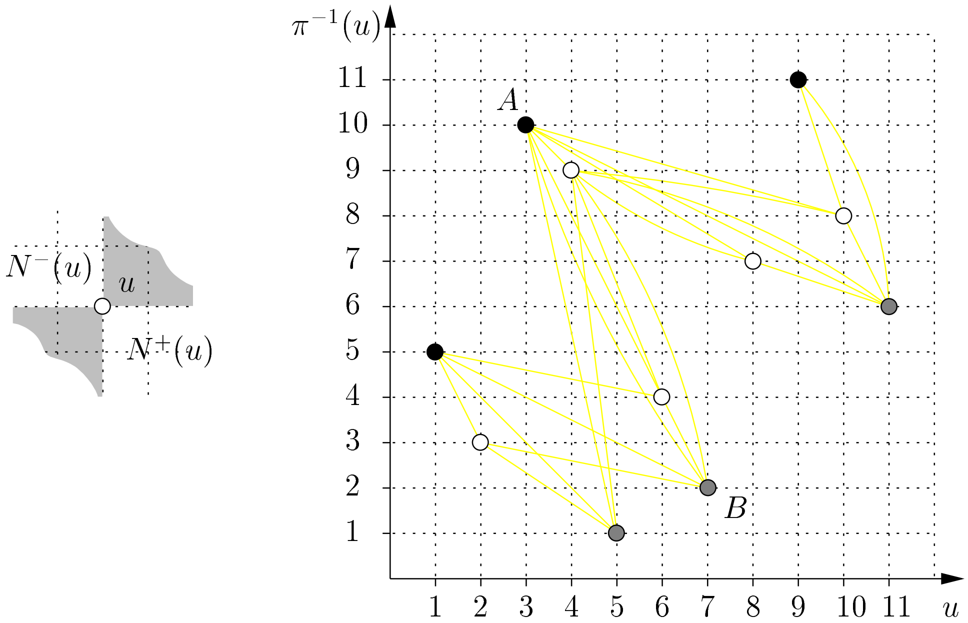

Figure 2: Graphic representation of the permutation graph of Fig. 1.

x: a contradiction. Similarly, if $w \geqslant u$, then $u \leqslant w \leqslant y$ yielding to w adjacent to y: a contradiction.

Therefore, such w does not exist and so $[x, y] \cap A \subseteq N[u] \cap A$ completing the proof. $\square$

According to Lemma 2.2, $N[u] \cap A$ and $N[u] \cap B$ are never empty and are consecutive in A and B respectively. Observe also that necessarily $a^-(u) = \min\{N[u] \cap A\}$ and $a^+(u) = \max\{N[u] \cap A\}$, and similarly for $b^-(u)$ and $b^+(u)$.

Lemma 2.3 *Let $u < v$ be two non-adjacent vertices. Then, $a^-(u) \leqslant a^-(v)$, $a^+(u) \leqslant a^+(v)$, $b^-(u) \leqslant b^-(v)$, and $b^+(u) \leqslant b^+(v)$.*

Proof. First let us show that $a^+(u) \leqslant u$. Indeed, if $a^+(u) > u$, then $\{a^+(u), u\} \in E$ ($a^+(u) \neq u$). Then, $a^+(u) \in N^+(u)$, or equivalently $u \in N^-(a^+(u))$: a contradiction with the fact that $N^-(a^+(u)) = \varnothing$. So, since $a^-(u) \leqslant a^+(u)$, we have also that $a^-(u) \leqslant u$.

Assume that $a^-(v) < a^-(u)$. Then we have $a^-(v) < a^-(u) \leqslant u < v$, i.e., $a^-(v) < u < v$. We have $\{a^-(u), v\} \in E$, and by assumption $\{u, v\} \notin E$, so by Lemma 2.1 u is adjacent to $a^-(v)$. It follows that there is a vertex $a^-(v) \in A$ smaller than $a^-(u)$ and adjacent to u: a contradiction with the definition of $a^-(u)$. Therefore, $a^-(v) \geqslant a^-(u)$.

Assume that $a^+(v) < a^+(u)$. Then, similarly, we have $a^+(v) < a^+(u) \leqslant u < v$, i.e., $a^+(v) < a^+(u) < v$. Again, by Lemma 2.1, since $\{a^+(v), v\} \in E$ and $\{a^+(v), a^+(u)\} \notin E$, v must be adjacent to $a^+(u)$. So there is a vertex $a^+(u) \in A$ greater than $a^+(v)$ and adjacent to v: a contradiction with the definition of $a^+(u)$. Therefore, $a^+(v) \geqslant a^+(u)$.

The proof that $b^-(u) \leqslant b^-(v)$ and $b^+(u) \leqslant b^+(v)$ is similar. $\square$

Lemma 2.4 *Let $u < v$ be two non-adjacent vertices. Then there exists a shortest path $u, w_1, \ldots, w_{k-1}, v$ such that $w_1 \in \{a^+(u), b^+(u)\}$, and, if $d_G(u,v) \geqslant 3$, $w_{k-1} \in \{a^-(v), b^-(v)\}$.*

Proof. Let $P = p_0, p_1, \ldots, p_{k-1}, p_k$ be a shortest path between $u = p_0$ and $v = p_k$ with $k = d_G(u,v) \geqslant 2$.

We consider the two first edges of P and show that p_1 can be replaced by a vertex satisfying the statement of the lemma. First let us show $p_0 < p_2$. If $p_2 < p_0$, then from the graphic representation we have:

1. v belongs to the North-East quadrant of $u = p_0$;
2. p_2 belongs to the South-West quadrant of u; and
3. there is no edge connecting the South-West to the North-East quadrant of u.

Therefore, because $\mathbb{R}^2$ is connected, every path from u to v through p_2 has to contain an intermediate vertex w located either in the North-East or South-West quadrant of u. A contradiction since w would be adjacent to u resulting a path shorter than P.

Observe that since $\{p_0, p_2\} \notin E$, either $p_1 \in N^-(p_0) \cap N^-(p_2)$ or $p_1 \in N^+(p_0) \cap N^+(p_2)$.

Assume that $p_1 \in N^-(p_0) \cap N^-(p_2)$. It suffices to show that $a^+(p_0)$ is adjacent to p_2 (and so w_1 can be chosen as $a^+(u)$). We have $a^+(p_1) \in N(p_0)$ and $a^+(p_1) \in N(p_2)$ as well. In other words, $a^+(p_1) \in [a^-(p_0), a^+(p_0)] \cap [a^-(p_2), a^+(p_2)]$ implying $a^-(p_2) \leqslant a^+(p_0)$. By Lemma 2.3 applied on the non-adjacent vertices $p_0 < p_2$, we obtain $a^+(p_0) \leqslant a^+(p_2)$, and thus $a^-(p_2) \leqslant a^+(p_0) \leqslant a^+(p_2)$. It follows that $a^+(p_0) \in [a^-(p_2), a^+(p_2)]$, and therefore $a^+(p_0)$ is adjacent to p_2.

The case $p_1 \in N^+(p_0) \cap N^+(p_2)$ is similar by proving that $b^+(p_0)$ is adjacent to p_2.

Finally, if u and v are at distance $\geqslant 3$, i.e., $k \geqslant 3$, then we can consider the two last edges of P along the vertices p_{k-2}, p_{k-1}, p_k and show that $p_{k-1} \neq p_1$ can be replaced by a vertex satisfying the statement of the lemma. The proof is symmetric exchanging the role of p_k with p_0, p_{k-1} with p_1, and p_{k-2} with p_2, and taken into account the fact that now $p_k > p_{k-2}$. $\square$

Hereafter, we denote by G_A the intersection graph of the family $\{[b^-(u), b^+(u)] \mid u \in A\}$. Similarly, we denote by G_B the intersection graph of the family $\{[a^-(u), a^+(u)] \mid u \in B\}$. By construction, G_A and G_B are interval graphs with vertex sets A and B respectively (see Fig. 3). As we will see later in Lemma 3.2, G_A and G_B are proper interval graphs.

The interesting connection between G and G_A, G_B is the following:

Lemma 2.5 *For all $u, v \in A$, $d_G(u,v) = 2d_{G_A}(u,v)$, and for all $u, v \in B$, $d_G(u,v) = 2d_{G_B}(u,v)$.*

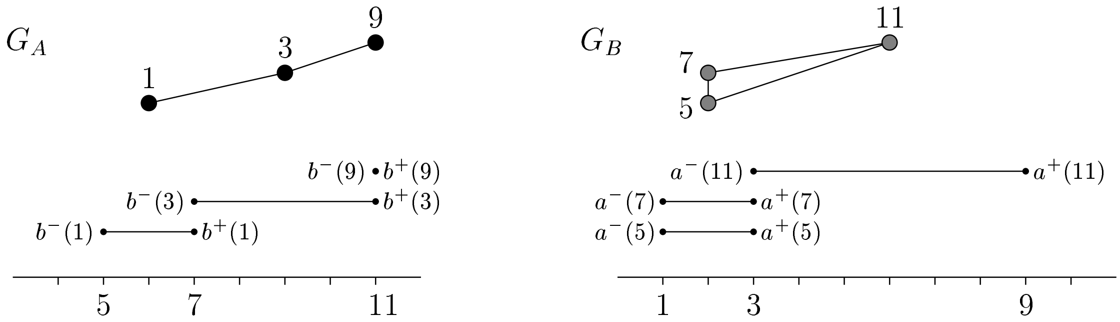

Figure 3: The proper interval graphs G_A and G_B from Fig. 2.

Proof. The lemma is true if $u = v$, so assume $u \neq v \in A$. We have $\{u, v\} \notin E$, and

$$\begin{aligned} d_{G_A}(u,v) = 1 \Leftrightarrow [b^-(u), b^+(u)] \cap [b^-(v), b^+(v)] \quad &\Leftrightarrow \quad [b^-(u), b^+(u)] \cap [b^-(v), b^+(v)] \cap B \neq \varnothing \\ N(u) \cap N(v) \neq \varnothing \quad &\Leftrightarrow \quad d_G(u,v) = 2 \ . \end{aligned}$$

So, if $d_{G_A}(u,v) = k$, then $d_G(u,v) = 2k$. Similarly for distinct $u, v \in B$. $\square$

E.g., in our example, $d_G(1,9) = 4$ because $1, 9 \in A$ and $d_{G_A}(1,9) = 2$ (see Fig. 3). By Lemma 2.5, since G is connected, G_A and G_B are connected too. We are now ready to prove the main result of this section that is the heart of our distance decoder.

