



Optimal Distance Labeling for Permutation Graphs*

Paweł Gawrychowski ✉ 

Institute of Computer Science, University of Wrocław, Poland

Wojciech Janczewski ✉ 

Institute of Computer Science, University of Wrocław, Poland

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 $\mathcal{O}(\log^2 n)^1$ bits. Later, Bazzaro and Gavaille [Discret. Math. 309(11)] obtained an asymptotically optimal bound 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 $3 \log n + \mathcal{O}(1)$ bits.

2012 ACM Subject Classification Theory of computation \rightarrow Distributed algorithms; Theory of computation \rightarrow Graph algorithms analysis

Keywords and phrases informative labeling, permutation graph, distance labeling

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.86

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: <https://arxiv.org/abs/2407.12147>

1 Introduction

A 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 when 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.

* Partially supported by the Polish National Science Centre grant number 2023/51/B/ST6/01505.

¹ In this paper, all logarithms are in base 2.



© Paweł Gawrychowski and Wojciech Janczewski;
licensed under Creative Commons License CC-BY 4.0

52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025).

Editors: Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis

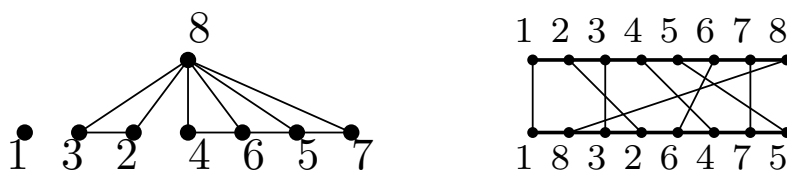
Article No. 86; pp. 86:1–86:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany





■ **Figure 1** Permutation graph described by $\pi = 18326475$. To the right is its representation as an intersection graph of line segments whose endpoints lie on two parallel lines.

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 [10]. 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 [9], graphs of bounded degree [22], trees [5], 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 informative 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 [7] 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 [6] 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 Uznański [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 [8]. 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 Gavaille and Paul to $5 \log n$ distance 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 a tight $3 \log n + \log \log n + \mathcal{O}(1)$ distance labeling for interval graphs. For connected permutation graphs, Bazzaro and Gavaille in [11, 12] showed a distance labeling scheme of size $9 \log n + \mathcal{O}(1)$, 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 graphs, by showing the following theorem.

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

We note that if the graph is not necessarily connected, a black-box modification allows for extending our scheme at the expense of increasing the size of each label to $3 \log n + \mathcal{O}(\log \log n)$.

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 $1/2 \log^2 n$ [8] and then $1/4 \log^2 n + o(\log^2 n)$ [26], optimal up to second-order term. Adjacency labeling for planar graphs 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], 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 $(\Delta/2) \log n + \mathcal{O}(1)$ [4]. 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], $n/2 + \mathcal{O}(1)$ [9] and $n/2 + 1$ [3] labelings were constructed. Similar sharply optimal labelings were shown also 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 $(3/2) \log n$ [1], $\log n + \mathcal{O}(\log n / \log \log n)$ [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 Zamaraev [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 a global array of size $\mathcal{O}(n)$ bits and labels consisting of 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 ([2] and [34]).

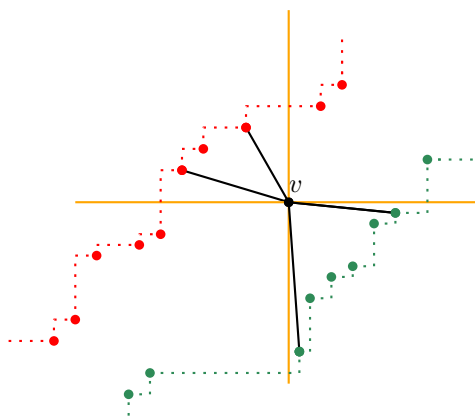
2 Overview and Organisation

In Section 3, we present basic definitions concerning labeling schemes and permutation graphs. Then, in Section 4, we build on the methods of Gavaille and Paul [28], as well as of Bazzaro and Gavaille [11, 12], to design a distance labeling scheme for connected permutation graphs on n vertices with each label consisting of $5 \log n + \mathcal{O}(1)$ bits. Finally, we briefly sketch how to further improve on this construction and obtain labels consisting of $3 \log n + \mathcal{O}(1)$ bits in Section 5, with the details described in the full version of this paper.

From now on, we represent the 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 a 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 (south-east) quadrant. Points on the boundaries are especially important, as 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 the boundaries. As the set of all boundary points forms a bipartite graph, such shortest path strictly alternates between the boundaries.

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

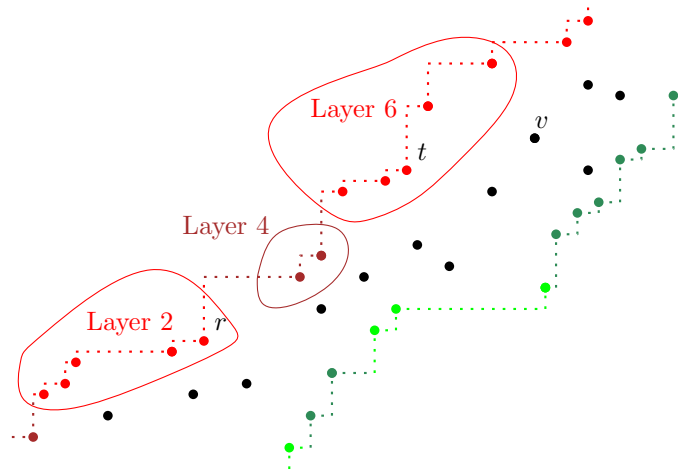


■ **Figure 2** Green points form the bottom boundary, red points form the top boundary. v is not on the boundary, orange lines show its quadrants. For v , there are four points of special interest, extreme neighbours on both boundaries.

