

Optimal Distance Labeling for Permutation Graphs

Paweł Gawrychowski* and Wojciech Janczewski†

University of Wrocław

Abstract

A permutation graph is the intersection graph of a set of segments between two parallel lines. In other words, they are defined by a permutation π on n elements, such that u and v are adjacent if and only if $u < v$ but $\pi(u) > \pi(v)$. We consider the problem of computing the distances in such a graph in the setting of informative labeling schemes.

The goal of such a scheme is to assign a short bitstring $\ell(u)$ to every vertex u , such that the distance between u and v can be computed using only $\ell(u)$ and $\ell(v)$, and no further knowledge about the whole graph (other than that it is a permutation graph). This elegantly captures the intuition that we would like our data structure to be distributed, and often leads to interesting combinatorial challenges while trying to obtain lower and upper bounds that match up to the lower-order terms.

For distance labeling of permutation graphs on n vertices, Katz, Katz, and Peleg [STACS 2000] showed how to construct labels consisting of $\mathcal{O}(\log^2 n)$ bits. Later, Bazzaro and Gavaille [Discret. Math. 309(11)] obtained an asymptotically optimal bounds by showing how to construct labels consisting of $9 \log n + \mathcal{O}(1)$ bits, and proving that $3 \log n - \mathcal{O}(\log \log n)$ bits are necessary. This however leaves a quite large gap between the known lower and upper bounds. We close this gap by showing how to construct labels consisting of $3 \log n + \mathcal{O}(\log \log n)$ bits.

*gawry@cs.uni.wroc.pl

†wojciech.janczewski@cs.uni.wroc.pl

1 Introduction

Geometric intersection graph is a graph where each vertex corresponds to an object in the plane, and two such vertices are adjacent when their corresponding objects have non-empty intersection. Usually, one puts some restriction on the objects, for example they should be unit disks. The motivation for such a setup is twofold. First, it allows for modelling many practical problems. Second, it leads to nice combinatorial questions. This is a large research area, and multiple books/survey are available [21, 36, 42, 46] (to name just a few).

In this paper, we are interested in one of the most basic classes of geometric intersection graphs, namely permutation graphs. A permutation graph is the intersection graph of a set of segments between two parallel lines. An alternative (and more formal) definition is as follows. A graph $G = (V, E)$, where $V = \{1, 2, \dots, n\}$, is a permutation graph if there exists a permutation π on n elements, such that u and v are adjacent exactly when $u < v$ but $\pi(u) > \pi(v)$. See Figure 1 for a small example.

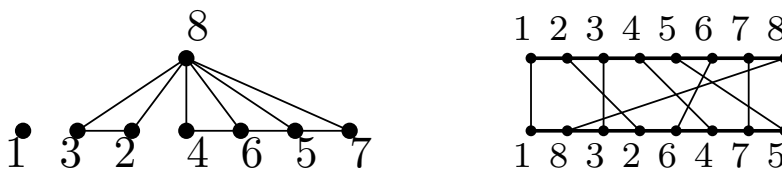


Figure 1: Permutation graph described by $\pi = 18326475$.

Permutation graphs admit a few alternative definitions. For example, G is a permutation graph if and only if both G and its complement are comparability graphs [23]. Alternatively, they can be defined as comparability graphs of two-dimensional posets [11]. From the algorithmic point of view, the motivation for studying such graphs is that they can be recognised in linear time [41], and multiple problems that are computationally difficult on general graphs admit efficient algorithms on permutation graphs [17, 18, 43]. In this paper, we consider constructing a distributed data structure capable of efficiently reporting the distance between two given vertices of a permutation graph.

Informative labeling schemes. We work in the mathematically elegant model of informative labeling schemes, formally introduced by Peleg [48]. Such a scheme is meant to represent graphs in an extremely distributed way. Instead of storing a single global data structure, a scheme assigns to each vertex v of a given graph a binary string $\ell(v)$, called a label. Later, given the labels of two vertices (and no additional information about the graph), we should be able to compute some fixed function on those two vertices.

In the context of informative labeling schemes, the first function that one usually considers is adjacency, where we simply want to decide whether the two vertices in question are neighbours in the graph. As observed by Kannan, Naor, and Rudich [37], this is equivalent to finding a so-called vertex-induced universal graph, and predates the more general notion of informative labeling schemes. Non-trivial adjacency labeling schemes have been constructed for many classes of graphs, for example undirected, directed, and bipartite graphs [10], graphs of bounded degree [22], trees [6], planar graphs [14, 20], comparability graphs [13], or general families of hereditary graphs [15, 33]. In every case, the length of each *individual* label is much smaller than the size of a centralised structure, often by a factor close to $\Theta(n)$, i.e., we are able to evenly distribute the whole adjacency information. Other functions considered in the context of labeling schemes are ancestry in trees [25, 31], routing [24, 30, 50] or connectivity [39]. However,

from the point of view of possible applications, the next most natural question is that of distance labelings, where given labels of two vertices we need to output the exact distance between them in a graph. This properly generalises adjacency and usually needs much longer labels.

Distance labelings. The size of a labeling scheme is defined by the maximum length of any label assigned by the encoder. If not stated otherwise, all graphs are unweighted and undirected, and consist of n vertices. For general undirected graphs, Alstrup, Gavoille, Halvorsen, and Petersen [8] constructed distance labeling of size $(\log 3)n/2 + o(n)$, while the known lower bound is $\lceil n/2 \rceil$ bits. Alstrup, Dahlgaard, Knudsen, and Porat [7] describe a slightly sublinear $o(n)$ -bits labeling for sparse graphs. In case of planar graphs, scheme of size $\mathcal{O}(\sqrt{n})$ bits is presented by Gawrychowski and Uznanski [32], and the known lower bound is $\Omega(n^{1/3})$ bits. Shur and Rubinchik [49] designed a scheme using $n^{1.5}/\sqrt{6} + \mathcal{O}(n)$ distinct labels for families of cycles, against a lower bound of $\Omega(n^{4/3})$ [40]. For trees, we do not need a polynomial number of bits, as they can be labeled for distances using only $1/4 \log^2 n + o(\log^2 n)$ bits as shown by Freedman, Gawrychowski, Nicholson, and Weimann [26], which is optimal up to the second-order terms [9]. Of course, the interesting question is to find natural classes of graphs that admit small distance labeling schemes.