Theorem 2.1 *Let u, v be two vertices with $u < v$. Then,*

1. *if $\pi^{-1}(u) > \pi^{-1}(v)$, then $d_G(u,v) = 1$;*
2. *otherwise, if $a^-(v) \leqslant a^+(u)$ or $b^-(v) \leqslant b^+(u)$, then $d_G(u,v) = 2$;*
3. *otherwise, if $a^-(v) \leqslant a^+(b^+(u))$ or $b^-(v) \leqslant b^+(a^+(u))$, then $d_G(u,v) = 3$;*
4. *otherwise, $d_G(u,v)$ is the minimum between the four distances:*
   - $2 + 2d_{G_B}(b^+(u), b^-(v))$
   - $3 + 2d_{G_A}(a^+(b^+(u)), a^-(v))$
   - $2 + 2d_{G_A}(a^+(u), a^-(v))$
   - $3 + 2d_{G_B}(b^+(a^+(u))), b^-(v))$

Proof. Let $u < v$ be two vertices. If $\pi^{-1}(u) > \pi^{-1}(v)$, then by definition, u and v are adjacent, completing the proof of Case 1.

Now, suppose that $\pi^{-1}(u) < \pi^{-1}(v)$. Then, by definition, $\{u, v\} \notin E$. Since $u < v$, by Lemma 2.3, $a^-(u) \leqslant a^-(v)$ and $b^-(u) \leqslant b^-(v)$. So in Case 2, if $a^-(v) \leqslant a^+(u)$, then $a^-(v) \in [a^-(u), a^+(u)] \cap A \subseteq N[u]$ (Lemma 2.2). Actually, $a^-(v) \in N(u)$, because if $a^-(v) = u$, then $d_G(u,v) \leqslant 1$ which is impossible. Therefore, $u, a^-(v), v$ is a path in G, and $d_G(u,v) = 2$. Similarly, if $b^-(v) \leqslant b^+(u)$, $u, b^-(v), v$ is a path in G. So $d_G(u,v) = 2$ in both cases completing the proof of Case 2.

Now, suppose that we are neither in the first nor in the second case. Let us first show that $d_G(u,v) > 2$. Indeed, by Lemma 2.4, if $d_G(u,v) = 2$, then there exists a shortest path u, w, v with some $w \in A \cup B$. However, since $a^+(u) < a^-(v)$, we have $[a^-(u), a^+(u)] \cap [a^-(v), a^+(v)] = \varnothing$ that implies $N[u] \cap N[v] \cap A = \varnothing$, and similarly we have $N[u] \cap N[v] \cap B = \varnothing$ (because $b^+(u) < b^-(v)$). Hence such w does not exist, and $d_G(u, v) > 2$.

We are now going to prove that if $a^-(v) \leqslant a^+(b^+(u))$, then $d_G(u, v) = 3$. We have $u \leqslant v$ and $\{u, b^+(u)\} \in E$. If $v \leqslant b^+(u)$ then by Lemma 2.1, v neighbors u or $b^+(u)$. This is not possible because $d_G(u, v) > 2$. So $b^+(u) \leqslant v$. Clearly, v and $b^+(u)$ are not neighbors since $d_G(u, v) > 2$. We are now under the hypothesis of Case 2 applied between $b^+(u)$ and v, and thus $d_G(b^+(u), v) = 2$. So there is a path of length three between u and v, and since $d_G(u, v) > 2$, we get that $d_G(u, v) = 3$. The case $b^-(v) \leqslant b^+(a^+(u))$ can be treated similarly. This completes Case 3.

We now assume that we are not in the three first cases. We have that v is not a neighbor of $b^+(u)$, and we have seen previously that $b^+(u) \leqslant v$. Previously, we have seen that when $u \leqslant v$, and u and v are not in the two first cases, $d_G(u, v) > 2$. Now we can apply the same argument between $b^+(u)$ and v to obtain $d_G(b^+(u), v) > 2$ and consequently $d_G(u, v) > 3$.

By Lemma 2.4, let $P = u, w_1, \ldots, w_{k-1}, v$ be a shortest path with $w_1 \in \{a^+(u), b^+(u)\}$ and $w_{k-1} \in \{a^-(v), b^-(v)\}$. If $w_1, w_{k-1} \in A$ or $w_1, w_{k-1} \in B$, then, by Lemma 2.5,

$$d_G(u, v) = 2 + 2d_{G_B}(b^+(u), b^-(v)) \quad \text{or} \tag{1}$$

$$d_G(u, v) = 2 + 2d_{G_A}(a^+(u), a^-(v)) \ . \tag{2}$$

So assume that $w_1 \in B$ and $w_{k-1} \in A$, i.e., $w_1 = b^+(u)$ and $w_{k-1} = a^-(v)$ (the proof is similar if $w_1 \in A$ and $w_{k-1} \in B$). Since $d_G(b^+(u), v) \geqslant 3$ we can apply Lemma 2.4 between w_1 and v, and let $P' = w'_1, \ldots, w'_{k-2}, w'_{k-1}, v$ be this shortest path from $w'_1 = w_1$ to v. Observe that necessary $w'_2 \in A$ (since $w'_1 \in B$) and thus $w'_2 = a^+(b^+(u))$.

If $w'_{k-1} \in B$, then $u, b^+(u), w'_2, \ldots, w'_{k-2}, w'_{k-1}, v$ is a shortest path, and by Lemma 2.5,

$$d_G(u, v) = 2 + 2d_{G_B}(b^+(u), b^-(v)) \ . \tag{3}$$

If $w'_{k-1} \in A$, then, since $w'_2 \in A$, by Lemma 2.5,

$$d_G(u, v) = 3 + 2d_{G_A}(a^+(b^+(u)), a^-(v)) \ . \tag{4}$$

Similarly, if $w_1 \in A$ and if w_{k-1} belongs to A or to B, then the distance is

$$d_G(u, v) = 2 + 2d_{G_A}(a^+(u), a^-(v)) \quad \text{or} \tag{5}$$

$$d_G(u, v) = 3 + 2d_{G_B}(b^+(a^+(u)), b^-(v)) \ . \tag{6}$$

Let d_i be the right-value provided by Eq. (i). For each equation, we check that, if $u < v$ and $d_G(u, v) \geqslant 4$, then there is a path from u to v using the vertices involved in the corresponding formula. E.g., there always exists a path from u to v using $b^+(u)$ and $b^-(v)$ if $d_G(u, v) \geqslant 4$. So, $d_G(u, v) = \min_i \{d_i\}$.

Finally, we observe that Eq. (1) and (3), and Eq. (2) and (5) are the same. Therefore,

$$d_G(u,v) \;=\; \min\{d_1, d_2, d_4, d_6\} \;.$$

□

3 Implementation of the Scheme

3.1 Distance labeling for proper interval graphs

An interval graph is the intersection graph of a family of intervals of the real line. The *layout* of an interval graph is the set of intervals associated with each vertex of the graph. A layout is *proper* if no intervals are strictly contained in another one, i.e., there are no intervals $[a,b]$ and $[c,d]$ with $a < c$ and $d < b$ (however $a = c$, or $b = d$ is possible). The graphs having such layouts are called proper interval graphs.

Motivated by Lemma 2.5, we will use as a sub-routine the distance labeling scheme of [GP03b] for proper interval graphs that we need to detail. To compute all the labels in $O(n)$ time, the scheme of [GP03b] takes as input a proper layout in a special form: a normalized proper layout. A layout is *normalized* if: (1) the left boundaries of the intervals are distinct; (2) the intervals are sorted according to their left boundaries (so there is a way to list them in $O(n)$ time by increasing left boundary); (3) the boundaries are integers bounded by some polynomials of n (so that boundaries can be manipulated in constant time on RAM-word computers). Note that the layouts provided by the linear time recognizing algorithms of [CKN+95, dFHMPdM95] have such properties. Observe also that a layout satisfying properties (2) and (3) can be easily transformed in $O(n)$ time in a normalized layout by scanning all the intervals by increasing left boundaries.

Consider any connected proper interval graph H with n vertices with a normalized proper layout $\mathcal{L}_H = \{[\mathrm{L}_H(u), \mathrm{R}_H(u)] \mid u \in V(H)\}$. In [GP03b], it has been proved that in $O(n)$ time it is possible to compute two mappings $\lambda_H, \sigma_H : V(H) \to \{0, \ldots, n-1\}$ such that the distance between any two vertices x, y can be computed as combinations of $\lambda_H(x), \sigma_H(x)$ and $\lambda_H(y), \sigma_H(y)$. In other words, the family of proper interval graphs supports a distance labeling scheme with $2\lceil \log n \rceil$ bit labels, and this is tight up to an additive $\log\log n$ term [GP03b]. The mappings λ_H and σ_H are respectively based on a BFS and a DFS initiated from the vertex x_0 of H whose left boundary is minimum in $\mathcal{L}_H$.

Let us define the binary relation $\mathrm{adj}_H(x,y) \in \{0,1\}$ by: $\mathrm{adj}_H(x,y) = 1$ if and only if $\lambda_H(x) < \lambda_H(y)$ and $\sigma_H(x) > \sigma_H(y)$. These mappings satisfy the following properties:

Lemma 3.1 ([GP03b]) *For all distinct vertices x and y of H with $\lambda_H(x) \leqslant \lambda_H(y)$:*

1. $d_H(x,y) = \lambda_H(y) - \lambda_H(x) + 1 - \mathrm{adj}_H(x,y)$.
2. $\lambda_H(x) = d_H(x_0, x)$.

3. σ_H is bijective.

4. If $\lambda_H(x) = \lambda_H(y)$, then $\mathrm{L}_H(x) < \mathrm{L}_H(y)$ if and only if $\sigma_H(x) < \sigma_H(y)$.

Remark. The function $\mathrm{adj}_H(x, y)$ involved in Lemma 3.1 can be seen as the matrix of a permutation graph (clearly, $\mathrm{adj}_H(x, y)$ defines a comparability graph of a two-dimensional poset whose linear extensions are λ_H and the reverse of σ_H). A surprising fact is that, in the scheme of [GP03b], distance computation in interval graphs is based on adjacency in permutation graphs. In this paper we use distances in proper interval graphs to design distance labeling for permutation graphs. Fortunately, this does not defined an infinite recursive loop!

