Localized and Compact Data-Structure for Comparability Graphs (Extended Abstract)

Fabrice Bazzaro* and Cyril Gavoille*

LaBRI UMR CNRS 5800, Université Bordeaux 1,
33405 Talence Cedex, FRANCE.

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. 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 (we prove that for comparability graphs of three-dimensional posets, every distance labeling scheme requires $\Omega(n^{1/3})$ bit labels).

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.

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 [5]. 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 [15]. It is NP-complete to recognize posets of dimension three [33] whereas linear time (linear in the size of the relation) algorithms exist for two-dimensional posets [26]. Actually, the comparability graphs of two-dimensional posets are exactly the permutation graphs, namely the intersection graphs of straight segments between two parallel lines [3]. Intersection graphs are graphs in which vertices are mapped to objects, with the vertices

* The two authors are supported by the project "PairAPair" of the ACI Masses de Données. {*bazzaro,gavoille*}*@labri.fr*

defined to be adjacent if and only if the corresponding objects have nonempty intersection. See [27] for a comprehensive introduction to the intersection graphs.

This paper deals with the problem of the distance computation and distributed abilities of comparability 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 surveyed and widely discussed in [19].

We are especially interested in the distance labeling problem, introduced in [28]. 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.

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 [13], 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 [12, 25, 31], bounded tree-width graphs [6], interval graphs [2, 7, 18], 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 [28], it generalizes adjacency labeling [1, 23] 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 [20], and that trees [1, 28], bounded tree-width graphs [20], distance-hereditary graphs [17], bounded clique-width graphs [11], some non-positively curved plane graphs [8], 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 [16, 30–32].

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 [20], implying that all the results mentioned above are tight as well, excepted for interval and permutation graphs that does not contain trees. Recently, [18] 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 [29]. 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.

Our results. In this paper, we are only interested 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 [15], as an intersection graph of segments between two parallel lines [21] or as an Asteroidal Triple-free graph [10].

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 [26] 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 [24] 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 [14]. 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.

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-East or

North-West 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-West quadrant (call A), and those with no neighbors in their South-East 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 difficulty 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.

Due to space limitation, the proofs can be found in [4].

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. 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 [21]. 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 $\pi^{-1}(u) > \pi^{-1}(v)$.

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 [3].

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 $(u, \pi^{-1}(u))$.

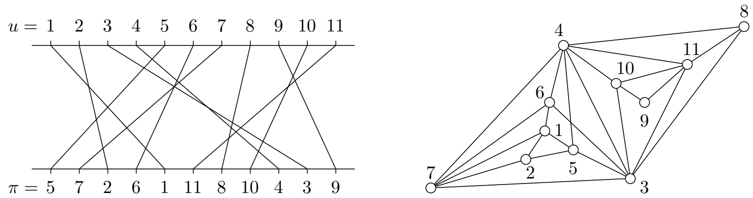

Fig. 1. A permutation graph G with the realizer $\pi = (5, 7, 2, 6, 1, 11, 8, 10, 4, 3, 9)$.

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 $N^+(u) := \{v \in N(u) \mid v > u\}$ and $N^-(u) := \{v \in N(u) \mid v < u\}$ (see left side of Fig. 2).

We distinguish two particular subsets of vertices, A and B depicted in black and gray on Fig. 2, defined by $A := \{u \in V \mid N^-(u) = \varnothing\}$ and $B := \{u \in V \mid N^+(u) = \varnothing\}$. 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 1. *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 $N[u] \cap B = [b^-(u), b^+(u)] \cap B$.*

According to Lemma 1, $N[u] \cap A$ 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 $b^+(u)$.

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. As we will see later in Lemma 4, G_A and G_B are proper interval graphs.

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

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

E.g., in our example, $d_G(1, 9) = 4$ because $1, 9 \in A$ and $d_{G_A}(1, 9) = 2$. By Lemma 2, 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 1. *Let u, v be two vertices with $u < v$. Then,*

1. *if $\pi^{-1}(u) > \pi^{-1}(v)$, then $d_G(u, v) = 1$;*

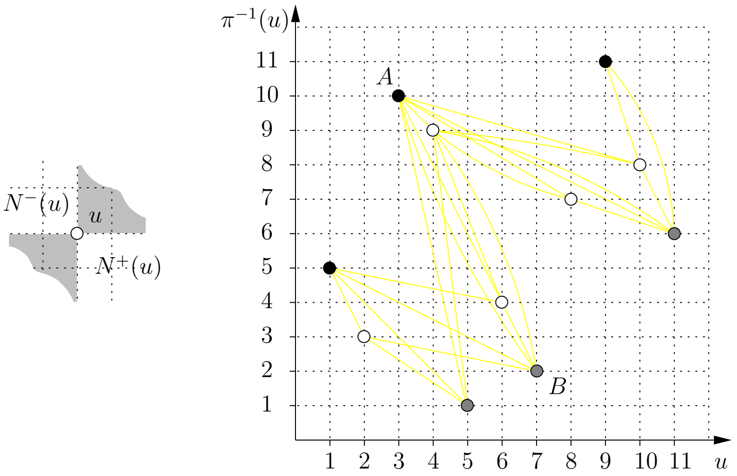

Fig. 2. Graphic representation of the permutation graph of Fig. 1.