Distance labeling for permutation graphs. Katz, Katz and Peleg [38] presented distance labeling scheme of size $\mathcal{O}(\log^2 n)$ for interval and permutation graphs. This was improved by Gavoille and Paul to $5 \log n$ labeling for interval graphs [28], with a lower bound of $3 \log n - \mathcal{O}(\log \log n)$. Very recently, He and Wu [35] presented tight $3 \log n + \mathcal{O}(\log \log n)$ distance labeling for interval graphs. For connected permutation graphs, Bazzaro and Gavoille in [12] showed a distance labeling scheme of size $9 \log n + \mathcal{O}(1)$ bits, and a lower bound of $3 \log n - \mathcal{O}(\log \log n)$. As noted in their work, this is especially interesting as there are very few hereditary graph classes that admit distance labeling schemes of size $o(\log^2 n)$. As our main result, we close the gap between the lower and upper bounds on the size of distance labeling for permutation graph, by showing the following theorem.

Theorem 1. *There is a distance labeling scheme for permutation graphs with n vertices using labels of size $3 \log n + \mathcal{O}(\log \log n)$ bits. The distance decoder has constant time complexity, and labels can be constructed in polynomial time.*

On constants. We stress that in the area of informative labeling scheme, it is often relatively easy to obtain asymptotically optimal bounds on the size of a scheme, and the real challenge is to determine the exact constant for the higher-order term. This has been successfully done for multiple classes, e.g. distance labeling for trees, where $\mathcal{O}(\log^2 n)$ [47] was first improved to $(1/2) \log^2 n$ [9] and then $(1/4)(\log^2 n) + o(\log^2 n)$ [26], optimal up to second-order term. Adjacency labeling for trees is a particularly good example, with the first scheme having a size of $6 \log n$ based on [45], then $4 \log n$ [37], $(2 + o(1)) \log n$ [27], $(4/3 + o(1)) \log n$ [14], finally $\log n + \mathcal{O}(\sqrt{\log n \log \log n})$ [20] and $\log n + \mathcal{O}(\sqrt{\log n})$ [29] were presented, the last two being optimal up to the second order terms. For adjacency in bounded-degree graphs with odd Δ , initial $(\Delta/2 + 1/2) \log n + \mathcal{O}(1)$ [16] was improved to $(\Delta/2 + 1/2 - 1/\Delta) \log n + \mathcal{O}(\log \log n)$ [22] and then to optimal $(\Delta/2) \log n + \mathcal{O}(1)$ [5]. In the case of adjacency labelings for general undirected graphs, starting with the classical result presenting labels of size $n/2 + \mathcal{O}(\log n)$ [44], $n/2 + \mathcal{O}(1)$ [10] and $n/2 + 1$ [4] labelings were constructed. Similar sharply optimal labelings are shown for directed graphs, tournaments, bipartite graphs, and oriented graphs. Finally, the first described ancestry labeling schemes for trees was of size $2 \log n$ [37], and then $(3/2) \log n$ [2], $\log n + \mathcal{O}(\log n / \log \log n)$ [50], $\log n + \mathcal{O}(\sqrt{\log n})$ [1], $\log n + 4 \log \log n + \mathcal{O}(1)$ [25], $\log n + 2 \log \log n + \mathcal{O}(1)$ [19] schemes were provided, achieving optimality up to second-order terms.

Related works. The challenge of designing labeling schemes with short labels is related to that of designing succinct data structures, where we want to store the whole information about the input (say, a graph) using very few bits, ideally at most the information theoretical minimum. This is a rather large research area, and we only briefly describe the recent results on succinct data structures for the interval and permutation graphs. Tsakalidis, Wild, and Zama-raev [51] described a structure using only $n \log n + o(n \log n)$ bits (which is optimal) capable of answering many types of queries for permutation graphs. They also introduce the concept of semi-distributed representation, showing that for distances in permutation graphs it is possible to store global array of size $\mathcal{O}(n)$ bits and labels on only $2 \log n$ bits, offering a mixed approach which can overcome $3 \log n$ lower bound for distance labeling. For interval graphs, a structure using $n \log n + \mathcal{O}(n)$ bits (which is again optimal) is known ([3] and [34]).

2 Overview and organisation

In Section 3, we present basic definitions for labeling schemes and our approach to permutation graphs. Then, in Section 4, we build on the methods of Gavaille and Paul [28], as well as Bazzaro and Gavaille [12] for creating distance labelings of interval and permutation graphs. We can represent a permutation graph as a set of points in the plane, where two points (two vertices) are adjacent when one is above and to the left of the other.

The first thing we need to notice when considering distances is the presence of two *boundaries* in such representation. We say that the top boundary is formed by points with empty (containing no other points) top-left (north-west) quadrant, and bottom boundary by points with empty bottom-right (SE) quadrant. Points on the boundaries are especially important – it can be seen that for any pair of points, there is a shortest path between them with all internal points of the path being on boundaries. As a set of boundary points forms a bipartite graph, such shortest path strictly alternates between boundaries.

We can also observe that for a point v not a boundary, there are four boundary points of special interest for it, see Figure 2. These are pairs of extreme points on both boundaries from the points adjacent to v . Any shortest path from v to u with $d(v, u) > 2$ can have as a second point one of these special points for v , and as a penultimate point one of the special points for u . We need to handle distances of 1 and 2 separately, but otherwise, this means it is enough to be able to compute distances between boundary points.

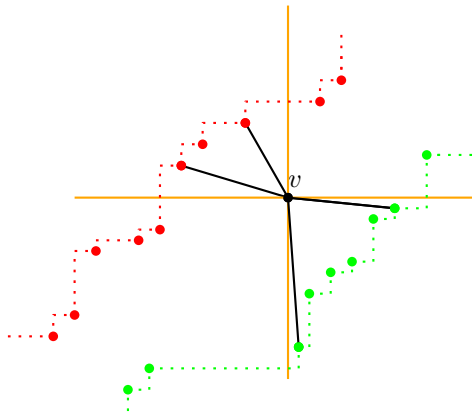


Figure 2: Green points form bottom boundary, red points top boundary. v is not on the boundary, orange lines show its quadrants. For v , there are four points of special interest, extreme neighbours on both boundaries.