Lemma 3.2 *The families $\mathcal{I}_A := \{[b^-(u), b^+(u)] \mid u \in A\}$ and $\mathcal{I}_B := \{[a^-(u), a^+(u)] \mid u \in B\}$ are proper layouts of the graphs G_A and G_B respectively.*

Proof. Let $u, v \in A$ with $u < v$. Lemma 2.3 applies to u and v, since u and v cannot be adjacent (A is a stable). It follows that $b^-(u) \leqslant b^-(v)$ and $b^+(u) \leqslant b^+(v)$. Hence $b^-(u) < b^-(v) < b^+(v) < b^+(u)$ is impossible, as well $b^-(v) < b^-(u) < b^+(u) < b^+(v)$. So $[b^-(u), b^+(u)]$ is not a proper sub-interval of $[b^-(v), b^+(v)]$, as well $[b^-(v), b^+(v)]$ is not a proper sub-interval of $[b^-(u), b^+(u)]$. Therefore, $\mathcal{I}_A$ is a proper layout of G_A. Similarly, $\mathcal{I}_B$ is a proper layout of G_B. $\square$

3.2 Distance decoder

At this step, we have enough material to prove that permutation graphs enjoy an $O(\log n)$ bit distance labeling scheme. Indeed, Theorem 2.1 can be easily implemented with short labels. Each vertex u could store: (1) the integers u and $\pi^{-1}(u)$ in order to compute distances $\leqslant 1$; (2) the x-coordinates about the six vertices $a^-(u), a^+(u), b^-(u), b^+(u), a^+(b^+(u)), b^+(a^+(u))$ for distances 2 and 3; and (3) the distance labels (two integers per label) in the proper interval graphs G_A and G_B about the same six vertices for distances $\geqslant 4$. The resulting label is composed of a total of $2 + 6 + 6 \times 2 = 20$ integers in $O(n)$. Thus such a label is of length $20 \log n + O(1)$ bits. However, we will show how to significantly reduce this length.

In order to apply the result of [GP03b] (Lemma 3.1) we need to transform the initial layouts $\mathcal{I}_A$ and $\mathcal{I}_B$ into the normalized layouts that we denote hereafter by $\mathcal{L}_A = \{[\mathrm{L}_A(u), \mathrm{R}_A(u)] \mid u \in A\}$ and $\mathcal{L}_B = \{[\mathrm{L}_B(u), \mathrm{R}_B(u)] \mid u \in B\}$ respectively. Clearly, all the boundaries of $\mathcal{I}_A$ and of $\mathcal{I}_B$ are in $O(n)$, so sorting them can be done in $O(n)$ time. Actually, we check that the left boundaries of $\mathcal{I}_A$ and $\mathcal{I}_B$ are already sorted if A and B are sorted. To produce normalized layouts, the left boundaries are ordered and distinguished with the following rule:

$$\forall\, u, v \in A, \quad \mathrm{L}_A(u) < \mathrm{L}_A(v) \ \text{ if and only if } \ b^-(u) < b^-(v), \text{ or } b^-(u) = b^-(v) \text{ and } u < v.$$

Similary, left boundaries of B are ordered such that $\text{L}_B(u) < \text{L}_B(v)$ if and only if $a^-(u) < a^-(v)$, or $a^-(u) = a^-(v)$ and $u < v$. This can be done in linear time by scanning the left boundaries of $\mathcal{I}_A$ (and of $\mathcal{I}_B$) by increasing order, and by shifting them to produce $\mathcal{L}_A$. We check that the resulting boundaries are in $O(n)$.

Thanks to Lemma 2.3 we can prove the following properties for the layouts $\mathcal{L}_A$ and $\mathcal{L}_B$:

Lemma 3.3 *For all $u, v \in A$, $u < v$ if and only if $\text{L}_A(u) < \text{L}_A(v)$. Similarly, for all $u, v \in B$, $u < v$ if and only if $\text{L}_B(u) < \text{L}_B(v)$.*

Proof. We prove the lemma for $u, v \in A$ only, the proof being symmetric for $u, v \in B$.

Assume $u < v$, and let us show that it implies $\text{L}_A(u) < \text{L}_A(v)$. By the normalization, it suffices to show that $b^-(u) \leqslant b^-(v)$. A is a stable, thus $u < v$ implies $b^-(u) \leqslant b^-(v)$ by Lemma 2.3.

Assume $\text{L}_A(u) < \text{L}_A(v)$, and let us show that it implies $u < v$. By the normalization, it suffices to show that $b^-(u) \leqslant b^-(v)$ implies $u < v$. If $b^-(u) = b^-(v)$, then we are done: $u < v$ by the priority rule in the normalization of $\mathcal{L}_A$. Assume $b^-(u) < b^-(v)$. Using the converse of Lemma 2.3 (again u and v are not adjacent), we obtain $u \leqslant v$, and since $u = v$ is impossible ($\text{L}_A(u) \neq \text{L}_A(v)$), hence $u < v$. □

Let λ_A, σ_A and λ_B, σ_B be the mappings obtained after application of Lemma 3.1 for the graphs G_A and G_B with normalized proper layouts $\mathcal{L}_A$ and $\mathcal{L}_B$. We are now ready to define the label of u, which is composed of the 14 following fields:

$$\text{label}(u) := \langle u, \pi^{-1}(u),\ \lambda_A(a), \sigma_A(a),\ \lambda_B(b), \sigma_B(b),\ \ldots \rangle$$

$$\text{for all } a \in \{a^-(u), a^+(u), a^+(b^+(u))\} \text{ and } b \in \{b^-(u), b^+(u), b^+(a^+(u))\}.$$

The information about the six vertices needed to compute the 4 distances involved at Step 4 of Theorem 2.1 are available in the labels. So, when implementing the distance decoder the only problem concerns the comparison tests between vertices, involved at Steps 2 and Steps 3 of Theorem 2.1. For these two steps, we need to make comparison tests between some vertices of A or some vertices of B. Typically, in Step 2 for instance, we need to test whether $a^-(v) \leqslant a^-(u)$ or not. The problem is solved thanks to the next lemma.

Lemma 3.4

For all $a_u, a_v \in A$,
$a_u \leqslant a_v$ if and only if:

- *$\sigma_A(a_u) = \sigma_A(a_v)$, or*
- *$\lambda_A(a_u) < \lambda_A(a_v)$, or*
- *$\lambda_A(a_u) = \lambda_A(a_v)$ and $\sigma_A(a_u) < \sigma_A(a_v)$.*

For all $b_u, b_v \in B$,
$b_u \leqslant b_v$ if and only if:

- *$\sigma_B(b_u) = \sigma_B(b_v)$, or*
- *$\lambda_B(b_u) < \lambda_B(b_v)$, or*
- *$\lambda_B(b_u) = \lambda_B(b_v)$ and $\sigma_B(b_u) < \sigma_B(b_v)$.*

Proof. We prove the result only for $a_u, a_v \in A$, the proof being symmetric if $a_u, a_v \in B$.

First, $a_u = a_v$ can be tested using the mapping σ_A using its bijectivity (see Property 3 of Lemma 3.1). So the lemma holds if $a_u = a_v$. So let us assume that $a_u \neq a_v$.

By Lemma 3.3, $a_u < a_v \Leftrightarrow \mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$ for all $a_u, a_v \in A$. So it remains to prove:

$$\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v) \Leftrightarrow \lambda_A(a_u) < \lambda_A(a_v), \text{ or } (\lambda_A(a_u) = \lambda_A(a_v) \text{ and } \sigma_A(a_u) < \sigma_A(a_v)) \quad (7)$$

First, let us show that:

Claim 3.1 *For $a_u \neq a_v$,*

$$\lambda_A(a_u) < \lambda_A(a_v) \;\Rightarrow\; \mathtt{L}_A(a_u) < \mathtt{L}_A(a_v) \;\Rightarrow\; \lambda_A(a_u) \leqslant \lambda_A(a_v)\ . \quad (8)$$

Proof. Let a_0 be the vertex of A with minimum left boundary in $\mathcal{L}_A$. Consider any shortest path $P = w_0, w_1, \ldots, w_k$ from $a_0 = w_0$ to $a_v = w_k$ in G_A. For every i, $\mathtt{L}_A(w_i) < \mathtt{L}_A(w_{i+1})$. Indeed, if $\mathtt{L}_A(w_0) < \cdots < \mathtt{L}_A(w_{i-1}) < \mathtt{L}_A(w_i)$ and $\mathtt{L}_A(w_i) > \mathtt{L}_A(w_{i+1})$, then w_{i-1} would be adjacent to w_{i+1}: contradiction, P is shortest path.

Assume $\lambda_A(a_u) < \lambda_A(a_v)$, i.e., $d_{G_A}(a_0, a_u) < d_{G_A}(a_0, a_v)$ by Property 2 of Lemma 3.1. It is shown in [GP03b], that each set $V_i := \{w \mid d_{G_A}(a_0, w) = i\}$ induces a clique in the proper interval graph G_A. So a_u is adjacent to $w_i \in V_i$ where $i = d_{G_A}(a_0, a_u) < k$. If $\{a_u, w_{i+1}\} \in E$, then one can choose the shortest path P from a_0 to a_v through a_u, and as seen previously, it implies that $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$. So assume that $\{a_u, w_{i+1}\} \notin E$. Thus $\mathtt{L}_A(a_u) \leqslant \mathtt{R}_A(a_u) < \mathtt{L}_A(w_{i+1})$. Previously we have seen that $\mathtt{L}_A(w_i) < \mathtt{L}_A(w_{i+1}) \leqslant \mathtt{L}_A(w_k) = \mathtt{L}_A(a_v)$. Therefore, we have proved the first implication of Eq. (8), that is $\lambda_A(a_u) < \lambda_A(a_v)$ implies $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$.

Let us prove the second implication, and assume $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$. Let i be such that $\mathtt{L}_A(w_{i-1}) < \mathtt{L}_A(a_u) \leqslant \mathtt{L}_A(w_i)$. Since the intervals of w_{i-1} and w_i intersect, a_u is a neighbor of w_{i-1}. Hence $d_{G_A}(a_0, a_u) \leqslant d_{G_A}(a_0, w_{i-1}) + 1 = i$. We have assumed that $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$ thus $i < k$. Therefore, $d_{G_A}(a_0, a_u) < k = d_{G_A}(a_0, a_v)$, i.e., $\lambda_A(a_u) \leqslant \lambda_A(a_v)$, completing the proof of Eq. (8). □

It remains to prove Eq. (7).

$\Rightarrow$ By Eq. (8), $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$ implies $\lambda_A(a_u) \leqslant \lambda_A(a_v)$, i.e., either $\lambda_A(a_u) < \lambda_A(a_v)$ or $\lambda_A(a_u) = \lambda_A(a_v)$. However, By Property 4 of Lemma 3.1, if $\lambda_A(a_u) = \lambda_A(a_v)$, then $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v) \Leftrightarrow \sigma_A(a_u) < \sigma_A(a_v)$. Therefore, $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$ implies $\lambda_A(a_u) < \lambda_A(a_v)$, or $\lambda_A(a_u) = \lambda_A(a_v)$ and $\sigma_A(a_u) < \sigma_A(a_v)$.