If we can build a distance labeling scheme only for the boundary points, and store the labels of the special points efficiently, we can obtain a good distance labeling scheme for the whole graph. This is possible as the boundaries are highly structured, in particular ordered. [12] views the two boundaries as two proper interval graphs, and deals with them using the methods from [28]. An interval graph is proper when no interval is completely contained in another one. [28] first partition vertices of a proper interval graph into distance *layers*, by distances to the vertex representing the leftmost interval. Let us denote the layer number of vertex u by $L(u)$. It can be seen that for any two vertices u, v in an 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 the exact distances. Going back to permutation graphs, when we view two boundaries as proper interval graphs, it is possible to obtain a 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 the authors were able to reduce 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 this end, 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 difference increased by two. It can be shown that again some ordering λ 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 packed into $\log n + \mathcal{O}(1)$ bits, and four λ values for extreme neighbours, in order to compute the distances between the boundary points. This can be reduced to $5 \log n + \mathcal{O}(1)$ 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 the full version of the paper, we can improve the length to $3 \log n + \mathcal{O}(1)$. This is since, roughly speaking, one can collapse the information stored for two pairs of extreme boundary neighbours into just two numbers, due to some properties of the graph and the layers. More precisely, we can observe that we store excessive information about the set of four extreme neighbours. For a 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 the 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 (informally speaking) made equal 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 such adjustments did not interfere with the correctness of deciding about distances 1 and 2, which behave differently than distances larger than 2.

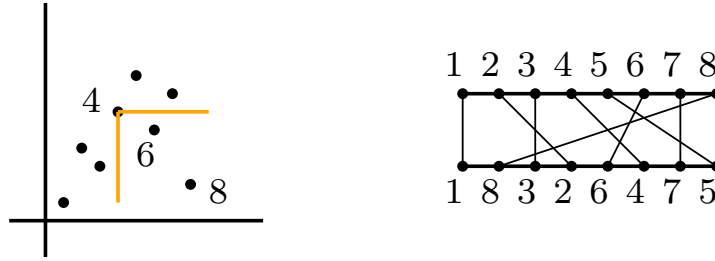
3 Preliminaries

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 a 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.

We will assume that the given permutation graph is connected. It is standard to extend our scheme to permutation graphs that are not necessarily connected at the expense of increasing the size of each label by $\mathcal{O}(\log \log n)$, for completeness we describe how at the end of the paper. We note that for connected graphs of size at least two, no point could be on both boundaries, as it would be isolated.

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 $u \in V(G)$. The decoder receives labels $\ell(u)$ and $\ell(w)$, such that $u, w \in V(G)$ for some $G \in \mathcal{G}$ and $u \neq w$, and should report the exact distance $d(u, w)$



■ **Figure 4** Geometric representation of the graph from Figure 1, so the permutation is $[1,8,3,2,6,4,7,5]$. 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.

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} . Note that the output of the decoder given labels of vertices from two different graphs can be arbitrary. We are interested in minimizing the maximum length of a label, that is, $\max_{G \in \mathcal{G}} \max_{u \in V(G)} |\ell(u)|$, as a function of $n = |V(G)|$.

Organization of the labels. The final labels will consist of a constant number of parts (in the final version of our scheme, there are three such parts), each of length $\log n + \mathcal{O}(1)$. In fact, from the construction it will be clear that we guarantee that the length of each part is the same, as each of them is an integer number with possibly a constant number of additional bits. This allows us to simply concatenate all the parts together in some predetermined order. Then, given the concatenation, extracting all the parts is straightforward.

4 Scheme of Size $5 \log n + \mathcal{O}(1)$

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

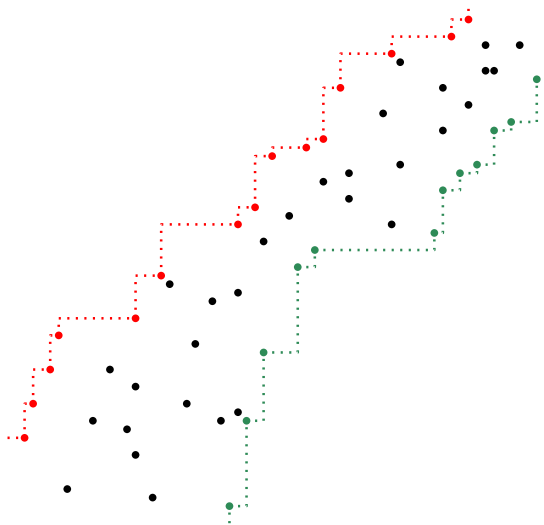
4.1 Properties of Boundaries

For a set of points S , we have its top boundary defined as a subset of points from S whose top-left quadrants are empty, and bottom boundary as a subset of points whose 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 denote this relation on boundary points.