If we can build distance labeling for boundary points, and store labels of special points efficiently, we can obtain good distance labelings for permutation graphs. This is possible as boundaries are highly structured, in particular ordered. In [12] authors view two boundaries as two proper interval graphs and deal with them using methods from [28]. An interval graph is proper when no interval is completely contained by another one. Gavaille and Paul first partition vertices of a proper interval into distance *layers*, by distances to the vertex representing the leftmost interval. Let us denote layer number of vertex u by $L(u)$. It can be seen that for any two vertices u, v in interval graph we have either $d(u, v) = |L(u) - L(v)|$ or $d(u, v) = |L(u) - L(v)| + 1$. Then the following lemma is used [28]:

Lemma 2. *There exists a total ordering λ of vertices of proper interval graph such that given $\lambda(v), \lambda(u)$ and layer numbers $L(u) < L(v)$ for two vertices u, v , we have $d(u, v) = L(v) - L(u)$ if and only if $\lambda(u) > \lambda(v)$.*

In other words, we can assign to each vertex v just two numbers $L(v), \lambda(v)$, and then still be able to determine all exact distances. Going back to permutation graphs, when we view two boundaries as proper interval graphs, it is possible to obtain straightforward distance labeling for permutation graphs using $20 \log n$ bits, where the big constant is due to storing many distance labels for interval graphs completely independently. Then authors are able to reduce the size of labels to $9 \log n$ bits, after eliminating many redundancies in the stored sub-labels.

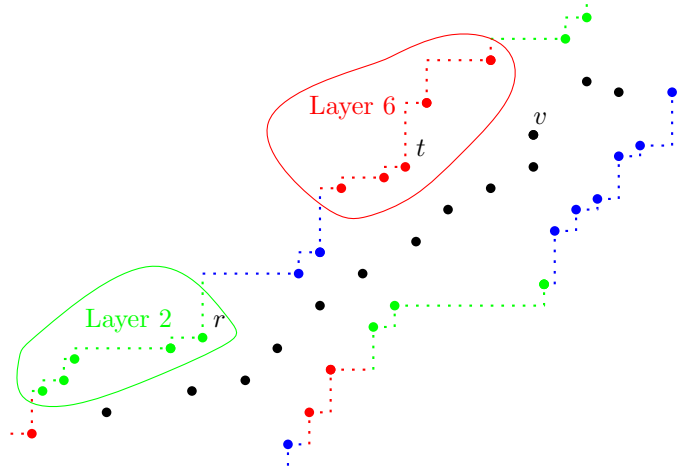


Figure 3: Boundary points partitioned into layers. r, t are on the top boundary, but in layers 2 and 6. For any two points in layers a and b , the distance between them is always either $|a - b|$ or $|a - b| + 2$; here, $d(r, t) = 4$. v is not on the boundary, and any such point can be adjacent to points from at most three different layers.

In this paper, we show that working with both boundaries at once can yield better results. To do this, we modify the methods of Bazzaro and Gavaille and then carefully remove even more redundancies. First, we partition points on both boundaries into layers, defined by distances from some initial point, see Figure 3. As we use distances from a single point to define layers, the distance between any two boundary points is a difference of their layer numbers, or this value increased by two. It can be shown that again some λ ordering can be used, and storing it takes around $\log n$ bits for each boundary point.

As a single point is adjacent to at most three layers, layer numbers of four special points are easy to store, and we could achieve labeling of length $(2 + 1 + 4) \log n + \mathcal{O}(1) = 7 \log n + \mathcal{O}(1)$ by storing for each point respectively its 2D coordinates, layer numbers of neighbours and four

times λ values for extreme neighbours, in order to compute distances between boundary points. This can be reduced to $5 \log n$ by dealing with distances 1 and 2 more carefully, allowing us to not store point coordinates explicitly. All of the above is described in Section 4.

After additional analysis and reductions laid out in Section 5, we can decrease the size to $3 \log n$. This is since, roughly speaking, one can collapse information stored for two pairs of extreme boundary neighbours into just two numbers, due to useful graph and layers properties. More precisely, we can observe that we store excessive information about the set of four extreme neighbours. For vertex v , two extreme right points on both boundaries are used to reach points to the right of v , and extreme left are used to reach points to the left. But we do not need the exact distance between points to the left of v and right extreme points, thus we have some possibility to adjust the stored λ values. Particularly, the main case is when λ value of the right extreme point on the bottom boundary is smaller than λ value of the left extreme point on the top boundary; it turns out that these two values can be equalised and stored as some single value in between the original values. The second pair of extreme points can be dealt with in a similar manner, and then we need to ensure that all of this did not interfere with the correctness of deciding about distances 1 and 2, which are different cases than all distances larger than 2.

3 Preliminaries

Permutation graphs. Permutation graph G_π is a graph with vertices representing elements of permutation π , where there is an edge between two vertices if and only if the elements they represent form an inversion in the permutation. See Figure 1. In [41] McConnell and Spinrad show that it is possible to test in linear time whether a given graph is a permutation graph, and also construct the corresponding permutation.

We will use a geometric (or 'grid') representation of permutation graph G_π on n vertices as a set of points with coordinates in $[1, n]$, with point $(i, \pi^{-1}(i))$ for each $i \in [1, n]$. Considering a point p , we always denote its coordinates by $p = (p_x, p_y)$. Top-left quadrant of point p , TL_p , is a subset of points $\{v : v_x < p_x \wedge v_y > p_y\}$ from the graph. Similarly, we have TR_p (top-right), BL_p (bottom-left) and BR_p (bottom-right) quadrants. Two points are adjacent in the graph iff one is in TL or BR quadrant of the other. See Figure 4. We have transitivity in a sense that if $w \in BR_v$ and $u \in BR_w$, then $u \in BR_v$; similarly for other quadrants. By distance $d(u, v)$ between two points we will mean distance in the graph.

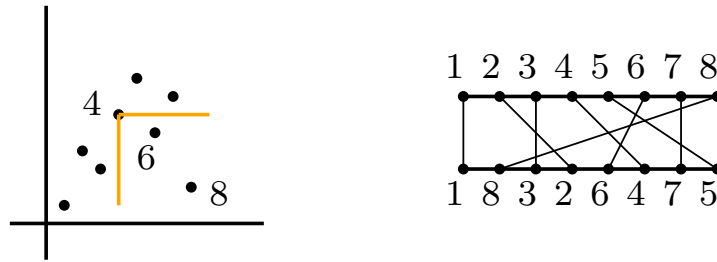


Figure 4: Geometric representation of the graph from Figure 1, so permutation $[1,8,3,2,6,4,7,5]$. Point $(4,6)$ is adjacent to $(6,5)$ and $(8,2)$, as these two points are in its bottom-right quadrant, while top-left quadrant of $(4,6)$ is empty.

We will assume the given permutation graph is connected. There are standard ways to enhance labelings to disconnected graphs by adding at most $\mathcal{O}(\log \log n)$ bits to the labels, and we will describe how it can be done after the main theorem. We note that for connected graphs of size at least two, no point could be on both boundaries, as it would be isolated otherwise.