$\Leftarrow$ By Eq. (8), $\lambda_A(a_u) < \lambda_A(a_v)$ implies $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$. And, by Property 4 of Lemma 3.1, $\lambda_A(a_u) = \lambda_A(a_v)$ and $\sigma_A(a_u) < \sigma_A(a_v)$ implies $\mathtt{L}_A(a_u) < \mathtt{L}_A(a_v)$, completing this part, and the proof of Lemma 3.4. □

Therefore, according to Lemma 3.1, Lemma 3.4 and Theorem 2.1, the distance decoder takes a constant number of integer additions and comparisons.

3.3 Label size

Since each field of label(u) ranges in $\{0, \ldots, n-1\}$ (formally we need to decrease by 1 the two first fields), the label size is a priori $14\lceil\log n\rceil$. However we will work a little and use correlations between some values of label(u) to reduced it to $9\log n + O(1)$. As we will see all the six λ's values involved in label(u) are strongly related.

The precise description of the final implementation of label(u) is given in the proof of Lemma 3.6, which is based on the following lemma:

Lemma 3.5 *Let $u \in V$. For all $a \in N[u] \cap A$ and $b \in N[u] \cap B$, $|\lambda_A(a) - \lambda_B(b)| \leqslant 1$.*

Proof. Let $a_0 := \min A$, and $b_0 := \min B$. Note that a_0 has minimum left boundary in $\mathcal{L}_A$ since $a^-(a_0) = a_0 = \min A$. Similarly, since $b^-(b_0) = b_0 = \min B$, b_0 has minimum left boundary in $\mathcal{L}_B$. Hence, by Property 2 of Lemma 3.1, $\lambda_A(a) = d_{G_A}(a_0, a)$ for all $a \in A$, and $\lambda_B(b) = d_{G_B}(b_0, b)$ for all $b \in B$. It remains to show that $|d_{G_A}(a_0, a) - d_{G_B}(b_0, b)| \leqslant 1$ for all $a \in N[u] \cap A$ and $b \in N[u] \cap B$.

Note that $\{a_0, b_0\} \in E$, otherwise G would be not connected. Let $k := d_{G_A}(a_0, a)$ and $k' := d_{G_B}(b_0, b)$. We need to show that $|k' - k| \leqslant 1$. By the triangle inequality (observe that a and b are neighbors),

$$d_G(b_0, b) \;\leqslant\; d_G(b_0, a_0) + d_G(a_0, a) + d_G(a, b) \;=\; d_G(a_0, a) + 2 \;.$$

By Lemma 2.5, $d_G(b_0, b) = 2d_{G_B}(b_0, b) = 2k'$ and $d_G(a_0, a) = 2d_{G_A}(a_0, a) = 2k$. So $2k' \leqslant 2k + 2$ implying $k' - k \leqslant 1$. Similarly, $d_G(a_0, a) \leqslant d_G(b_0, b) + 2$ implying $k - k' \leqslant 1$. Therefore, $|k' - k| \leqslant 1$. □

Lemma 3.6 *Labels are of $9\lceil\log n\rceil + 6$ bits at most, and it takes constant time to decode the distance from the labels.*

Proof. Consider any vertex u.

Because $\mathrm{L}_A(a^-(u)) \leqslant \mathrm{L}_A(a^+(u))$, we have that $\lambda_A(a^-(u)) \leqslant \lambda_A(a^+(u))$ (by Eq. (8) in the proof of Lemma 3.4, and using the fact that $a^-(u) \leqslant a^+(u)$). It follows that $d_G(a^-(u), a^+(u)) = 0$ or 2. Thus by Lemma 2.5, $d_{G_A}(a^-(u), a^+(u)) = 0$ or 1. So, $\lambda_A(a^+(u)) - \lambda_A(a^-(u)) \leqslant 1$.

Similarly, $\lambda_A(a^+(u)) \leqslant \lambda_A(a^+(b^+(u)))$. Vertices $a^+(u)$ and $b^+(u)$ are neighbors because, if $u \notin A \cup B$ (this is trivial if $u \in A \cup B$), $a^+(u) \in N^-(u)$ and $b^+(u) \in N^+(u)$. It follows that $d_G(a^+(u), a^+(b^+(u))) = 2$, and $d_{G_A}(a^+(u), a^+(b^+(u))) = 1$ by Lemma 2.5. So, $\lambda_A(a^+(b^+(u))) - \lambda_A(a^+(u)) \leqslant 1$.

We have similarly, $\lambda_B(b^+(u)) - \lambda_B(b^-(u)) \leqslant 1$, and $\lambda_B(b^+(a^+(u))) - \lambda_B(b^+(u)) \leqslant 1$.

For every $u \in V$, we set $\delta(u) := \min\{\lambda_A(a^-(u)), \lambda_B(b^-(u))\}$, and let $\Delta(u)$ be the array with the six entries defined as follows:

$$
\begin{aligned}
\Delta(u)[1] &:= \lambda_A(a^-(u)) - \delta(u) & \Delta(u)[4] &:= \lambda_B(b^-(u)) - \delta(u)\\
\Delta(u)[2] &:= \lambda_A(a^+(u)) - \lambda_A(a^-(u)) & \Delta(u)[5] &:= \lambda_B(b^+(u)) - \lambda_B(b^-(u))\\
\Delta(u)[3] &:= \lambda_A(a^+(b^+(u))) - \lambda_A(a^+(u)) & \Delta(u)[6] &:= \lambda_B(b^+(a^+(u))) - \lambda_B(b^+(u))
\end{aligned}
$$

Given $\delta(u)$ and the array $\Delta(u)$, all the λ's can be retrieved. E.g., $\lambda_B(a^+(b^+(u))) = \delta(u) + \Delta(u)[1] + \Delta(u)[2] + \Delta(u)[3]$.

From Lemma 3.5, the 1st and the 4th entries of $\Delta(u)$ are 0 or 1, and from the above upper bounds, the four other entries are 0 or 1 as well. Therefore, label(u) can be implemented with $\langle u, \pi^{-1}(u), \delta(u), \Delta(u), \sigma_A(a), \sigma_B(b), \ldots \rangle$ for all $a \in \{a^-(u), a^+(u), a^+(b^+(u))\}$ and $b \in \{b^-(u), b^+(u), b^+(a^+(u))\}$.

Let $s := \max\{|A|, |B|\}$. Note that all λ's and σ's are in $\{0, \ldots, s-1\}$. Therefore, the length of this label is at most $2\lceil \log n \rceil + 7\lceil \log s \rceil + 6 \leqslant 9\lceil \log n \rceil + 6$ bits. $\square$

3.4 Computing the labels

We describe the main steps of our procedure to compute all the labels previously defined. We denote by A^- the array containing the values $a^-(u)$ for all $u \in V = \{1, \ldots, n\}$. Moreover precisely, $A^-[u] = a^-(u)$. We define similarly the arrays A^+, B^-, B^+. Observe that the input of our procedure can be obtained in $O(n+m)$ time if the input graph is given only by its adjacency list [MS97, KMMS03], m being the number of edges.

INPUT: a realizer π of a connected permutation graph G of n.
OUTPUT: label(u) for all $u \in V$.

1. Compute π^{-1}.
2. Compute the sets A and B.
3. Compute the arrays A^-, A^+, B^-, B^+.
4. Compute the layouts $\mathcal{I}_A$ and $\mathcal{I}_B$ from A^-, A^+, B^-, B^+.
5. Normalize $\mathcal{I}_A$ and $\mathcal{I}_B$ into $\mathcal{L}_A$ and $\mathcal{L}_B$.
6. Compute the mappings $\lambda_A(w), \sigma_A(w)$ for all $w \in A$, and $\lambda_B(w), \sigma_B(w)$ for all $w \in B$.
7. Compute $\delta(u) := \min\{\lambda_A(a^-(u)), \lambda_B(b^-(u))\}$ and the array $\Delta(u)$ (which is described in the proof of Lemma 3.6), and then compose label(u) for all $u \in V$.

The correctness of the procedure follows from the previous paragraphs. Steps 1,4,5, and 7 clearly take $O(n)$ time. Step 6 takes $O(n)$ time from [GP03b]. So, we only detail, Steps 2 and 3 to show that they also take $O(n)$ time.

By definition $u \in A$ if and only if $N^-(u) = \varnothing$. So, from the graphic representation, we can easly construct A in $O(n)$ time by maintaining the current vertex a_0 with highest y-coordinate, i.e., π^{-1}, as follows:

1. $A := \{1\}$ and $a_0 := 1$

2. For $u = 2$ to n
3. If $\pi^{-1}(u) > \pi^{-1}(a_0)$, then $a_0 := u$ and $A := A \cup \{u\}$

Similarly, B can be computed by maintaining the current vertex b_0 with lowest y-coordinate with initially $B := \{n\}$ and $b_0 := n$, and successively checking from $u = n - 1$ downto 1 whether $\pi^{-1}(u) < \pi^{-1}(b_0)$ or not.

The computation of $a^+(u)$ and $b^-(u)$ for all u can be done as follows:

1. For $u = 1$ to n
2. If $u \in A$, then $a^+(u) := u$ and $a_0 := u$
3. Else $a^+(u) := a_0$

1. For $u = n$ downto 1
2. If $u \in B$, then $b^-(u) := u$ and $b_0 := u$
3. Else $b^-(u) := b_0$

Then, sorting the points of V by their y-coordinate, and this can be done in $O(n)$ time, one can compute $a^-(u)$ and $b^+(u)$ for all u with the similar procedures, completing Step 3:

1. For $\pi^{-1}(u) = n$ downto 1
2. If $u \in A$, then $a^-(u) := u$ and $a_0 := u$
3. Else $a^-(u) := a_0$

1. For $\pi^{-1}(u) = 1$ to n
2. If $u \in B$, then $b^+(u) := u$ and $b_0 := u$
3. Else $b^+(u) := b_0$

Combining this previous algorithm with Lemma 3.6, we obtain:

Theorem 3.1 *Permutation graphs with n vertices enjoy a distance labeling scheme using labels of $9\lceil\log n\rceil + 6$ bits. The distance decoder has constant time complexity, and given a realizer of the graph, i.e., a permutation of $\{1, \ldots, n\}$, all the labels can be computed in $O(n)$ time.*

4 Extensions

Here we propose several extensions to our result.