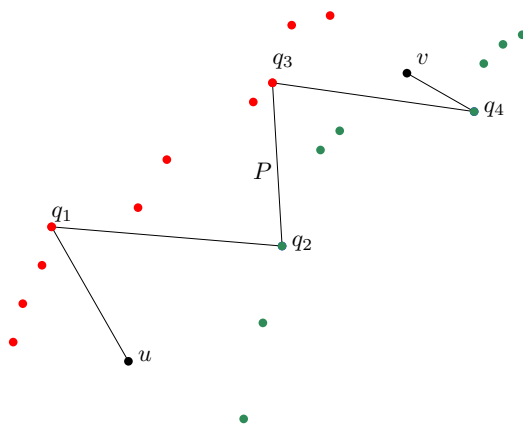
Boundaries are particularly useful when considering distances between points:

► **Proposition 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. Take any shortest path P' and any adjacent q_i and q_{i+1} on P' , assume without loss of generality that $q_{i+1} \in \text{TL}_{q_i}$. Suppose that q_{i+1} is not on the top boundary. We either have $q_{i+2} \in \text{TL}_{q_{i+1}}$ or $q_{i+2} \in \text{BR}_{q_{i+1}}$. Note that by transitivity if $q_{i+2} \in \text{TL}_{q_{i+1}}$, then $q_{i+2} \in \text{TL}_{q_i}$ and we could have a shorter path by removing q_{i+1} . Thus, assume $q_{i+2} \in \text{BR}_{q_{i+1}}$. If q_{i+1} is not on the top boundary, then by definition there exists a boundary point $q' \in \text{TL}_{q_{i+1}}$, and it must be that $q_{i+2} \in \text{BR}_{q'}$. This means that we could replace q_{i+1} by q' , increasing the number of points from the path lying on the boundary, and then repeat the argument.



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

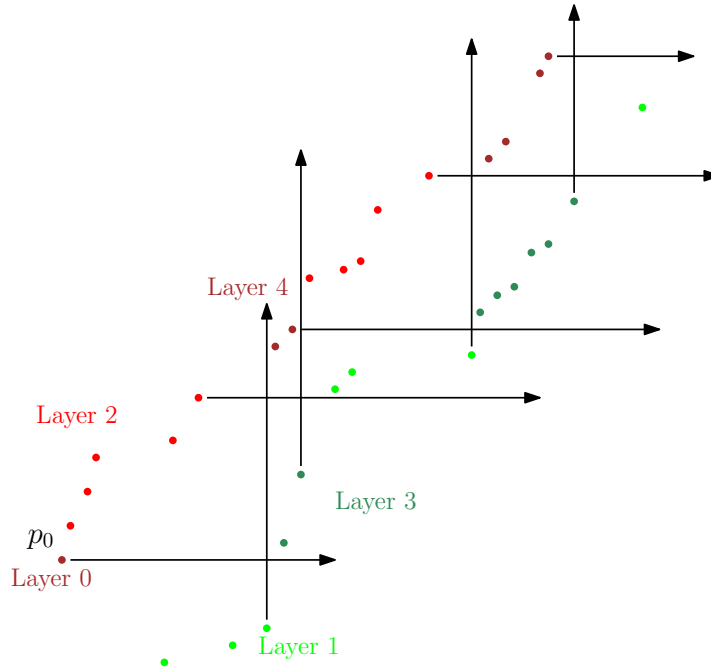


■ **Figure 6** Path between two points alternating between top and bottom boundaries. It is not beneficial to move through points not on boundaries.

Therefore, we have that all points except the first and last ones can always lie on boundaries. See Figure 6 for an illustration. These must be alternating boundaries, as by definition no two points on the same boundary are adjacent. ◀

We partition all points on the boundaries into layers, in the following way. Layer 0 consists of a single left-most point p_0 in the whole set S . Note that p_0 is on the top boundary. 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 for some intuition, 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 almost independent interval graphs.



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

► **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 $\lambda(u) > \lambda(v)$.*

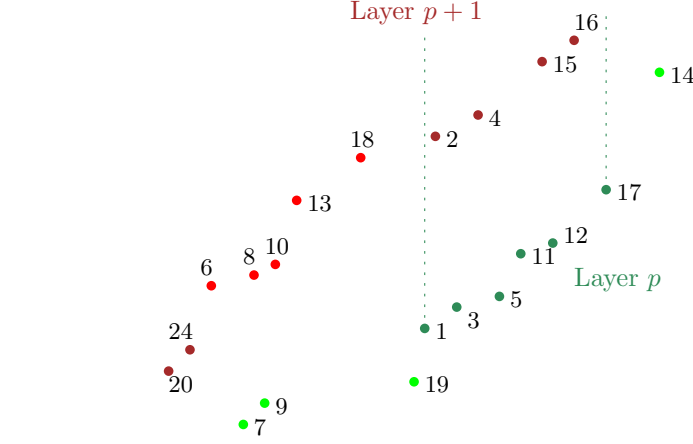
Proof. As noted, points on both boundaries are ordered, and layers switch between boundaries, starting with layer 0 containing just a single left-most point from the top layer. Say ordered points on the top layer are t_0, t_1, t_2, \dots . We prove that there exist strictly increasing numbers $i_0 = 0, i_2, i_4, \dots$ such that layer 0 consists of t_0 , and then any layer $2k$ consists of consecutive points $t_{i_{2k-2}+1}, \dots, t_{i_{2k}}$. Similarly, for points b_0, b_1, \dots on bottom layer, there exists numbers i_1, i_3, \dots defining analogous ranges. Denote by $\text{last}(q)$ the last point (with largest coordinates) in layer q . We prove by induction, in a given order, some intuitive properties (focusing without loss of generality on an odd layer):