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:*
   - $2 + 2d_{G_B}(b^+(u), b^-(v))$
   - $3 + 2d_{G_A}(a^+(b^+(u)), a^-(v))$
   - $2 + 2d_{G_A}(a^+(u), a^-(v))$
   - $3+2d_{G_B}(b^+(a^+(u))), b^-(v))$

3 Implementation of the Scheme

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, we will use as a sub-routine the distance labeling scheme of [18] for proper interval graphs that we need to detail. To compute all the labels in $O(n)$ time, the scheme of [18] 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 [9, 22] 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 [18], 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 [18]. 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 ([18]). *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 $\sigma_H(x) < \sigma_H(y)$.*

Lemma 4. *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.*

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 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 [18] (Lemma 3) 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:

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

Similary, left boundaries of B are ordered such that $\mathrm{L}_B(u) < \mathrm{L}_B(v)$ 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 $O(n)$. We can prove the following properties for the layouts $\mathcal{L}_A$ and $\mathcal{L}_B$:

Lemma 5. *For all $u, v \in A$, $u < v$ if and only if $\mathrm{L}_A(u) < \mathrm{L}_A(v)$. Similarly, for all $u, v \in B$, $u < v$ if and only if $\mathrm{L}_B(u) < \mathrm{L}_B(v)$.*

Let λ_A, σ_A and λ_B, σ_B be the mappings obtained after application of Lemma 3 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:

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

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

The information about the six vertices needed to compute the 4 distances involved at Step 4 of Theorem 1 is 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 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 6.

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* $\sigma_B(b_u) < \sigma_B(b_v)$.

Therefore, according to Lemma 3, Lemma 6 and Theorem 1, the distance decoder takes a constant number of integer additions and comparisons.

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 7, which is based on the following fact: Let $u \in V$. For all $a \in N[u] \cap A$ and $b \in N[u] \cap B$, $|\lambda_A(a) - \lambda_B(b)| \leqslant 1$.

Lemma 7. *Labels are of $9 \lceil \log n \rceil + 6$ bits at most, and it takes constant time to decode the distance from the labels.*

In [4] we show that label(u) for all $u \in V$ can be computed in linear time once the realizer π is given. Combining this with Lemma 7, we obtain:

Theorem 2. *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

The full explanations of the following results are presented in [4].

Theorem 3. *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).*

Theorem 4. *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. Moreover, given a realizer of the graph, addresses and routing tables can be constructed in $O(n)$ time.*

Theorem 5. *Circular permutation graphs with n vertices enjoy a distance labeling scheme with $O(\log n)$ bit labels and constant time distance decoder.*

Theorem 6. *There are comparability graphs of three-dimensional posets for which every distance labeling scheme requires $\Omega(n^{1/3})$ bit labels.*

Theorem 7. *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.*

Theorem 8. *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.*

References

1. S. Alstrup, P. Bille, and T. Rauhe, *Labeling schemes for small distances in trees*, in 14^{th} Symp. on Disc. Algorithms (SODA), ACM-SIAM, 2003, pp. 689–698.
2. M. Atallah, D. Chen, and D. Lee, *An optimal algorithm for shortest paths on weighted interval and circular-arc graphs, with applications*, Algorithm., 14 (1995).
3. K. A. Baker, P. C. Fishburn, and F. S. Roberts, *Partial orders of dimension 2*, Networks, 2 (1971), pp. 11–28.
4. F. Bazzaro and C. Gavoille, *Localized and compact data-structure for comparability graphs*, RR-1343-05, LaBRI, University of Bordeaux 1, Feb. 2005.
5. A. Brandstädt, V. B. Le, and J. P. Spinrad, *Graph Classes – A survey*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, 1999.
6. S. Chaudhuri and C. D. Zaroliagis, *Shortest paths in digraphs of small treewidth. Part I: Sequential algorithms*, Algorithmica, 27 (2000), pp. 212–226.
7. D. Chen, D. Lee, R. Sridhar, and C. Sekharan, *Solving the all-pair shortest path query problem on interval and circular-arc graphs*, Networks, 31 (1998).
8. V. D. Chepoi, F. F. Dragan, and Y. Vaxes, *Distance and routing labeling schemes for non-positively curved plane graphs*, J. of Algorithms, (2005).
9. D. G. Corneil, H. Kim, S. Natarajan, S. Olariu, and A. P. Sprague, *Simple linear time algorithm of unit interval graphs*, Inf. Proc. Letters, 55 (1995).