4.1 All-pair shortest paths

We observe that Theorem 2.1 give us, not only the distance between u and v with $u < v$, but also the first edge incident to u on a shortest path from u to v. This first edge is $\{u, v\}$ if the distance is 1, and is $\{u, a^+(u)\}$ or $\{u, b^+(u)\}$ for the other cases, depending on which particular case occurs. (For instance the edge is $\{u, a^+(u)\}$ if $a^-(v) \leqslant a^-(u)$, by Case 2 and Lemma 2.4). So, adding pointers to the labels of the next vertices, $a^+(u)$ or $b^+(u)$, allows us to answer the first edge on a shortest path from u to v, if $u < v$. It is clear that a similar data-structure can be constructed to support first shortest-path edge queries between v to u with $u < v$. (Case 3 and 4 of Theorem 2.1 needs to be adapted, for instance, by considering the vertices $a^-(b^-(v))$ and $b^-(a^-(v))$ instead of $a^+(b^+(v))$ and $b^+(a^+(v))$).

Overall, doubling our data-structure allows us to support first shortest-path edge queries between any pair of vertices in constant time. It turns out that length-d shortest path can be extracted in $O(d)$ time (following up the first shortest-path edge). Therefore we have:

Theorem 4.1 *Given a realizer of a permutation graph of n vertices, one can construct in $O(n)$ time an $O(n)$ space data-structure supporting all-pair shortest path extraction in linear time (linear in the length of the shortest path returned).*

4.2 Compact routing schemes

Another application of our localized data-structure is the compact routing, whose goal is to design short addresses and succinct local routing tables for each vertex such that the route from any source u to any destination v can be computed from the local table of u and the address of v. By "computing the route" we mean here that the routing algorithm running in u must return the port number of a first shortest-path edge between u and v, that is an integer taken from $\{1, \ldots, \deg(u)\}$. The trivial solution for this problem is the standard routing tables, where each vertex u stores the mapping of the entire possible destination set to a port number, using $O(n \log \deg(u))$ bits for u.

One can use our scheme to provide a better solution as follows: the local table of u is composed of the distance label of u and of a mapping from $N(u)$ to the port numbers of u, which represents $O(\deg(u) \log n)$ bits, whereas addresses consist of the distance label only, so are on $O(\log n)$ bits. This leads to an improvement upon the additive stretch-1 routing scheme of [DL02].

Theorem 4.2 *Permutation graphs have a shortest-path routing scheme with constant time protocol, and with $O(\log n)$ bit addresses, and, for every vertex u, the routing table of u is of size $O(\deg(u) \log n)$ bits. If the graph is bipartite then the size of the local routing tables can be reduced to $O(\log n)$ bit per vertex.*

Proof. Using the previous distance labeling scheme, the routing between non-adjacent vertices u to v can be done with $O(\log n)$ bits/vertex only. Indeed, from Lemma 2.4, the routes are done through $a^+(u)$ or $b^+(u)$ ($a^-(u)$ or $b^-(u)$ if $u > v$), so it suffices that u stores the 4 corresponding port numbers of these neighbors. The only difficulty concerns routing between adjacent vertices.

We can use an $O(\deg(u) \log n)$ bit table for a mapping from $N(u)$ to $\{1, \ldots, \deg(u)\}$. However, if the graph is bipartite, the routing can be simplified, and the port numbers can be permuted so that the memory requirement is $O(\log n)$ bits only for each vertex. In that case $V = A \cup B$ with edges between A and B.

We only detail the port number labeling of a link from a vertex $u \in A$ to a vertex $v \in B$ (the case $u \in B$ and $v \in A$ is symmetric). The port number of the edge $\{u, v\}$ is p if and only if v is the p-th largest neigbor of u. Let $r(b) = |\{w \in B \mid w \leqslant b\}|$ denote the rank of any vertex $b \in B$. Vertex u stores $r(b^-(u))$ and $r(b^+(u))$, whereas v stores in its address its own rank $r(v)$. First, if $r(v) \in [r(b^-(u)), r(b^+(u))]$, then v is adjacent to u and the port number of this edge is $r(v) - r(b^-(u)) + 1$, so the routing can be done through this port. Otherwise, the routing can be done through the port 1 or $\deg(u)$ (i.e., through the neighbors $b^-(u)$ or $b^+(u)$) depending whether $v < u$ or not. Actually, the adress of a vertex v can be even reduced to the pair (r, t) of $\lceil \log n \rceil + 1$ bits, where r is the rank of v in its stable (either in A or in B), and t a boolean indicating whether $v \in A$ or $v \in B$. If we allow to store $|A|$ in all the vertices, we can even use $\lceil \log n \rceil$ bit addresses by mapping the address of v to $r + t \cdot |A|$. □

4.3 Circular permutation graphs

The distance labeling scheme for permutation graphs can be naturally extended to circular permutation graphs. A circular permutation graph is the intersection graph of a family of geodesics of a cylinder between to parallel non-contractible cycles of a cylinder. A permutation graph is simply a circular permutation graph in which the diameter of the cycles is infinite.

Using a similar idea of [ACL95, CLSS98, GP03b], the cylinder can be "unrolled" twice to produce a permutation graph H of size twice the input circular permutation graph G. Each vertex u of G appears twice in H, say as u_1 and u_2. The distance from u to v in G can be computed by taken the minimum of the distances $d_H(u_1, v_1)$ and $d_H(v_1, u_2)$. So, if permutation graphs have an $\ell(n)$ bit distance labeling scheme, then circular permutation graphs enjoy a $2\ell(2n)$ bit distance labeling scheme, that is an $O(\log n)$ bit labeling scheme from Theorem 3.1.

Theorem 4.3 *Circular permutation graphs with n vertices enjoy a distance labeling scheme with $O(\log n)$ bit labels and constant time distance decoder.*

Obviously, Theorem 4.1 and Theorem 4.2 can be similarly extended to obtain a localized and compact data-structure for all-pair shortest path and routing in circular permutation graphs as well.

5 Lower Bounds

5.1 Higher dimensions

A simple counting argument shows that every adjacency labeling scheme, and every distance labeling scheme as well, of any family containing $F(n)$ labeled graphs of n vertices requires some labels of length at least $\frac{1}{n} \log F(n)$ bits (see [KNR92]).

Now, it is not difficult to check that there are $2^{\Theta(n^2)}$ comparability graphs of n vertices. For instance, the family of split graphs, obtained from a bipartite graph in which one part is completed by a clique, clearly contains at least $2^{(n/2)^2}$ graphs. Split graphs are comparability graphs (i.e., comparability graphs of some posets) because their edges can be transitively oriented by: (1) choose any transitive orientation of the clique edges; and (2) orient all the non-clique edges in the same direction (e.g., from the stable to the clique). Therefore, we have:

Proposition 5.1 *There are comparability graphs for which every distance labeling scheme requires $\Omega(n)$ bit labels.*

This latter result is optimal in the sense that there is a distance labeling scheme that guaranties $O(n)$ bit labels for every graph [GPPR04], moreover with a $O(\log\log n)$ time distance decoder.

Adjacency of comparability graphs of d-dimensional posets can be done with labels of $d\lceil\log n\rceil$ bits by simply associating with each vertex its d coordinates, one for each linear extension. So comparability graphs of posets of bounded dimension have $O(\log n)$ bit adjacency labeling. In general, split graphs are comparability graphs of unbounded dimension posets. Therefore, the argument used for Proposition 5.1, a reduction from adjacency labeling scheme for split graphs, cannot be used efficiently for comparability graphs of, say, three-dimensional posets.

We overcome this problem, and show that the behavior between two- and three-dimensional poset is quite different when considering distances. The result is based on a reduction to distance labeling in planar graphs.

Theorem 5.1 *There are comparability graphs of three-dimensional posets for which every distance labeling scheme requires $\Omega(n^{1/3})$ bit labels.*

Proof. For every graph $G = (V, E)$, the vertex/edge inclusion poset is a height one poset $P = (V \cup E, <_G)$ such that $v <_G e$ if and only if $v \in V$, $e \in E$, and e is incident to v. Let H be the comparability graph of P. It is clear that $u, v \in V$ are adjacent in G if and only if there is an edge $e \in E$ incident to u and v, i.e., $d_G(u,v) = 1$ if and only if $d_H(u,v) = 2$. It follows that, for all $u, v \in V$, $d_H(u,v) = 2d_G(u,v)$.

Schnyder in [Sch89] showed that the vertex/edge inclusion poset of every planar graph G has dimension at most three. From [GPPR04], for every distance labeling on the family planar graphs with at most n vertices, there is a planar graph $G_0 = (V_0, E_0)$ with a label of size $\ell(n) = \Omega(n^{1/3})$.

Assume now that the family of comparability graphs of three-dimensional poset over n elements enjoys a distance labeling scheme with labels of length at most $k(n)$, and a scheme with $k(n)$ as smallest as possible. From [GPPR04], $k(n)$ is at most linear in n.

Observe that $k(n)$ is non-decreasing because removing an element of a d-dimensional poset (and thus a vertex of its comparability graph) results a d-dimensional poset as well.

Let P_0 be the vertex/edge poset of G_0, and let H_0 be its comparability graph. H_0 has $|V_0| + |E_0| < 4n$ vertices. For all $u, v \in V_0$, $d_{G_0}(u, v) = \frac{1}{2} d_{H_0}(u, v)$. Therefore, applying the distance labeling scheme only on the vertices of V_0 of H_0 yields a distance labeling for G_0. The labels for the vertices of G_0 are of length at most $k(4n)$ (because $k(n)$ is non-decreasing). It follows that $k(4n) \geqslant \ell(n) = \Omega(n^{1/3})$. Therefore, since $k(n)$ is at most linear in n, we have $k(n) = \Omega(n^{1/3})$. $\square$

5.2 Dimension two

It is not difficult to see that for any connected proper interval graph H with n vertices, there is a permutation graph G of $O(n)$ vertices (actually a bipartite permutation graph) with a subset A with $|A| = n$ corresponding to the vertices of H such that distances in G between the vertices of A are exactly twice all the distances in H. In other words, distance labeling in permutation graphs is at least as hard as in proper interval graphs. So, the $2 \log n - O(\log \log n)$ bit lower bound for distance labels of proper interval graphs [GP03b] holds also for permutation graphs. Actually, it is possible to improve this lower bound to assymptotically $3 \log n$ using the technique of [GP03b]. As we will see, this technique shows that any distance labeling scheme on permutation graphs requires labels of length roughly $\frac{1}{n} \log P(n) + \log n$, where $P(n)$ denotes the number of labeled permutation graphs with n vertices. The main difficulty is to prove that $\frac{1}{n} \log P(n) \sim 2 \log n$.

Let us first present the technique of [GP03b]. An α-graph, for integer $\alpha \geqslant 1$, is a graph H having a pair of vertices (l, r), possibly with $l = r$, such that l and r are of eccentricity at most α. Let $S = (H_0, H_1, \dots, H_k)$ be a sequence of α-graphs, and let (l_i, r_i) denote the pair of vertices that defines the α-graph H_i, for $i \in \{0, \dots, k\}$. For each non-null integer sequence $W = (w_1, \dots, w_k)$, we denotes by $S \circ W$ the graph obtained by attaching a path of length w_i between the vertices r_{i-1} and l_i, for every $i \in \{1, \dots, k\}$ (see Fig. 4).