► **Proposition 5.** *All of the following holds:*

1. *All points from layer $2k + 1$ are to the right of all points from layer $2k$ (and all points from layer $2k$ are above layer $2k - 1$).*
2. *All points from layer $2k + 1$ are adjacent to $\text{last}(2k)$.*
3. *Layer $2k + 1$ is formed by consecutive points $b_{i_{2k-1}+1}, \dots, b_{i_{2k+1}}$.*
4. *Ordered points from layer $2k$ are adjacent to increasing prefixes of the sequence of points $b_{i_{2k-1}+1}, \dots, b_{i_{2k+1}}$. This means that, firstly, any point t_j with $L(t_j) = 2k$ is adjacent exactly to points $b_{i_{2k-1}+1}, \dots, b_q$ from layer $2k + 1$, for some $q \leq i_{2k+1}$. Secondly, for t_{j+1} with $L(t_{j+1}) = 2k$, t_{j+1} is adjacent to points $b_{i_{2k-1}+1}, \dots, b_r$ with $q \leq r$.*

Proof. The properties for layer $2k + 1 = 1$ are apparent, except for the third point, which can be done as in the induction step. Now consider layer $2k + 1$. As all points from layer $2k$ are adjacent to $\text{last}(2k - 1)$, meaning they are in $\text{TL}_{\text{last}(2k-1)}$, all these points are to the left

of points from layer $2k + 1$. Moreover, the last point in layer $2k$ has the largest y coordinate, thus if any point from layer $2k + 1$ is adjacent to some point from layer $2k$, then it is also adjacent to $\text{last}(2k)$. This gives us the first two properties.

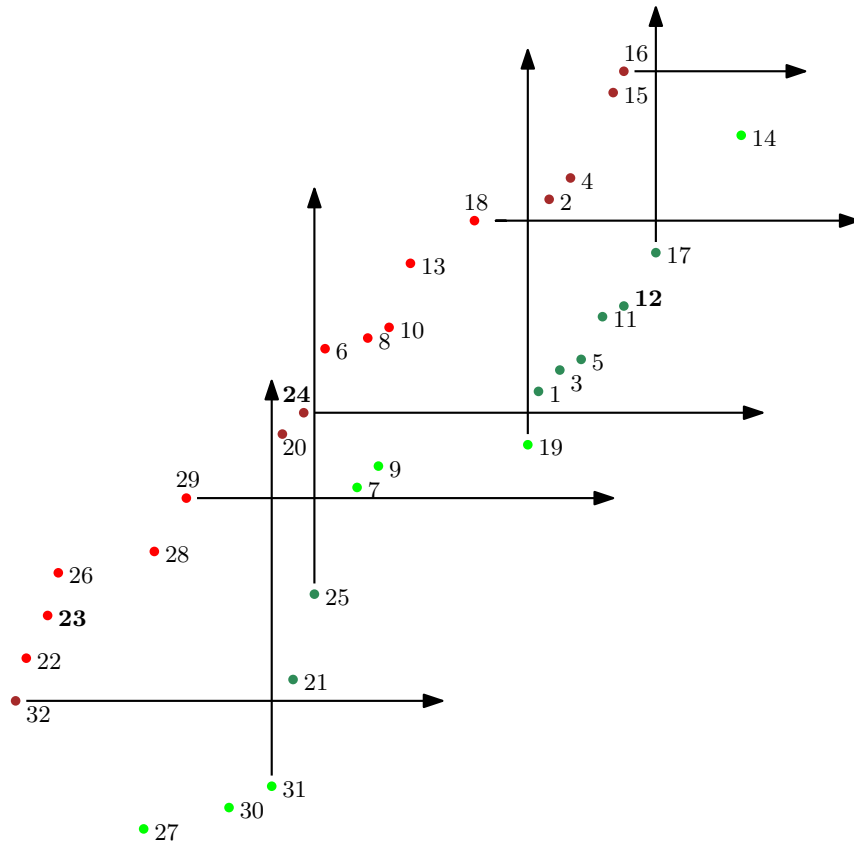


■ **Figure 8** Considering layers p and $p + 1$, a point with λ value of 1 is not adjacent to any point from layer $p + 1$, an empty prefix. The point with value 3 is adjacent to just point with value 2, so range $[2, 2]$. 5 and 11 are adjacent to prefix $[2, 4]$. 12 is adjacent to $[2, 15]$, and finally, 17 is adjacent to all points from layer $p + 1$.

Now, by definition, if points b_q, b_r with $r > q$ are adjacent to some point v on the top boundary, then all points b_q, b_{q+1}, \dots, b_r are adjacent to v . Thus, if $b_{i_{2k-1}+1}$ is adjacent to $\text{last}(2k)$, we get the third property. But it must be adjacent, as otherwise it would be above $\text{last}(2k)$, and then layer $2k + 1$ would be empty. We can apply the above principles to any point from layer $2k$ - it either neighbours the first point in layer $2k + 1$ or no point from this layer. This means any point from layer $2k$ neighbours prefix of points from layer $2k + 1$ (possibly empty, possibly full layer). Moreover t_{j+1} is adjacent to the same points from layer $2k + 1$ that t_j is, and possibly more, as t_{j+1} is above t_j . See Figure 8. ◀

By the definition of layers, for points u, v , $d(u, v) \geq |L(u) - L(v)|$. If $d(u, v) = |L(u) - L(v)|$, we say that there is a *quick path* between them. Going back to the statement of the lemma, it says there exists an ordering λ of boundary points such that there is a quick path between two points exactly when relations between their λ values and layer numbers are opposite. We can create ordering λ in the following greedy way: starting from λ value of 1, always assign current value to the lowest point in the lowest layer such that it is adjacent to no points in the next layer without λ values already assigned, then increment current value. In other words, if we direct all edges from lower to higher layers, we repeatedly choose the lowest (by layer) possible sink, assign to it the current value, and remove the sink from the graph. This is always possible, at the extreme case we can always choose a point from the highest layer containing points without assigned λ value. See Figure 9 for an example.