Labeling schemes. Let \mathcal{G} be a family of graphs. A distance labeling scheme for \mathcal{G} consists of an encoder and a decoder. The encoder takes a graph $G \in \mathcal{G}$ and assigns a label (binary string) $\ell(u)$ to every vertex $u \in G$. The decoder receives labels $\ell(u)$ and $\ell(w)$, such that $u, w \in G$ for some $G \in \mathcal{G}$ and $u \neq w$, and should report the exact distance $d(u, w)$ between u and w in G . The decoder is not aware of G and only knows that u and w come from the same graph belonging to \mathcal{G} . We are interested in minimizing the maximum length of a label, that is, $\max_{G \in \mathcal{G}} \max_{u \in G} |\ell(u)|$.

Organization of the labels. The final labels will consist of a constant number of parts. We can store at the beginning of each label a constant number of pointers to the beginning of each of those parts. As the total length of a label will be $\mathcal{O}(\log n)$, pointers add only $\mathcal{O}(\log \log n)$ bits to the labels.

4 Scheme of Size $5 \log n$

In this section, we describe how to use boundaries to design distance labeling of size $7 \log n + \mathcal{O}(1)$, and then how to refine it to reach $5 \log n + \mathcal{O}(1)$.

4.1 Properties of Boundaries

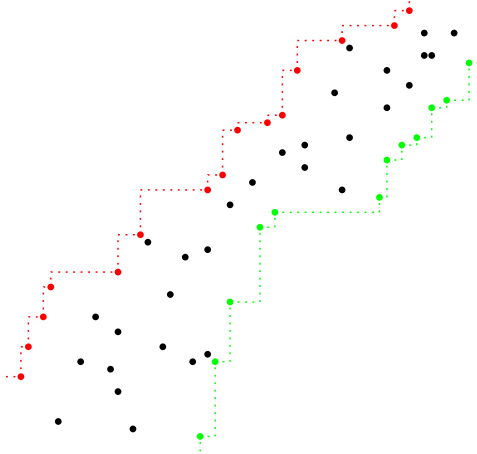


Figure 5: Green points are on the bottom boundary, red points are on the top boundary.

For a set of points S , we have its top boundary defined as a subset of points from S which top-left quadrants are empty, and bottom boundary as a subset of points which bottom-right quadrants are empty. See Figure 5. Observe that points on boundaries are ordered, that is, for u and v on the same boundary, either $u_x > v_x$ and $u_y > v_y$, or $u_x < v_x$ and $u_y < v_y$. We use $<$ to note this relation on boundary points.

Boundaries are particularly useful when considering distances between points:

Property 3. *For any two points u, v at distance d , there is a path $P = (u = q_0, q_1, q_2, \dots, q_d = v)$ of length d such that all points except possibly u, v are on alternating boundaries.*

Proof. **TOPROVE 0**

□

We partition all points on the boundaries into layers, in the following way. The layer number 0 consists of a single left-most point p_0 in the whole set S . Note that p_0 is on the top boundary.

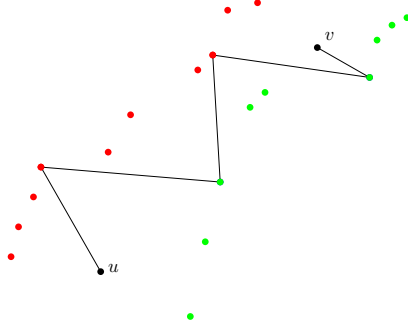


Figure 6: Path between two points alternating between top and bottom boundaries.

Then, a boundary point is in a layer number i if its distance to p_0 is i . By $L(u)$ we denote the layer number of u . See Figure 7, we will soon see that indeed layers are always nicely structured, as pictured. Observe that in even layers, we have only points from the top boundary, and in odd layers only from the bottom boundary, as points on a single boundary are non-adjacent. Thus, points in a single layer are ordered, by both coordinates.

To determine the distance between boundary points, we use a method similar to the one from the paper of Gavaille and Paul [28], precisely Theorem 3.8. This is also connected to what Bazzaro and Gavaille [12] do in their work, but not identical, as they use bottom and top boundaries separately, as two mostly independent interval graphs.

Lemma 4. *There exists a total ordering λ of boundary points such that given $\lambda(v), \lambda(u)$ and layer numbers $L(u) < L(v)$ for two boundary points u, v , we have $d(u, v) = L(v) - L(u)$ if and only if $\lambda(u) > \lambda(v)$.*

Proof. **TOPROVE 1** □

By Lemma 4, we are able to detect when quick paths exist, and to complete knowledge about distances between boundary points we observe the following:

Property 5. *For any boundary points u, v with $L(v) \geq L(u)$, $d(u, v)$ is equal to either $L(v) - L(u)$ or $L(v) - L(u) + 2$.*

Proof. **TOPROVE 2** □

To simplify our proofs, we will add some points to the original set. For each point v on the bottom boundary and from the original input, we add point $(v_x + \epsilon, v_y - \epsilon)$. It is easy to see that such a point lies on the bottom boundary, adding it does not change distances between any existing points, and it removes v from the bottom boundary. Then we change numeration to integer numbers again, increasing the range of numbers by some constant factor. Similarly, for any original point v on top boundary, we add $(v_x - \epsilon, v_y + \epsilon)$. What we achieve is that after this change no point from the original input lies on the boundary, which reduces the number of cases one needs to consider when assigning labels (only to the original points).

Now let us focus on any point v not on the boundary. Assume v is adjacent to some points from layers i and j , $j > i$. It cannot be that $j > i + 2$, by definition of layers as distances from p_0 . Thus, v is adjacent to points from at most three (consecutive) layers. We note that v is adjacent to a consecutive segment of ordered points from any layer i . Let us denote by $\text{Bfirst}(v)$ and $\text{Blast}(v)$ the first and last points on the bottom boundary adjacent to v and by $\text{Tfirst}(v)$ and $\text{Tlast}(v)$ the first and last points on the top boundary adjacent to v . Consult Figure 8.