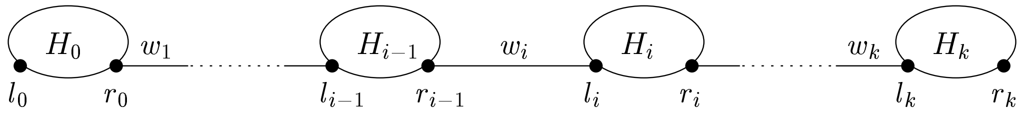

Figure 4: Linking a sequence of α-graphs.

A family $\mathcal{H}$ of graphs is *α-linkable* for $\mathcal{F}$ if every graph of $\mathcal{H}$ is an α-graph of $\mathcal{F}$ and if $S \circ W \in \mathcal{F}$ for every graph sequence S of $\mathcal{H}$ and every non-null integer sequence W.

Lemma 5.1 ([GP03b]) *Let $\mathcal{F}$ be any graph family, and let $\mathcal{H}$ be any α-linkable family for $\mathcal{F}$. If $H(N)$ denotes the number of labeled N-vertex graphs of $\mathcal{H}$, then every distance labeling scheme on the n-vertex graphs of $\mathcal{F}$ requires a label of length at least $\frac{1}{N} \log H(N) + \log N - 9$, where $N = \lfloor n/(\alpha \log n) \rfloor$.*

Lemma 5.2 *Any distance labeling scheme on the family of n-vertex permutation graphs requires a label of length at least $\frac{1}{N}\log P(N-1)+\log N-9$ where $N=\lfloor n/\log n\rfloor$.*

Proof. Let $\mathcal{H}$ be the family of all permutation graphs having an universal vertex, i.e., a vertex connected to all the others. By construction, $\mathcal{H}$ is composed of 1-graphs. Clearly, adding a vertex connected to all other vertices of any permutation graphs, results in a permutation graph. So, if $H(N)$ denotes the number of N-vertex graphs of $\mathcal{H}$, then $H(N)\geqslant P(N-1)$.

Let us show that $\mathcal{H}$ is 1-linkable for the family of permutation graphs. Indeed, for each sequence $S=(H_0,H_1,H_2,\dots)$ of $\mathcal{H}$ and non-null integer sequence $W=(w_1,w_2,\dots)$, one can easily check that $S\circ W$ is a permutation graph from intersection representation depicted on Fig. 5 (the grayed boxes are arbitrary permutation graphs, the bold line being the universal vertex of each H_i).

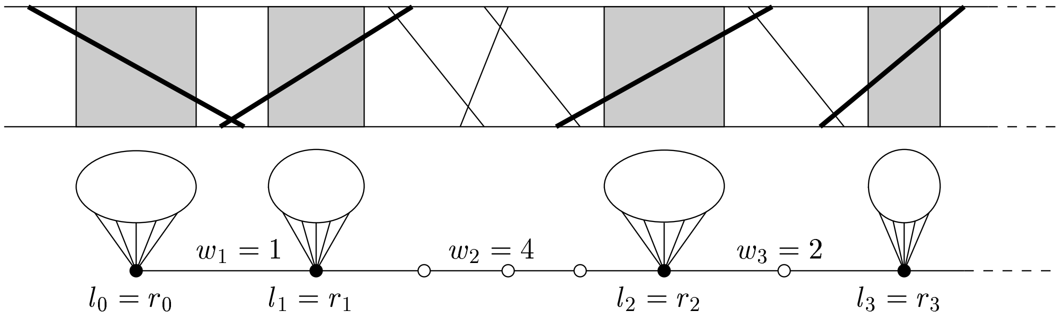

Figure 5: Linking permutation graphs with universal vertices.

From Lemma 5.1, the every distance labeling scheme on the n-vertex permutation graphs requires a label of length at least:

$$\frac{1}{N}\log H(N)+\log N-9 \;\geqslant\; \frac{1}{N}\log P(N-1)+\log N-9$$

where $N=\lfloor n/\log n\rfloor$. □

To conclude, it need to show that $\frac{1}{n}\log P(n)\sim 2\log n$. It is clear that $P(n)\leqslant U(n)\cdot n!\leqslant n!^2\leqslant n^{2n}$ where $U(n)$ denotes the number of unlabeled permutation graphs with n vertices. Hence $\frac{1}{n}\log P(n)\leqslant 2\log n$. However, establishing a lower bound on $P(n)$ is more tricky since an unlabeled permutation graph may have an exponential number of realizers (see Proposition 5.2 below). As there are $n!$ realizers, $U(n)\ll n!$ and thus $P(n)\ll n!^2$.

Proposition 5.2 *There are unlabeled connected permutation graphs of n vertices with $2^{\Omega(n)}$ realizers.*

Proof. With each binary string S of length $2t$ and with t ones, we associate a distinct realizer π_S of a permutation graph G_S as follows. The $n=2t+1$ vertices of G_S are

$a_1, b_1, \ldots, a_t, b_t, c$, and their 2D-coordinates that define the edges of G_S are, for every $i \in \{1, \ldots, t\}$ (see Fig. 6):

- $a_i = 2i - 1$, and $\pi_S^{-1}(a_i) = 2i + S[i]$.
- $b_i = 2i$, and $\pi_S^{-1}(b_i) = 2i + 1 - S[i]$.
- $c = n$, and $\pi_S^{-1}(c) = 1$.

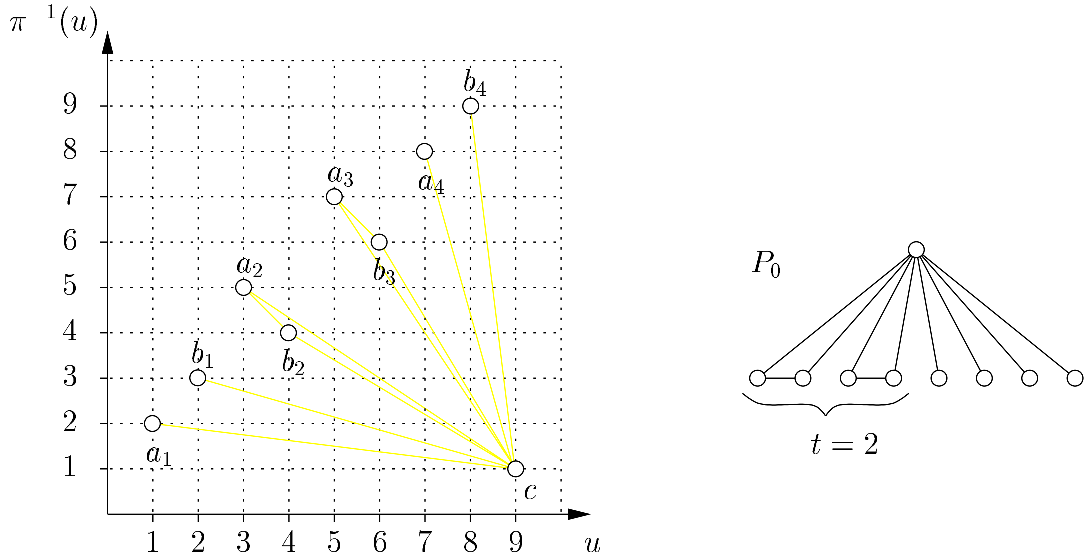

Figure 6: A graph G_S for $S = 0110$ isomorphic to P_0.

This points defines a permutation graph G_S having the realizer π_S. The graph is connected (because vertex c), and a_i is adjacent to b_i if and only if $S[i] = 1$. It is easy to see that each G_S graph is isomorphic to an unique permutation graph P_0 depicted on Fig. 6. The number of realizers for P_0 is $\binom{2t}{t} = \Omega(2^n/\sqrt{n}) = 2^{\Omega(n)}$. □

Theorem 5.2 *The number $P(n)$ of labeled n-vertex connected permutation graphs satisfies $\frac{1}{n} \log P(n) \geqslant 2 \log n - O(\log \log n)$. It follows that there are $2^{\Omega(n \log n)}$ unlabeled n-vertex permutation graphs.*

Proof. Let us overview the proof. We construct a particular family of points of $\mathbb{N}^2$, say $\mathcal{R}_{p,k}$, where p and k are integral parameters such that $p \sim n/\log n$ and $k \sim \log n$. Each set of $\mathcal{R}_{p,k}$ is composed of exactly n points and can be viewed as a permutation of $\{1, \ldots, n\}$: the coordinates of the points range in $\{1, \ldots, n\}$, and there is a unique point for each row and column of the square $\{1, \ldots, n\} \times \{1, \ldots, n\}$. We show that the family of labeled permutation graphs with n vertices having any $R \in \mathcal{R}_{p,k}$ as realizer contains at least $B(n) = (n - \log n)! \cdot |\mathcal{R}_{p,k}|$ labeled graphs. This is based on the fact that for each given permutation graph G of this family, there is a unique way to assign coordinates to each vertex so that the set so constructed points is a realizer $R \in \mathcal{R}_{p,k}$ for G. We conclude with the fact that the family $\mathcal{R}_{p,k}$ contains at least $(p-1)!^k \sim \Theta(n/\log n)^n$ realizers, and thus $B(n) \geqslant \Theta(n)^{n-\log n} \cdot \Theta(n/\log n)^n \geqslant \Theta(n/\log n)^{2n}$, proving that $\frac{1}{n}\log P(n) \geqslant \frac{1}{n}\log B(n) \geqslant 2\log n - O(\log\log n)$. Let us now formalize the proof.

Let $\mathcal{S}_{p,k}$ be the set of all sequences $(S_1, \ldots, S_p)$ of p subsets of size k such that:

1. for every $i \in \{1, \ldots, p\}$, $S_i \subseteq \{p+1, \ldots, p(k+1)\} \setminus \bigcup_{j<i} S_j$, setting $S_0 = \varnothing$; and
2. for every $i \in \{1, \ldots, p-1\}$, for all $x, y \in S_i$ with $x < y$, there exists $z \in S_j$ with some $j > i$ such that $x < z < y$.

Such sequences can be constructed by a greedy algorithm (see the proof of Claim 5.2 for details). For instance, $(\{11, 8, 6\}, \{12, 7, 4\}, \{10, 9, 5\}) \in \mathcal{S}_{3,3}$. Before counting the number of such sequences (cf. Claim 5.2), we show how to associate with each of them a realizer and a permutation graph.