For correctness, observe that when we choose v from layer k , for all layers larger than k the last point with assigned value is adjacent to all points with assigned values in the next layer. This is by the greedy procedure, as assume there is a layer $j > k$ such that u , the last point with assigned value in layer j , is not adjacent to w , the last point with assigned value in layer $j + 1$. It cannot be that $\lambda(w) < \lambda(u)$, as w does not need to be processed by the greedy procedure before choosing u . But $\lambda(w) > \lambda(u)$ is also impossible by definition of the procedure, since w cannot be processed at any moment after choosing u and before choosing



■ **Figure 9** Boundary points with their λ values. The point with value 23 can reach point 12 using five edges, by path 23-21-20-19-18-12. But 23 cannot reach 24 using two edges, it needs four.

v , as no other point from layer j was chosen between these events and procedure prioritises lower layers. Now, using the above, we can observe that at the moment we assign value for point v in layer k :

- There are quick paths from v to all points with assigned values and in layers larger than k . This is true by the choice of v and transitivity. It is given that v is adjacent to all points in layer $k + 1$ with assigned values, then the last of these points is adjacent to all points in layer $k + 2$ with assigned values, and so on.
- There are no quick paths from v to any point without assigned value in a layer larger than k . This is clear from the greedy procedure, since points are chosen only when all their neighbours from the next layer got assigned values. ◀

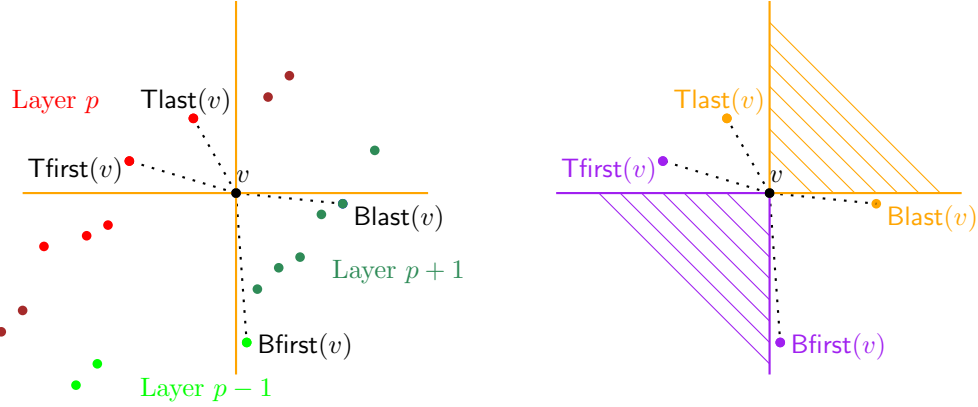
By Lemma 4, we are able to detect when quick paths exist, and to complete knowledge about distances between boundary points we observe the following:

► **Proposition 6.** *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. First let us argue that $d(u, v) \leq L(v) - L(u) + 2$. We observed $\text{last}(i)$ is adjacent to all points in layer $i + 1$. Thus, $(u, \text{last}(L(u) - 1), \text{last}(L(u)), \text{last}(L(u) + 1), \dots, \text{last}(L(v) - 1), v)$ is always a correct path with length $L(v) - L(u) + 2$. By definition of layers, $d(u, v)$ cannot be less than $L(v) - L(u)$. Finally, by Proposition 3 there is a shortest path that alternates between boundaries, so it cannot be of length $L(v) - L(u) + 1$, as we cannot change parity. ◀

86:12 Optimal Distance Labeling for Permutation Graphs

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 as 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 normalise the coordinates so that they are integers, 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).



■ **Figure 10** 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 layer p , one last point in 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 .

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 $Bfirst(v)$ and $Blast(v)$ the first and last points on the bottom boundary adjacent to v and by $Tfirst(v)$ and $Tlast(v)$ the first and last points on the top boundary adjacent to v . Consult Figure 10.

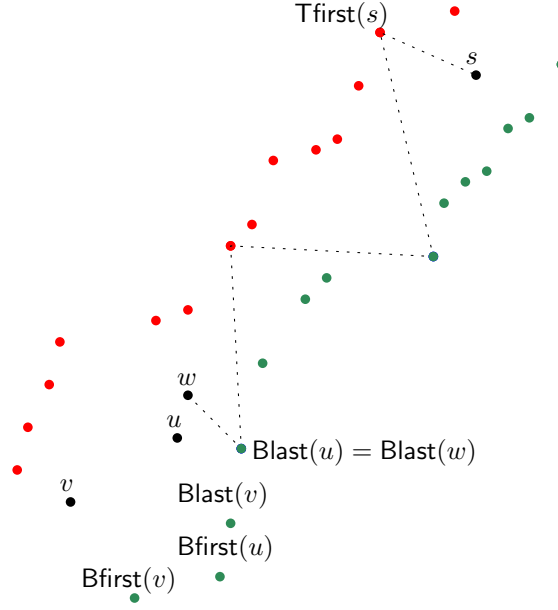
We can make easy observation on points at distance two:

► **Proposition 7.** *For any two points u, v , $d(u, v) \leq 2$ is equivalent to $[Bfirst(u), Blast(u)] \cap [Bfirst(v), Blast(v)] \neq \emptyset$ or $[Tfirst(u), Tlast(u)] \cap [Tfirst(v), Tlast(v)] \neq \emptyset$.*

This is since by Proposition 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 $[Tfirst(u), Tlast(u)] \subseteq [Tfirst(v), Tlast(v)]$, and ranges are never empty.