We can make easy observation on points at distance two:

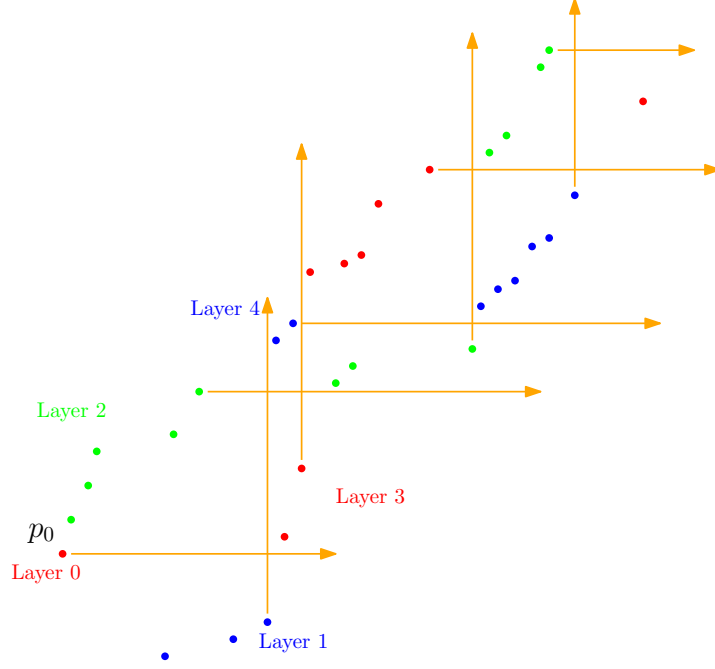


Figure 7: Layers on boundary points defined as distances from the leftmost point p_0 . Orange lines represent which points are adjacent to (being in BR or TL) the last point in the layer.

Property 6. For any two points u, v , $d(u, v) \leq 2$ is equivalent to $[\text{Bfirst}(u), \text{Blast}(u)] \cap [\text{Bfirst}(v), \text{Blast}(v)] \neq \emptyset$ or $[\text{Tfirst}(u), \text{Tlast}(u)] \cap [\text{Tfirst}(v), \text{Tlast}(v)] \neq \emptyset$.

This is since by Property 3, we must have a path between u, v at distance two going through a single point on the boundary. In the case of $d(u, v) = 1$, assume without loss of generality that $u \in TL_v$, then $[\text{Tfirst}(u), \text{Tlast}(u)] \subseteq [\text{Tfirst}(v), \text{Tlast}(v)]$, and ranges are never empty.

Considering points at a distance of at least three, we have the following:

Lemma 7. For any two non-boundary points u, v with $d(u, v) > 2$ and $u_x < v_x$, there is a shortest path from u to v with the second point being either $\text{Blast}(u)$ or $\text{Tlast}(u)$ and the penultimate point being either $\text{Bfirst}(v)$ or $\text{Tfirst}(v)$.

Proof. **TOPROVE 3** □

We established ways to determine the distance between any points using distances between specific boundary points. Additionally, observe that all conditions from Lemma 4 and Property 6 can be checked using just λ values and layer numbers. That is, for u, v on the same boundary we have $u \leq v$ iff $(L(u), \lambda(u)) \leq_{lex} (L(v), \lambda(v))$.

At this stage, we could create labels of length $7 \log n + \mathcal{O}(1)$, by storing for each point v coordinates v_x, v_y , and for all points of interest $\text{Bfirst}(v), \text{Blast}(v), \text{Tfirst}(v), \text{Tlast}(v)$, their $\lambda(\cdot)$ and $L(\cdot)$ values. As a point is adjacent to at most three layers, all four layer numbers can be stored on just $\log n + \mathcal{O}(1)$ bits. Coordinates allow us to check for distance 1, distance two is checked by using Property 6, and larger distances by Property 5, Lemma 4 and 7.

4.2 Auxiliary points

To better manage detecting points at distance 1 without explicitly storing coordinates, we will add, for each point in the set S , four additional artificial points, two on each boundary. Consider

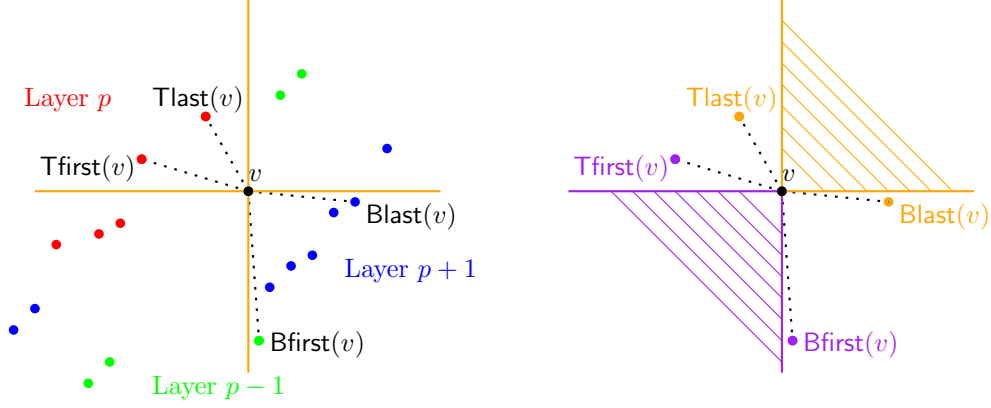


Figure 8: On the left: boundary points adjacent to v . Orange lines represent TL_v and BR_v . v is adjacent to the two last points in red layer p , one last point in green layer $p-1$, and the first five points in layer $p+1$. On the right: Two orange extreme neighbours are needed for paths to points in marked TR_v , and two purple ones for BL_v .

v from the initial set and its bottom-right quadrant BR_v . We add two points $v_b = (v_x - \epsilon, Bfirst(v)_y - \epsilon)$ and $v_{b'} = (Blast(v)_x + \epsilon, v_y + \epsilon)$ to the set S of points. See Figure 10. Then, we change the numeration of coordinates so that we still use the permutation of natural numbers up to $|S|$. This is repeated for all the initial points. First, we check that this addition did not disturb the properties of the points too much:

Lemma 8. *All added points are on the bottom boundary. Moreover, for any two points $u, w \in S$, $d(u, w)$ remains the same.*

Proof. TOPROVE 4 □

Similarly, for each point in the initial set, we add two points on the upper boundary. That is, consider v and TL_v . We add two points $v_t = (Tfirst(v)_x - \epsilon, v_y - \epsilon)$ and $v_{t'} = (v_x + \epsilon, Tlast(v)_y + \epsilon)$ to the set S of points. Then, we again change the numeration of coordinates. This is symmetric and has the same properties.

After adding four auxiliary points for all initial points, we have the desired property:

Lemma 9. *For any two points v, w from the initial set, $w \in BR_v$ is equivalent to $Bfirst(v) < Bfirst(w) \leq Blast(w) < Blast(v)$. Moreover, $w \in TL_v$ is equivalent to $Tfirst(v) < Tfirst(w) \leq Tlast(w) < Tlast(v)$.*

Proof. TOPROVE 5 □

The above lemma is useful when testing for adjacency of points – we do not need explicit coordinates to check it. Now, we are able to create labels of length $5 \log n + \mathcal{O}(1)$, as there is no longer a need to store coordinates, thus just four λ values, and additionally layer numbers using only $\log n + \mathcal{O}(1)$ bits. We can still improve on this, getting our final result in the next Section.