Let $S = (S_1, \ldots, S_p) \in \mathcal{S}_{p,k}$. Hereafter, we denote by $S_i[j]$ the j-th largest element of S_i, $j \in \{1, \ldots, k\}$. We associate with S a set R_S of points of $\mathbb{N}^2$ as follows:

1. there are p points $b_1, \ldots, b_p$, each b_i being of coordinates $((k+1)i, i)$; and
2. there are pk points denoted by $a_{i,j}$ such that, for every $(i,j) \in \{1, \ldots, p\} \times \{1, \ldots, k\}$, $a_{i,j}$ is of coordinates $((k+1)(i-1)+j, S_i[j])$.

In other words, R_S is composed of p strips ordered from left to right, two consecutive strips being delimited by a b_i, and the y-coordinates of the points of the i-th strip are given by S_i. See Fig. 7 for an example.

Let $n = p(k+1)$. Observe that every set R_S with $S \in \mathcal{S}_{p,k}$ contains n points. Also, each R_S can be considered as a realizer, and thus as a permutation graph with n vertices. Indeed, by construction the n points of R_S have coordinates ranging in $\{1, \ldots, n\}$, no two points having the same x-coordinate or y-coordinate.

Let $\mathcal{F}_{p,k}$ be the family of all labeled permutation graphs having a realizer R_S for some $S \in \mathcal{S}_{p,k}$ and such the vertices $a_{p,1}, \ldots, a_{p,k}$ and b_p are labeled respectively $1, \ldots, k+1$. Clearly, $|\mathcal{F}_{p,k}|$ is a lower bound on $P(n)$ since $\mathcal{F}_{p,k}$ is a subset of all labeled connected permutation graphs with n vertices (the connexity of the graphs follows from the ordering of the $a_{i,j}$'s).

By forcing the labeling of $k+1$ vertices, there remains at most $(n-(k+1))! = ((p-1)(k+1))!$ ways to label the other vertices of this graph. Thus $|\mathcal{F}_{p,k}| \leqslant ((p-1)(k+1))! \cdot |\mathcal{S}_{p,k}|$. Actually, and this is the heart of the proof, the equality holds.

Claim 5.1 *For all $p, k \geqslant 2$, $|\mathcal{F}_{p,k}| = ((p-1)(k+1))! \cdot |\mathcal{S}_{p,k}|$.*

Proof. Let G be any labeled graph of $\mathcal{F}_{p,k}$, thus having at least one realizer R_S for some $S \in \mathcal{S}_{p,k}$. Let $L = \{a_{p,1}, \ldots, a_{p,k}, b_p\}$. We said that a vertex x of G is *identified* if we can determine the coordinates of x in R_S. We will show how to identify all the vertices of G

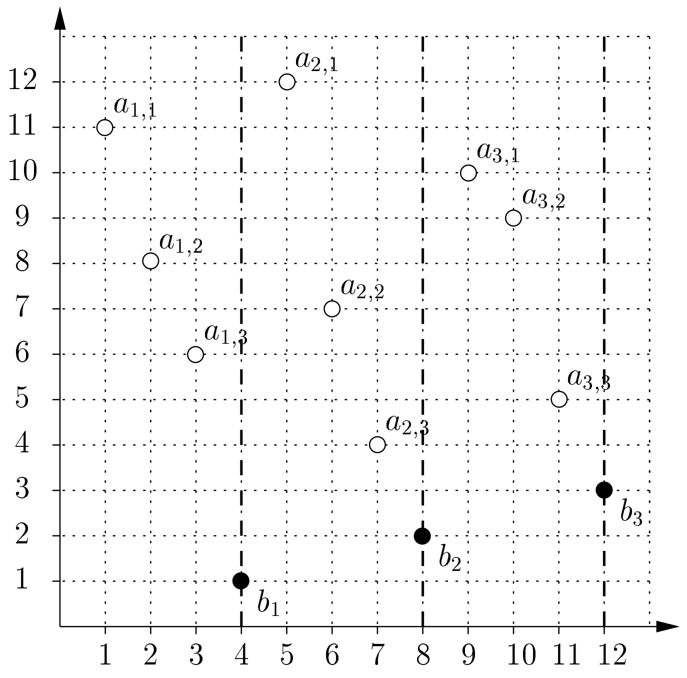

Figure 7: A realizer R_S from the sequence $S = (\{11,8,6\}, \{12,7,4\}, \{10,9,5\}) \in \mathcal{S}_{3,3}$.

(except those of L) in a unique way, i.e., x could not have any other coordinates for every vertex $x \notin L$. Note that once x has been identified, then we know the coordinates of x and also its label. It follows that all such labeled permutation graphs pairewise differ. More formaly, it will prove that there is an one-to-one (or injective) function $\phi : \mathcal{S}_{p,k} \times \mathcal{L}_{p,k} \to \mathcal{F}_{p,k}$, where $\mathcal{L}_{p,k}$ denotes the set of all permutations of $\{|L|+1, \ldots, n\}$. It will imply that $|\mathcal{F}_{p,k}| \geqslant (n-|L|)! \cdot |\mathcal{S}_{p,k}| = ((p-1)(k+1))! \cdot |\mathcal{S}_{p,k}|$ as required.

We now describe how to calculate $\phi^{-1}(G)$. We first show that the vertices $b_1, a_{1,1}, \ldots, a_{1,k}$ (the first strip) can be identified. Then, by induction, we will show how to process all the strips, excepted the last one. Note that the set of vertices L can be determined in G since their label are $\leqslant k+1$.

Identifiying Strip 1. The only vertex not in L having a degree k is b_1. (Recall that $V(G)$ is not reduced to L since $p \geqslant 2$.) Indeed, every vertex $a_{i,j}$ is adjacent to all the $k-1$ vertices of its strip plus b_i, and also plus $b_p \in L$. Thus its degree is $> k$. A vertex b_i is adjacent to all the $a_{i,j}$'s is its strip plus all the $a_{i',j}$'s where $i' < i$. Thus for $i > 1$, the degree of b_i is $> k$, and the degree of b_1 is k exactly.

It follows that the set of vertices of the first strip, say A_1, can be determined (its the b_1's neighbors). To identify each of them we need to precisely compute their coordinates. By construction, $a_{1,1}$ is the highest degree vertex among the vertices of A_1, and more generally, $a_{1,j}$ is the j-th highest degree vertex of A_1. The key point is that all these vertices have necessarily distinct degree and can be uniquely ranked. This is due to Property 2 in the definition of S. The degree of two vertices of a same strip differ by at least one since there is at least one vertex whose y-coordinates is between them.

Note that the y-coordinates of $a_{1,j}$ is exactly its degree minus $j-1$ (the value $j-1$ is due to the North-East quadrant of $a_{1,j}$ that contains $j-1$ neighbors). Therefore, all the vertices of the first strip can be identified.

Identifiying the other strips. The identification of the next strips, $A_2, \ldots, A_{p-1}$, is quite similarly. Assume we have identified all the vertices of the $i-1$ first strips, i.e., we know exactly the coordinates of all the vertices in $A_1 \cup \cdots \cup A_{i-1}$. The identification of A_i can be done as follows. We consider the graph G_i obtained from G by removing the already identified vertices (the strips of index $< i$). If G_i reduced to L, then we are done: all the vertices have been identified. Otherwise, as previously, b_i is the only vertex of degree k in G_i (note here that $i < p$). Then, the x-coordinate of the $a_{i,j}$'s can be determined by ranking their degree: the highest one should by $a_{i,1}$, and so one. Still thanks to Property 2 in the definition of S, the degree must differ. However, unlike previously, the degree in G_i of $a_{i,j}$ does not give directly its y-coordinates. However, it is not difficult to see that its y-coordinates can be precisely computed from the set of points in the strips of index $< i$ and from its degree in G_i (its degree in G_i is actually the number of neighbors located in its South-West quadrant. And the degree in G gives the sum of its number plus its North-East neighborhood).

This procedure can be repeated up to strip $i = p-1$. After that index, the vertex set of G_p reduces to L. Therefore, all the vertices, except those in L, can be identified, and this complete the proof of Claim 5.1. □

Claim 5.2 *For all $p, k \geqslant 2$, $|\mathcal{S}_{p,k}| > (p-1)!^k$.*

Proof. Let $(S_1, \ldots, S_p) \in \mathcal{S}_{p,k}$. By definition (item 1), $S_1, \ldots, S_p$ is a partition of $\{p+1, \ldots, n\}$, where $n = p(k+1)$. Thus S_1 can be constructed by selecting any k-set of $\{p+1, \ldots, n\}$ such that for all distinct elements x, y, we have $|x-y| \geqslant 2$, i.e., there are no consecutive elements. For that, one can first choose a k-set $W \subseteq \{p+1, \ldots, n-(k-1)\}$, and then one can enlarge by one each inter-space between two consecutive $x, y \in W$. The enlarged set W now ranges in the desired interval, i.e., $\{p+1, \ldots, n\}$, since there are $k-1$ inter-spaces in W. This construction is actually a bijection (from S_1 it is easy to rebuild the set W), showing that there are exactly $\binom{n-(k-1)-p}{k}$ ways to construct S_1.

The set S_2 can be constructed similarly by selecting a k-set in $\{p+1, \ldots, n-(k-1)\} \setminus S_1$ and by enlarging the $k-1$ inter-spaces by one. There are $\binom{n-(k-1)-p-k}{k}$ ways to choose S_2 given S_1. More generally, to construct S_i with $i < p$, one can select a k-set in $\{p+1, \ldots, n-(k-1)\} \setminus \bigcup_{j<i} S_j$ and by enlarging the $k-1$ inter-spaces by one. And there are $\binom{n-(k-1)-p-(i-1)k}{k} = \binom{(p-i)k+1}{k}$ ways to choose such S_i, simplifying and plugging $n = p(k+1)$.