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

► **Lemma 8.** *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 $Blast(u)$ or $Tlast(u)$ and the penultimate point being either $Bfirst(v)$ or $Tfirst(v)$.*



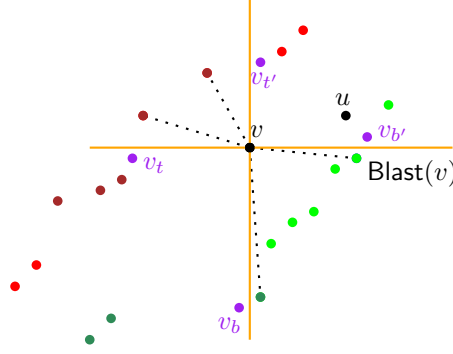
■ **Figure 11** We have $d(u, v) = 2$, even though all $Bfirst(v)$, $Blast(v)$, $Bfirst(u)$, $Blast(u)$ are different, but their ranges do intersect. Moreover, there is a shortest path from w to s with the second point being $Blast(w)$ and the penultimate point being $Tfirst(s)$.

Proof. We will prove the statement for the second point, as the penultimate point is symmetric. By Proposition 3 there always exists a shortest path P with all but extreme points lying on alternating boundaries, with $P = (u = q_0, q_1, q_2, \dots, w, q_{d(u,v)} = v)$, so we denote the penultimate point by w .

Consider layer number of w . As $d(u, v) > 2$ and $u_x < v_x$, it must be that $v, w \in TR_u$ and so $L(w) \geq \min(L(Tlast(u)), L(Blast(u)))$. If $L(w) \geq \max(L(Tlast(u)), L(Blast(u)))$, then by Proposition 6 and Lemma 4 we can replace q_1 with $Blast(u)$ or $Tlast(u)$ (which have the largest λ values in their layers among neighbours of u), while keeping the length of P and w as penultimate point. We are left with $L(w) = \min(L(Tlast(u)), L(Blast(u)))$. Assume $L(Blast(u)) > L(Tlast(u))$, so $L(w) = L(Tlast(u))$ and also $w > Tlast(u)$. Then w is adjacent to $last(L(w) - 1) \in BR_u$ and thus also to $Blast(u)$, so we can have $q_1 = Blast(u)$. In other case, we could similarly set $q_1 = Tlast(u)$. Thus, we can always change q_1 to be $Blast(u)$ or $Tlast(u)$, without changing w . ◀

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 Proposition 7 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 $(2+4+1) \log n + \mathcal{O}(1) = 7 \log n + \mathcal{O}(1)$, by storing for each point v coordinates v_x, v_y , and for all points of interest $Bfirst(v)$, $Blast(v)$, $Tfirst(v)$, and $Tlast(v)$, 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 Proposition 7, and larger distances by Proposition 6, Lemma 4 and 8.



■ **Figure 12** 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 $\text{Blast}(v)$.

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.

► **Definition 9.** Consider v from the initial set and its bottom-right quadrant BR_v . We add two points $v_b = (v_x - \epsilon, \text{Bfirst}(v)_y - \epsilon)$ and $v_{b'} = (\text{Blast}(v)_x + \epsilon, v_y + \epsilon)$ to the set S of points.

See Figure 12. Then, we normalise the coordinates so that we still operate on a permutation of natural numbers. 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 10.** All added points are on the bottom boundary. Moreover, for any two points $u, w \in S$, $d(u, w)$ remains the same.

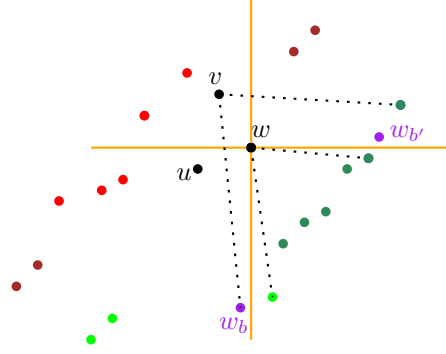
Proof. Considering the first property, we need to observe that when adding a point, its bottom-right quadrant is empty. For $v_{b'}$ it holds as the point is between $\text{Blast}(v)$ and the next point on the bottom boundary on both axes. Thus it changes the status of no point on the bottom boundary and itself is on this boundary. We have a similar situation with v_b .

For the second property, we notice that any point adjacent to $v_{b'}$ is also adjacent to $\text{Blast}(v)$. Since $v_{b'}$ and $\text{Blast}(v)$ lie on the bottom boundary, any adjacent point must be in their top-left quadrant. As $v_{b'} = (\text{Blast}(v)_x + \epsilon, v_y + \epsilon)$ and there are no points with x -coordinate between $\text{Blast}(v)_x$ and $\text{Blast}(v)_x + \epsilon$, if some point is to the left of $v_{b'}$, it is also to the left of $\text{Blast}(v)$, and by definition we have $v_{b'_y} > v_y > \text{Blast}(v)_y$. Similarly, any point adjacent to v_b is also adjacent to $\text{Bfirst}(v)$. Therefore, these points cannot offer any shortcuts in existing shortest paths. ◀

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 $v_t = (\text{Tfirst}(v)_x - \epsilon, v_y - \epsilon)$ and $v_{t'} = (v_x + \epsilon, \text{Tlast}(v)_y + \epsilon)$ to the set S of points. Then, we again normalise the coordinates. This is symmetric and has the same properties.

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

► **Lemma 11.** For any two points v, w from the initial set, $w \in \text{BR}_v$ is equivalent to $\text{Bfirst}(v) < \text{Bfirst}(w) \leq \text{Blast}(w) < \text{Blast}(v)$. Moreover, $w \in \text{TL}_v$ is equivalent to $\text{Tfirst}(v) < \text{Tfirst}(w) \leq \text{Tlast}(w) < \text{Tlast}(v)$.