5 Final Scheme of Size $3 \log n$

In this Section, the final improvement to label sizes is achieved, by collapsing two *pairs* of λ values into just two values, with an additional constant number of bits.

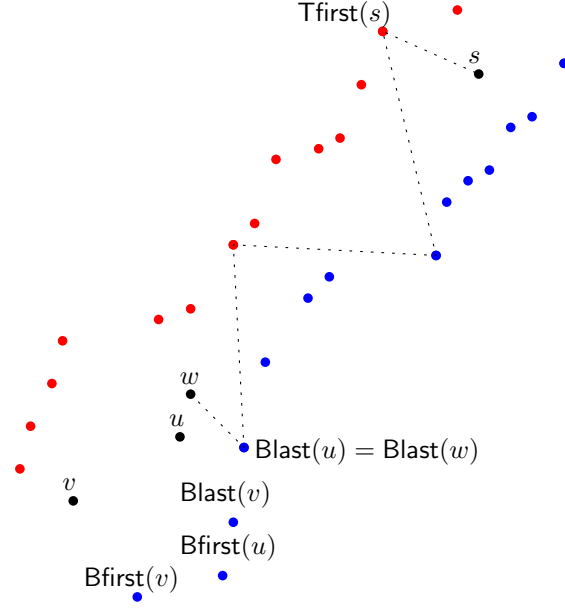


Figure 9: We have $d(u, v) = 2$, even though all $Bfirst(v)$, $Blast(v)$, $Bfirst(u)$, $Blast(u)$ are different, but their ranges do intersect. Moreover, there is a shortest path from w to s with the second point being $Blast(w)$ and the penultimate point being $Tfirst(s)$.

Lemma 10. *We can store for each input point v two integers $v_{x'}, v_{y'}$ with values in $\mathcal{O}(n)$, bit values v_{binf}, v_{tinf} , and layer numbers $L(Bfirst(v)), L(Blast(v)), L(Tfirst(v)), L(Tlast(v))$ smaller than n , such that distance queries for pairs of points can be answered using these values only.*

Proof. **TOPROVE 6** □

To conclude, we have that $\ell(v)$ consists of the following parts:

1. $L(Bfirst(v)), L(Blast(v)), L(Tfirst(v))$ and $L(Tlast(v))$, all stored on total $\log n + \mathcal{O}(1)$ bits due to differences between these values being at most 2.
2. Bit v_{binf} and value $v_{y'}$, on $\log n + \mathcal{O}(1)$ bits.
3. Bit v_{tinf} and value $v_{x'}$, like above.

We increased the number of vertices in the graph by a constant factor, so the final length is $3 \log n + \mathcal{O}(1)$ bits. Decoding can be done in constant time.

5.1 Disconnected graphs

Here we describe how to modify distance labeling for connected graphs into distance labeling for general graphs, by adding at most $\mathcal{O}(\log \log n)$ bits to the labels. This is a standard simple approach, present in some related works. We sort connected components of the graph by decreasing size, say we have C_1, C_2, \dots , then proceed with creating distance labeling for each individual connected component. The final label is the number of connected component of a vertex and then its label for the distance created in the component. If $v \in C_i$, we encode i on $\log i + \mathcal{O}(\log \log n)$ bits. We have $|C_i| \leq n/i$, so the final label size is at most $\log(n/i) + \mathcal{O}(\log \log n) + \log i = \log n + \mathcal{O}(\log \log n)$, as claimed.

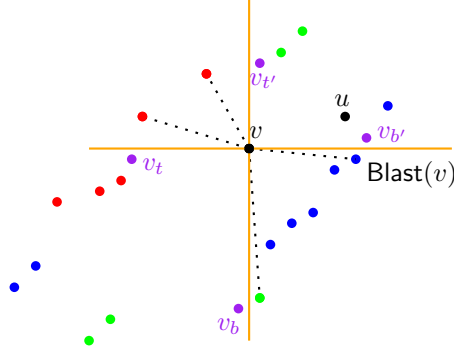


Figure 10: Four points added for a point v : v_b and $v_{b'}$ on bottom boundary, v_t and $v_{t'}$ on top. We have a property that whenever point u is adjacent to $v_{b'}$, it was already adjacent to $\text{Blast}(v)$.

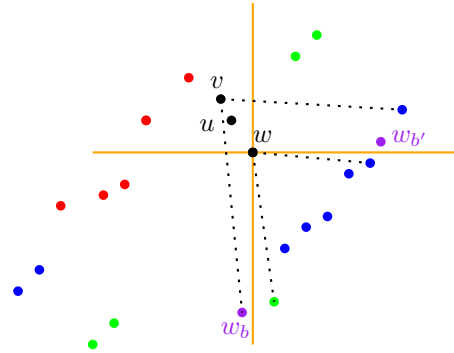


Figure 11: We have $w \in \text{BR}_v$, and $\text{Bfirst}(v) < \text{Bfirst}(w) < \text{Blast}(w) < \text{Blast}(v)$. Meanwhile, $u \notin \text{BR}_w$, as $\text{Blast}(u) = w_{b'} > \text{Blast}(w)$. Auxiliary points which would be added for u, v are not shown to avoid clutter.

6 Conclusion

Improving upon the previous results, we have described a distance labeling scheme for permutation graphs matching existing lower bound up to an additive second-order $\mathcal{O}(\log \log n)$ term. This also improves constants in distance labeling for circular permutation graphs, as described in [12]. Namely, one can construct distance labeling of size $6 \log n + \mathcal{O}(\log \log n)$ for such graphs. We leave as an open question determining the complexity of distance labeling for circular permutation graphs, and finding more interesting generalisations.

References

- [1] Serge Abiteboul, Stephen Alstrup, Haim Kaplan, Tova Milo, and Theis Rauhe. Compact labeling scheme for ancestor queries. *SIAM J. Comput.*, 35(6):1295–1309, 2006.
- [2] Serge Abiteboul, Haim Kaplan, and Tova Milo. Compact labeling schemes for ancestor queries. In *12th SODA*, pages 547–556, 2001.
- [3] Hüseyin Acan, Sankardeep Chakraborty, Seungbum Jo, and Srinivasa Rao Satti. Succinct encodings for families of interval graphs. *Algorithmica*, 83(3):776–794, 2021.