Noting that there is only one way for selecting S_p (which does not require property of item 2), the total number of ways to choose a sequence $(S_1, \ldots, S_p)$ is:

$$|\mathcal{S}_{p,k}| \;=\; \prod_{i=1}^{p-1} \binom{(p-i)k+1}{k} \;=\; \prod_{i=1}^{p-1} \binom{ik+1}{k}$$

However,

$$\binom{ik+1}{k} \;=\; \frac{ik+1}{k} \cdot \frac{ik}{k-1} \cdots \frac{ik-j}{k-j-1} \cdots \frac{ik-(k-2)}{1} \;>\; i^k$$

since $ik - j > i(k-j-1)$ for $i \geqslant 1$. It follows that

$$|\mathcal{S}_{p,k}| \;>\; \prod_{i=1}^{p-1} i^k \;=\; (p-1)!^k \,.$$

□

It remains to combine Claim 5.1 and Claim 5.2. Let us choose $p = \lfloor n/\log n \rfloor$. We have $k = n/p - 1 \geqslant \log n - 1$. Using the fact for x large enough $\log(x!) \geqslant x \log x - O(x)$, we obtain

$$|\mathcal{F}_{p,k}| \;>\; ((p-1)(k+1))! \cdot (p-1)!^k,$$

and thus

$$\begin{aligned}
\log|\mathcal{F}_{p,k}| \;&>\; (p-1)(k+1)\log((p-1)(k+1)) + k(p-1)\log(p-1) - O(pk) \\
&\geqslant\; (2pk - O(k+p))\log(p-1) - O(kp) \\
&\geqslant\; 2pk\log p - O((k+p)\log p) - O(kp) \\
&\geqslant\; 2n\log(n/\log n) - O(n) \\
&\geqslant\; 2n\log n - O(n \log\log n)
\end{aligned}$$

This completes the proof of Theorem 5.2. □

Theorem 5.3 *Any distance labeling scheme on the family of n-vertex permutation graphs requires a label of length at least $3\log n - O(\log\log n)$ bits.*

Proof. Using the lower bound on $\frac{1}{N-1}\log P(N-1)$ from Theorem 5.2, we have:

$$\begin{aligned}
\frac{1}{N}\log P(N-1) \;&=\; \frac{N-1}{N} \cdot \frac{1}{N-1}\log P(N-1) \\
&\geqslant\; 2\log(N-1) - \frac{1}{N} \cdot \frac{1}{N-1} \cdot \log P(N-1) - O(\log\log N)
\end{aligned}$$

Clearly, $P(N) \leqslant N!^2$ thus $\log P(N-1) \leqslant 2(N-1)\log(N-1)$ and:

$$\frac{1}{N} \cdot \frac{1}{N-1}\log P(N-1) \;\leqslant\; \frac{1}{N} \cdot 2\log(N-1) \;<\; 1 \,.$$

Using the fact that $N \leqslant n/\log n$ and $N-1 \geqslant n/(2\log n)$,

$$\begin{aligned}\frac{1}{N}\log P(N-1) &\geqslant 2\log(n/(2\log n)) - O(\log\log n)\\ &\geqslant 2\log n - O(\log\log n)\end{aligned}$$

By Lemma 5.2, the lower bound on the label length is:

$$\frac{1}{N}\log P(N-1) + \log N - 9 \geqslant 3\log n - O(\log\log n)\ .$$

□

6 Conclusion and Discussion

In this paper we have looked for a large family of graphs supporting distance labeling scheme with $O(\log n)$ bit labels, and more generally with $O(f(d)\log n)$ bit labels where d is some invariant of the family. In particular we have shown that comparability graphs of posets of dimension d enjoy an $O(\log n)$ bit distance labeling scheme for $d \leqslant 2$, and require $\Omega(n^{1/3})$ bit label in the worst-case if $d \geqslant 3$.

An interesting direction for further research would be to discover other large families of graphs supporting distance labeling schemes with logarithmic or poly-logarithmic labels (in n, the number of vertices). To our best knowledge, only interval and permutation graphs (and their immediate generalizations into circular-arc and circular permutation graphs) have $O(\log n)$ bit label schemes. Trees, bounded tree-width and bounded clique-width graphs have $O(\log^2 n)$ bit distance labeling schemes.

Intersection graphs, like interval or permutation graphs, seem a priori good candidates for this search, essential because such graphs are "simple" in the Kolmogorov Complexity sense: they support short programs to decide adjacency. However we moderate this feeling because many intersection graph families contain planar graphs. Although planar graphs have also short adjacency labels, $3\log n + O(\log^* n)$ bits per vertex from [AR02], it is known that every distance labeling scheme on this family requires some labels of $\Omega(n^{1/3})$ bits [GPPR04]. Some generalizations of interval graphs, namely interval number-3 graphs (the intersection graphs of the union of 3 intervals) and boxicity-3 graphs (the intersection graphs of the Cartesian product of 3 intervals), both contain planar graphs [SW83, Tho86]. So, the design of distance labeling with $O(\log n)$ bit labels (and related problems like all-pair shortest path and compact routing) remains open for these generalizations only for trapezoid graphs (see [MM99, Ch.7]), boxicity-2, and interval number-2 graphs.

Another direction is to extend our data-structure in a dynamic setting.

References

[ABR03] Stephen Alstrup, Philip Bille, and Theis Rauhe. Labeling schemes for small distances in trees. In 14^{th} *Symposium on Discrete Algorithms (SODA)*, pages 689–698. ACM-SIAM, January 2003.

[ACL95] Mikhail J. Atallah, Danny Z. Chen, and D. T. Lee. An optimal algorithm for shortest paths on weighted interval and circular-arc graphs, with applications. *Algorithmica*, 14(5):429–441, November 1995.

[AR02] Stephen Alstrup and Theis Rauhe. Small induced-universal graphs and compact implicit graph representations. In 43^{rd} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 53–62. IEEE Computer Society Press, November 2002.

[BFR71] K. A. Baker, P. C. Fishburn, and F. S. Roberts. Partial orders of dimension 2. *Networks*, 2:11–28, 1971.

[BLS99] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes – A survey.* SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1999.

[CDV04] Victor D. Chepoi, Feodor F. Dragan, and Yann Vaxes. Distance and routing labeling schemes for non-positively curved plane graphs. *Journal of Algorithms*, 2004. To appear.

[CKN+95] Derek G. Corneil, H. Kim, S. Natarajan, Stephan Olariu, and A. P. Sprague. Simple linear time algorithm of unit interval graphs. *Information Processing Letters*, 55(2):99–104, 1995.

[CLSS98] Danny Z. Chen, D. T. Lee, R. Sridhar, and Chandra N. Sekharan. Solving the all-pair shortest path query problem on interval and circular-arc graphs. *Networks*, 31(4):249–257, 1998.

[COS97] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. Asteroidal triple-free graphs. *SIAM Journal on Discrete Mathematics*, 10(3):399–430, August 1997.

[CV03] Bruno Courcelle and Rémi Vanicat. Query efficient implementation of graphs of bounded clique-width. *Discrete Applied Mathematics*, 131:129–150, 2003.

[CZ00] S. Chaudhuri and Christos D. Zaroliagis. Shortest paths in digraphs of small treewidth. Part I: Sequential algorithms. *Algorithmica*, 27:212–226, 2000.

[dFHMPdM95] C. M. de Figueiredo Herrera, J. Meidanis, and C. Picinin de Mello. A linear-time algorithm for proper interval recognition. *Information Processing Letters*, 56(3):179–184, 1995.

[DHZ00] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29:1740–1759, 2000.

[Dji96] Hristo N. Djidjev. Efficient algorithms for shortest path queries in planar digraphs. In 22^{nd} *International Workshop on Graph - Theoretic Concepts in Computer Science (WG)*, volume 1197 of Lecture Notes in Computer Science, pages 151–165. Springer, June 1996.

[DL02] Feodor F. Dragan and Irina Lomonosov. New routing schemes for interval graphs, circular-arc graphs, and permutation graphs. In 14^{th} *IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, pages 78–83, November 2002.

[DM41] B. Dushnik and E.W. Miller. Partially ordered sets. *Amer. J. Math.*, 63:600–610, 1941.

[GKK$^+$01] Cyril Gavoille, Michal Katz, Nir A. Katz, Christophe Paul, and David Peleg. Approximate distance labeling schemes. In F. Meyer auf der Heide, editor, 9^{th} *Annual European Symposium on Algorithms (ESA)*, volume 2161 of Lecture Notes in Computer Science, pages 476–488. Springer, August 2001.

[Gol80] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, Harcourt Brace Jovanovich, Academic Press edition, 1980.

[GP03a] Cyril Gavoille and Christophe Paul. Distance labeling scheme and split decomposition. *Discrete Mathematics*, 273(1-3):115–130, 2003.

[GP03b] Cyril Gavoille and Christophe Paul. Optimal distance labeling schemes for interval and circular-arc graphs. In G. Di Battista and U. Zwick, editors, 11^{th} *Annual European Symposium on Algorithms (ESA)*, volume 2832 of Lecture Notes in Computer Science, pages 254–265. Springer, September 2003.

[GP03c] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Journal of Distributed Computing*, 16:111–120, May 2003. PODC 20-Year Special Issue.

[GPPR04] Cyril Gavoille, David Peleg, Stéphane Pérennès, and Ran Raz. Distance labeling in graphs. *Journal of Algorithms*, 53(1):85–112, 2004.

[KKP00] Michal Katz, Nir A. Katz, and David Peleg. Distance labeling schemes for well-separated graph classes. In H. Reichel and S. Tison, editors, 17^{th} *Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 1770 of Lecture Notes in Computer Science, pages 516–528. Springer Verlag, February 2000.

[KMMS03] Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. In 14^{th} *Symposium on Discrete Algorithms (SODA)*, pages 158–167. ACM-SIAM, January 2003.

[KNR92] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5:596–603, 1992.

[KRRS94] P. Klein, Satish Rao, M. Rauch, and S. Subramanian. Faster shortest-path algorithms for planar graphs. In 26^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 27–37, May 1994.

[MM99] Terry A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. SIAM Monographs on Discrete Mathematics and Applications, 1999.

[MS97] Ross M. McConnell and Jeremy P. Spinrad. Linear-time transitive orientation. In 8^{th} *Symposium on Discrete Algorithms (SODA)*, pages 19–25. ACM-SIAM, January 1997.

[Pel00] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory*, 33:167–176, 2000.

[RLDL94] C. Rhee, Y. Daniel Liang, S. K. Dhall, and S. Lakshmivarahan. Efficient algorithms for finding depth-first and breadth-first search trees in permutation graphs. *Information Processing Letters*, 49(1):45–50, January 1994.

[Sch89] Walter Schnyder. Planar graphs and poset dimension. *Order*, 5:323–343, 1989.

[SW83] Edward R. Scheinerman and Douglas B. West. Interval number of a planar graph: Three intervals suffice. *Journal of Combinatorial Theory, Series B*, 35:224–239, 1983.

[Tal04] Kunal Talwar. Bypassing the embedding: Algorithms for low dimensional metrics. In 36^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 281–290, June 2004.

[Tho86] Carsten Thomassen. Interval representations of planar graphs. *Journal of Combinatorial Theory, Series B*, 40:9–20, 1986.

[Tho01] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In 42^{nd} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 242–251. IEEE Computer Society Press, October 2001.

[TZ01] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In 33^{rd} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192, July 2001.

[Yan82] Mihalis Yannakakis. The complexity of the parital order dimension problem. *SIAM J. Alg. Discrete Methods*, 3:351–358, 1982.