■ **Figure 13** We have $w \in \text{BR}_v$, and $\text{Bfirst}(v) < \text{Bfirst}(w) < \text{Blast}(w) < \text{Blast}(v)$. Meanwhile, $w \notin \text{BR}_u$, as $\text{Blast}(w) > \text{Blast}(u)$. Auxiliary points which would be added for u, v are not shown to avoid clutter.

Proof. The cases for both boundaries are symmetrical, so we focus on the bottom one.

If $w \in \text{BR}_v$, then by transitivity v is adjacent to $\text{Bfirst}(w)$, and then by definition of w_b , v is also adjacent to w_b . As $w_b < \text{Bfirst}(w)$, we get $\text{Bfirst}(v) < \text{Bfirst}(w)$. Analogous facts hold for $w_{b'}$, therefore implication in the right direction holds.

We consider the left direction and use contraposition. Firstly, we want to show that if $w \notin \text{BR}_v$, then either $\text{Bfirst}(v) > \text{Bfirst}(w)$ or $\text{Blast}(w) > \text{Blast}(v)$. $w \notin \text{BR}_v$ means $w_x < v_x$ or $w_y > v_y$. Assume $w_x < v_x$ and $\text{Bfirst}(v) \leq \text{Bfirst}(w)$. This can be only if $\text{Bfirst}(v) = \text{Bfirst}(w)$, since w is to the left of v and points on boundaries are ordered. As v_b is below $\text{Bfirst}(v)$ and between w_x and v_x , it must be that $v_b \in \text{BR}_w$. So we have $v_b \in \text{BR}_w$ and $v_b < \text{Bfirst}(v) = \text{Bfirst}(w)$, a contradiction, as then v_b should be $\text{Bfirst}(w)$. Similarly using $v_{b'}$ we can show that assuming $w_y > v_y$ and $\text{Blast}(w) \leq \text{Blast}(v)$ leads to a contradiction. See Figure 13. ◀

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 the four λ values, and additionally the layer numbers using only $\log n + \mathcal{O}(1)$ bits.

5 Final Scheme and Conclusion

Our main result, namely Theorem 1, is proved in the full version of this paper. We achieve the improvement on the scheme described in the previous section by further compressing information in the labels. This is possible when we consider values of $\lambda(\text{Blast}(v))$ and $\lambda(\text{Tfirst}(v))$, stored on $2 \log n$ bits, and show how to merge them into a single value somewhere between these two numbers. The main case is when $\lambda(\text{Blast}(v)) < \lambda(\text{Tfirst}(v))$. Denote by w a boundary point with the smallest $\lambda(w)$ value among points with $w_y > v_y$ and $\lambda(w) > \lambda(\text{Blast}(v))$. We observe that it must be that $\lambda(w) \leq \lambda(\text{Tfirst}(v))$. We then store in our label value of $v_{y'} = \lambda(w) - \epsilon$ (to be normalised later), and prove that such a single number can replace both values of $\lambda(\text{Blast}(v))$ and $\lambda(\text{Tfirst}(v))$. Similarly, we can compress $\lambda(\text{Bfirst}(v))$ and $\lambda(\text{Tlast}(v))$ into a single number.

Handling disconnected graphs. Here we describe how to modify distance labeling for connected graphs into distance labeling for possibly disconnected 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.

Assume there is some labeling scheme for the family of connected graphs, with labels of size $g(n) \geq \log n$. Given a not necessarily connected graph, we sort the connected components of the graph by decreasing size, let C_1, C_2, \dots be the obtained order, then proceed with creating a distance labeling for each individual connected component. The final label is the number of connected component of a vertex and then its label in the distance labeling created for the component. If $v \in C_i$, we encode i on $\log i + \mathcal{O}(\log \log n)$ bits, using binary representation of i and indicating its length. We have $|C_i| \leq n/i$, so the modified label size is at most $g(n/i) + \mathcal{O}(\log \log n) + \log i \leq g(n) + \mathcal{O}(\log \log n)$, as claimed.

Conclusion. We have described, improving upon the previous results, 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 the constant 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 other 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. doi:10.1137/S0097539703437211.
- 2 Hüseyin Acan, Sankardeep Chakraborty, Seungbum Jo, and Srinivasa Rao Satti. Succinct encodings for families of interval graphs. *Algorithmica*, 83(3):776–794, 2021. doi:10.1007/S00453-020-00710-W.
- 3 Noga Alon. Asymptotically optimal induced universal graphs. *Geometric and Functional Analysis*, 27, February 2017.
- 4 Noga Alon and Rajko Nenadov. Optimal induced universal graphs for bounded-degree graphs. In *28th SODA*, pages 1149–1157, 2017. doi:10.1137/1.9781611974782.74.
- 5 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. doi:10.1109/FOCS.2015.84.
- 6 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. doi:10.4230/LIPICS.ESA.2016.5.
- 7 Stephen Alstrup, Cyril Gavoille, Esben Bstrup Halvorsen, and Holger Petersen. Simpler, faster and shorter labels for distances in graphs. In *27th SODA*, pages 338–350, 2016. doi:10.1137/1.9781611974331.CH25.
- 8 Stephen Alstrup, Inge Li Gørtz, Esben Bstrup Halvorsen, and Ely Porat. Distance labeling schemes for trees. In *43rd ICALP*, pages 132:1–132:16, 2016. doi:10.4230/LIPICS.ICALP.2016.132.
- 9 Stephen Alstrup, Haim Kaplan, Mikkel Thorup, and Uri Zwick. Adjacency labeling schemes and induced-universal graphs. In *47th STOC*, pages 625–634, 2015. doi:10.1145/2746539.2746545.
- 10 K. A. Baker, Peter C. Fishburn, and Fred S. Roberts. Partial orders of dimension 2. *Networks*, 2(1):11–28, 1972. doi:10.1002/NET.3230020103.
- 11 Fabrice Bazzaro and Cyril Gavoille. Distance labeling for permutation graphs. *Electron. Notes Discret. Math.*, 22:461–467, 2005. doi:10.1016/J.ENDM.2005.06.098.
- 12 Fabrice Bazzaro and Cyril Gavoille. Localized and compact data-structure for comparability graphs. *Discret. Math.*, 309(11):3465–3484, 2009. doi:10.1016/J.DISC.2007.12.091.