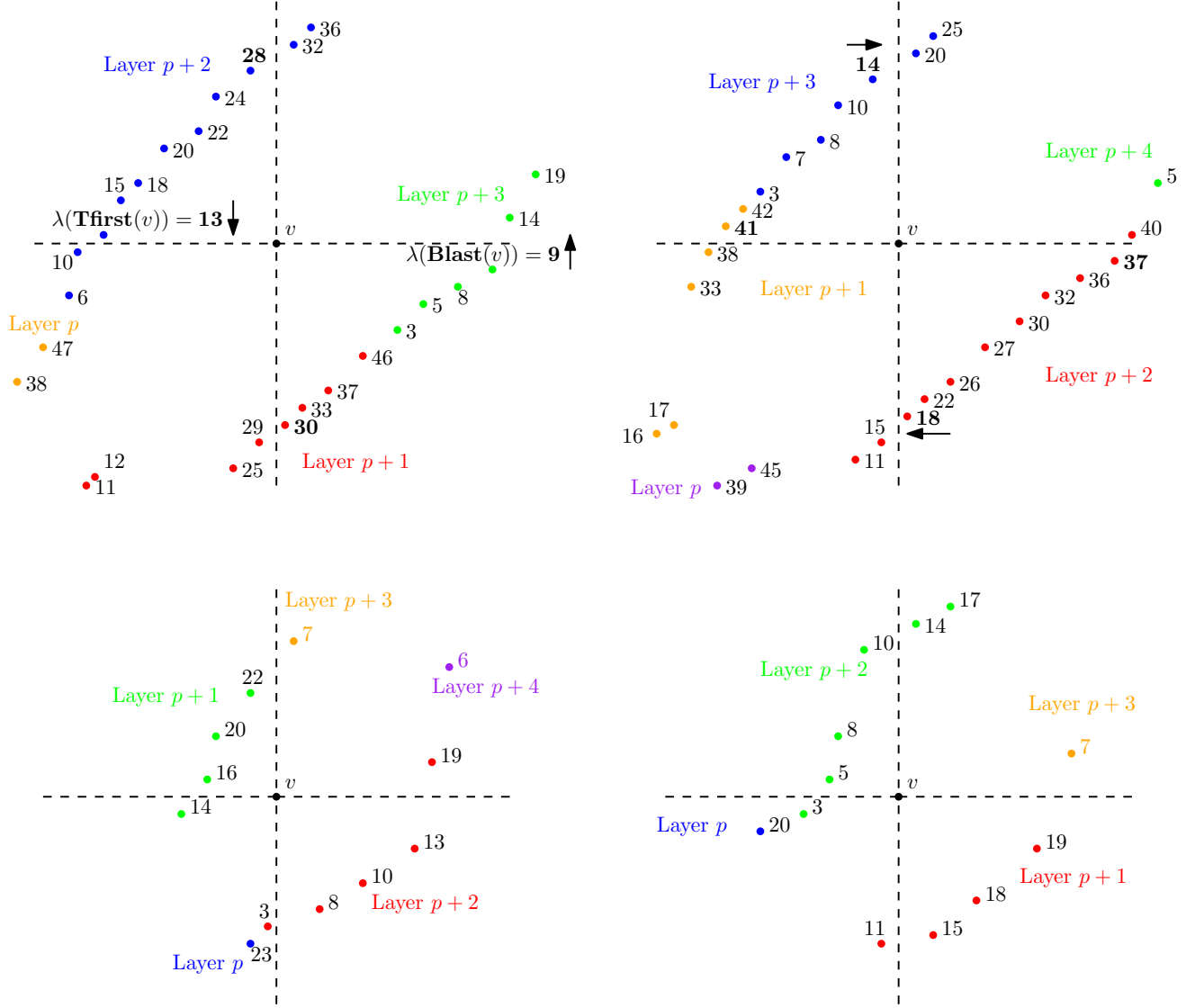


Figure 12: In the first case, label of v could store $\lambda(\text{Bfirst}(v))$, $\lambda(\text{Blast}(v))$, $\lambda(\text{Tfirst}(v))$, $\lambda(\text{Tlast}(v))$ values, which would be 30, 9, 13, and 28. But instead, we can use in its label just two values, $v_{y'} = 13 - \epsilon$ and $v_{x'} = 30 - \epsilon$. In the second case, point v could store values 18, 37, 41, 14. Instead, we can use $v_{y'} = 40 - \epsilon$ and $v_{x'} = 18 - \epsilon$. In the last case, we have point v adjacent to only two layers, and the lower one is on the bottom boundary. v could store values 15, 19, 5, and 10. Instead, we can use in its label just two numbers, $v_{y'} = 5$, $v_{x'} = 14 - \epsilon$, with two more bits indicating this case. We use the fact that a point with λ value of 19 is necessarily the last one in its layer and we won't need its exact value.

- [4] Noga Alon. Asymptotically optimal induced universal graphs. *Geometric and Functional Analysis*, 27, 02 2017.
- [5] Noga Alon and Rajko Nenadov. Optimal induced universal graphs for bounded-degree graphs. In *28th SODA*, pages 1149–1157, 2017.
- [6] Stephen Alstrup, Søren Dahlgaard, and Mathias Bæk Tejs Knudsen. Optimal induced universal graphs and adjacency labeling for trees. In *56th FOCS*, pages 1311–1326, 2015.