10. D. G. Corneil, S. Olariu, and L. Stewart, *Asteroidal triple-free graphs*, SIAM Journal on Discrete Mathematics, 10 (1997), pp. 399–430.
11. B. Courcelle and R. Vanicat, *Query efficient implementation of graphs of bounded clique-width*, Discrete Applied Mathematics, 131 (2003), pp. 129–150.
12. H. N. Djidjev, *Efficient algorithms for shortest path queries in planar digraphs*, in 22^{nd} WG, vol. 1197 of LNCS, Springer, June 1996, pp. 151–165.
13. D. Dor, S. Halperin, and U. Zwick, *All-pairs almost shortest paths*, SIAM Journal on Computing, 29 (2000), pp. 1740–1759.
14. F. F. Dragan and I. Lomonosov, *New routing schemes for interval graphs, circular-arc graphs, and permutation graphs*, in 14^{th} PDCS, Nov. 2002, pp. 78–83.
15. B. Dushnik and E. W. Miller, *Partially ordered sets*, American Journal of Mathematics, 63 (1941), pp. 600–610.
16. C. Gavoille, M. Katz, N. A. Katz, C. Paul, and D. Peleg, *Approximate distance labeling schemes*, in 9^{th} ESA, vol. 2161 of LNCS, 2001, pp. 476–488.
17. C. Gavoille and C. Paul, *Distance labeling scheme and split decomposition*, Discrete Mathematics, 273 (2003), pp. 115–130.
18. ———, *Optimal distance labeling schemes for interval and circular-arc graphs*, in 11^{th} ESA, vol. 2832 of LNCS, Springer, Sept. 2003, pp. 254–265.
19. C. Gavoille and D. Peleg, *Compact and localized distributed data structures*, J. of Distributed Computing, 16 (2003), pp. 111–120. PODC 20-Year Special Issue.
20. C. Gavoille, D. Peleg, S. Pérennès, and R. Raz, *Distance labeling in graphs*, Journal of Algorithms, 53 (2004), pp. 85–112.
21. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, Harcourt Brace Jovanovich, Academic Press ed., 1980.
22. C. M. Herrera de Figueiredo, J. a. Meidanis, and C. Picinin de Mello, *A linear-time algorithm for proper interval recognition*, Information Processing Letters, 56 (1995), pp. 179–184.
23. S. Kannan, M. Naor, and S. Rudich, *Implicit representation of graphs*, SIAM Journal on Discrete Mathematics, 5 (1992), pp. 596–603.
24. M. Katz, N. Katz, and D. Peleg, *Distance labeling schemes for well-separated graph classes*, in 17^{th} STACS, vol. 1770 of LNCS, 2000, pp. 516–528.
25. P. N. Klein, S. Rao, M. R. Henzinger, and S. Subramanian, *Faster shortest-path algorithms for planar graphs*, J. of Comp. and Sys. Sci., 55 (1997), pp. 3–23.
26. R. M. McConnell and J. P. Spinrad, *Linear-time transitive orientation*, in 8^{th} Symp. on Discrete Algorithms (SODA), ACM-SIAM, Jan. 1997, pp. 19–25.
27. T. A. McKee and F. R. McMorris, *Topics in Intersection Graph Theory*, SIAM Monographs on Discrete Mathematics and Applications, 1999.
28. D. Peleg, *Proximity-preserving labeling schemes*, Journal of Graph Theory, 33 (2000), pp. 167–176.
29. C. J. Rhee, Y. D. 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 (1994), pp. 45–50.
30. K. Talwar, *Bypassing the embedding: Algorithms for low dimensional metrics*, in 36^{th} STOC, ACM Press, June 2004, pp. 281–290.
31. M. Thorup, *Compact oracles for reachability and approximate distances in planar digraphs*, in 42^{nd} FOCS, IEEE Computer Society Press, Oct. 2001, pp. 242–251.
32. M. Thorup and U. Zwick, *Approximate distance oracles*, in 33^{rd} Annual ACM Symp. on Theory of Computing (STOC), ACM Press, July 2001, pp. 183–192.
33. M. Yannakakis, *The complexity of the parital order dimension problem*, SIAM Journal on Algebraic and Discrete Methods, 3 (1982), pp. 351–358.