- 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. doi:10.1145/3406325.3451102.
- 14 Marthe Bonamy, Cyril Gavoille, and Michał Pilipczuk. Shorter labeling schemes for planar graphs. In *31st SODA*, pages 446–462, 2020. doi:10.1137/1.9781611975994.27.
- 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, 2024. doi:10.4230/LIPICS.ICALP.2024.31.
- 16 Steve Butler. Induced-universal graphs for graphs with bounded maximum degree. *Graphs and Combinatorics*, 25:461–468, 2009. doi:10.1007/S00373-009-0860-X.
- 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. doi:10.1016/S0166-218X(98)00145-0.
- 18 Charles J. Colbourn. On testing isomorphism of permutation graphs. *Networks*, 11(1):13–21, 1981. doi:10.1002/NET.3230110103.
- 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. doi:10.1007/978-3-662-47666-6_45.
- 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. doi:10.1109/FOCS46700.2020.00060.
- 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. doi:10.1016/J.IPL.2008.04.020.
- 23 Shimon Even, Amir Pnueli, and Abraham Lempel. Permutation graphs and transitive graphs. *J. ACM*, 19(3):400–410, 1972. doi:10.1145/321707.321710.
- 24 Pierre Frgaignaud and Cyril Gavoille. Routing in trees. In *28th ICALP*, pages 757–772, 2001. doi:10.1007/3-540-48224-5_62.
- 25 Pierre Frgaignaud and Amos Korman. An optimal ancestry scheme and small universal posets. In *42th STOC*, pages 611–620, 2010. doi:10.1145/1806689.1806773.
- 26 Ofer Freedman, Paweł Gawrychowski, Patrick K. Nicholson, and Oren Weimann. Optimal distance labeling schemes for trees. In *36th PODC*, pages 185–194, 2017. doi:10.1145/3087801.3087804.
- 27 Cyril Gavoille and Arnaud Labourel. Shorter implicit representation for planar graphs and bounded treewidth graphs. In *15th ESA*, pages 582–593, 2007. doi:10.1007/978-3-540-75520-3_52.
- 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. doi:10.1137/050635006.
- 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 Łopuszanski. Shorter labels for routing in trees. In *32nd SODA*, pages 2174–2193, 2021.
- 31 Paweł Gawrychowski, Fabian Kuhn, Jakub Łopuszanski, 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 Uznański. Better distance labeling for unweighted planar graphs. *Algorithmica*, 85(6):1805–1823, 2023. doi:10.1007/S00453-023-01133-Z.
- 33 Hamed Hatami and Pooya Hatami. The implicit graph conjecture is false. In *63rd FOCS*, pages 1134–1137, 2022. doi:10.1109/FOCS54457.2022.00109.

- 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. doi:10.4230/LIPICS.ISAAC.2020.25.
- 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. doi:10.4230/LIPICS.CPM.2024.17.
- 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. doi:10.1016/S0012-365X(00)00204-1.
- 37 Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM J. Discret. Math.*, 5(4):596–603, 1992. doi:10.1137/0405049.
- 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. doi:10.1016/J.DAM.2004.03.005.
- 39 Amos Korman. Labeling schemes for vertex connectivity. *ACM Trans. Algorithms*, 6(2):39:1–39:10, 2010. doi:10.1145/1721837.1721855.
- 40 Amos Korman, David Peleg, and Yoav Rodeh. Constructing labeling schemes through universal matrices. *Algorithmica*, 57(4):641–652, 2010. doi:10.1007/S00453-008-9226-7.
- 41 Ross M. McConnell and Jeremy P. Spinrad. Modular decomposition and transitive orientation. *Discret. Math.*, 201(1-3):189–241, 1999. doi:10.1016/S0012-365X(98)00319-7.
- 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, March 2000. doi:10.1002/(SICI)1097-0118(200003)33:3<167::AID-JGT7>3E3.O.CO;2-5.
- 48 David Peleg. Informative labeling schemes for graphs. *Theor. Comput. Sci.*, 340(3):577–593, 2005. doi:10.1016/J.TCS.2005.03.015.
- 49 Arseny M. Shur and Mikhail Rubinchik. Distance labeling for families of cycles. In *49th SOFSEM*, pages 471–484, 2024. doi:10.1007/978-3-031-52113-3_33.
- 50 Mikkel Thorup and Uri Zwick. Compact routing schemes. In *13th SPAA*, pages 1–10, 2001. doi:10.1145/378580.378581.
- 51 Konstantinos Tsakalidis, Sebastian Wild, and Viktor Zamaraev. Succinct permutation graphs. *Algorithmica*, 85(2):509–543, 2023. doi:10.1007/S00453-022-01039-2.