- [7] Stephen Alstrup, Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Ely Porat. Sublinear distance labeling. In *24th ESA*, volume 57, pages 5:1–5:15, 2016.
- [8] Stephen Alstrup, Cyril Gavoille, Esben Bistrup Halvorsen, and Holger Petersen. Simpler, faster and shorter labels for distances in graphs. In *27th SODA*, pages 338–350, 2016.
- [9] Stephen Alstrup, Inge Li Gørtz, Esben Bistrup Halvorsen, and Ely Porat. Distance labeling schemes for trees. In *43rd ICALP*, pages 132:1–132:16, 2016.
- [10] Stephen Alstrup, Haim Kaplan, Mikkel Thorup, and Uri Zwick. Adjacency labeling schemes and induced-universal graphs. In *47th STOC*, pages 625–634, 2015.
- [11] K. A. Baker, Peter C. Fishburn, and Fred S. Roberts. Partial orders of dimension 2. *Networks*, 2(1):11–28, 1972.
- [12] Fabrice Bazzaro and Cyril Gavoille. Distance labeling for permutation graphs. *Electron. Notes Discret. Math.*, 22:461–467, 2005.
- [13] Marthe Bonamy, Louis Esperet, Carla Groenland, and Alex D. Scott. Optimal labelling schemes for adjacency, comparability, and reachability. In *53rd STOC*, pages 1109–1117, 2021.
- [14] Marthe Bonamy, Cyril Gavoille, and Michał Pilipczuk. Shorter labeling schemes for planar graphs. In *31st SODA*, pages 446–462, 2020.
- [15] Édouard Bonnet, Julien Duron, John Sylvester, Viktor Zamaraev, and Maksim Zhukovskii. Tight bounds on adjacency labels for monotone graph classes. In *51st ICALP*, pages 31:1–31:20.
- [16] Steve Butler. Induced-universal graphs for graphs with bounded maximum degree. *Graphs and Combinatorics*, 25:461–468, 2009.
- [17] H.S. Chao, F.R. Hsu, and R.C.T. Lee. An optimal algorithm for finding the minimum cardinality dominating set on permutation graphs. *Discret. Appl. Math.*, 102(3):159–173, 2000.
- [18] Charles J. Colbourn. On testing isomorphism of permutation graphs. *Networks*, 11(1):13–21, 1981.
- [19] Søren Dahlgaard, Mathias Bæk Tejs Knudsen, and Noy Rotbart. A simple and optimal ancestry labeling scheme for trees. In *42nd ICALP*, pages 564–574, 2015.
- [20] Vida Dujmovic, Louis Esperet, Cyril Gavoille, Gwenaël Joret, Piotr Micek, and Pat Morin. Adjacency labelling for planar graphs (and beyond). In *61st FOCS*, pages 577–588, 2020.
- [21] David Ellis. Intersection problems in extremal combinatorics: theorems, techniques and questions old and new. *Surveys in Combinatorics*, pages 115–173, 2022.
- [22] Louis Esperet, Arnaud Labourel, and Pascal Ochem. On induced-universal graphs for the class of bounded-degree graphs. *Inf. Process. Lett.*, 108(5):255–260, 2008.
- [23] Shimon Even, Amir Pnueli, and Abraham Lempel. Permutation graphs and transitive graphs. *J. ACM*, 19(3):400–410, 1972.
- [24] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In *28th ICALP*, pages 757–772, 2001.

- [25] Pierre Fraigniaud and Amos Korman. An optimal ancestry scheme and small universal posets. In *42th STOC*, pages 611–620, 2010.
- [26] Ofer Freedman, Paweł Gawrychowski, Patrick K. Nicholson, and Oren Weimann. Optimal distance labeling schemes for trees. In *36th PODC*, pages 185–194, 2017.
- [27] Cyril Gavoille and Arnaud Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In *15th ESA*, pages 582–593, 2007.
- [28] Cyril Gavoille and Christophe Paul. Optimal distance labeling for interval graphs and related graph families. *SIAM J. Discret. Math.*, 22(3):1239–1258, 2008.
- [29] Paweł Gawrychowski and Wojciech Janczewski. Simpler adjacency labeling for planar graphs with b-trees. In *5th SOSA*, pages 24–36, 2022.
- [30] Paweł Gawrychowski, Wojciech Janczewski, and Jakub Lopuszanski. Shorter labels for routing in trees. In *32nd SODA*, pages 2174–2193, 2021.
- [31] Paweł Gawrychowski, Fabian Kuhn, Jakub Lopuszanski, Konstantinos Panagiotou, and Pascal Su. Labeling schemes for nearest common ancestors through minor-universal trees. In *29th SODA*, pages 2604–2619, 2018.
- [32] Paweł Gawrychowski and Przemysław Uznanski. Better distance labeling for unweighted planar graphs. *Algorithmica*, 85(6):1805–1823, 2023.
- [33] Hamed Hatami and Pooya Hatami. The implicit graph conjecture is false. In *63rd FOCS*, pages 1134–1137, 2022.
- [34] Meng He, J. Ian Munro, Yakov Nekrich, Sebastian Wild, and Kaiyu Wu. Distance oracles for interval graphs via breadth-first rank/select in succinct trees. In *31st ISAAC*, pages 25:1–25:18, 2020.
- [35] Meng He and Kaiyu Wu. Closing the gap: Minimum space optimal time distance labeling scheme for interval graphs. In *35th CPM*, pages 17:1–17:18, 2024.
- [36] Petr Hliněný and Jan Kratochvíl. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discret. Math.*, 229(1-3):101–124, 2001.
- [37] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM J. Discret. Math.*, 5(4):596–603, 1992.
- [38] Michal Katz, Nir A. Katz, and David Peleg. Distance labeling schemes for well-separated graph classes. *Discret. Appl. Math.*, 145(3):384–402, 2005.
- [39] Amos Korman. Labeling schemes for vertex connectivity. *ACM Trans. Algorithms*, 6(2):39:1–39:10, 2010.
- [40] Amos Korman, David Peleg, and Yoav Rodeh. Constructing labeling schemes through universal matrices. *Algorithmica*, 57(4):641–652, 2010.
- [41] Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discret. Math.*, 201(1-3):189–241, 1999.
- [42] Terry A. McKee and F. R. McMorris. *Topics in Intersection Graph Theory*. Society for Industrial and Applied Mathematics, 1999.

- [43] Rolf H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In *Graphs and Order: The Role of Graphs in the Theory of Ordered Sets and Its Applications*, pages 41–101. Springer Netherlands, Dordrecht, 1985.
- [44] J. W. Moon. On minimal n -universal graphs. *Proceedings of the Glasgow Mathematical Association*, 7(1):32–33, 1965.
- [45] John H. Muller. Local structure in graph classes. *PhD thesis, School of Information and Computer Science*, 1988.
- [46] Madhumangal Pal. Intersection graphs: An introduction. *Annals of Pure and Applied Mathematics*, 4(1):43–91, 2013.
- [47] David Peleg. Proximity-preserving labeling schemes. *Journal of Graph Theory - JGT*, 33:167–176, 03 2000.
- [48] David Peleg. Informative labeling schemes for graphs. *Theor. Comput. Sci.*, 340(3):577–593, 2005.
- [49] Arseny M. Shur and Mikhail Rubinchik. Distance labeling for families of cycles. In *49th SOFSEM*, pages 471–484, 2024.
- [50] Mikkel Thorup and Uri Zwick. Compact routing schemes. In *13th SPAA*, pages 1–10, 2001.
- [51] Konstantinos Tsakalidis, Sebastian Wild, and Viktor Zamaraev. Succinct permutation graphs. *Algorithmica*, 85(2):509–543, 2023.