Track Layouts, Forbidden Patterns, and Degree Bounds

Axel Fridman

December 5, 2025

Abstract

We study ordered layouts of a graph on k "tracks" in which edges are constrained by nearest-neighbor rules. This notion comes from the Investigathon formulation in terms of forbidden colored patterns on triples of vertices. We make the correspondence between the two formalisms precise, define *k-track triplet-legal layouts*, and then investigate how many tracks (i.e., "colors") are necessary for a given graph. We prove degree bounds, planarity results for $k \leq 2$, and structural examples such as cycles and graphs obtained from a path by adding one extra edge. Along the way we highlight exactly when $k <= 2$ ("less than three colors") is possible.

Contents

1 Basic Setup: Tracks, Order, and Legal Neighbors

Throughout, $G = (V, E)$ is a finite simple undirected graph.

1.1 Ordered k-track layouts

We begin with the purely combinatorial structure of tracks and a global order.

Definition 1.1 (Ordered k-track layout)**.** Let $k \in \mathbb{N}$. A *k-track ordered layout* of a graph $G = (V, E)$ is a pair

$$(\tau, p)$$

consisting of

- a *track assignment*
$$\tau : V \to \{1, \dots, k\},$$
- and a bijective *position map*
$$p : V \to \{1, \dots, |V|\}.$$

We write $x < y$ if and only if $p(x) < p(y)$ and think of all vertices laid out from left to right according to p.

An example is shown in Figure 1.

For each track $t \in \{1, \dots, k\}$ we define the vertex set

$$V_t := \{v \in V : \tau(v) = t\},$$

ordered by increasing $p(\cdot)$ along that track.

Definition 1.2 (Predecessor, successor on a track)**.** Given a k-track layout (τ, p), a vertex $v \in V$, and a track t, we define:

$$\text{pred}_t(v) := \text{the vertex in } V_t \text{ with largest } p(\cdot) \text{ strictly less than } p(v) \text{ (if it exists)},$$
$$\text{succ}_t(v) := \text{the vertex in } V_t \text{ with smallest } p(\cdot) \text{ strictly greater than } p(v) \text{ (if it exists)}.$$

If such a vertex does not exist, the predecessor/successor is said to be "nonexistent." Intuitively, these are the nearest neighbors of v on track t to the left/right in the global order.

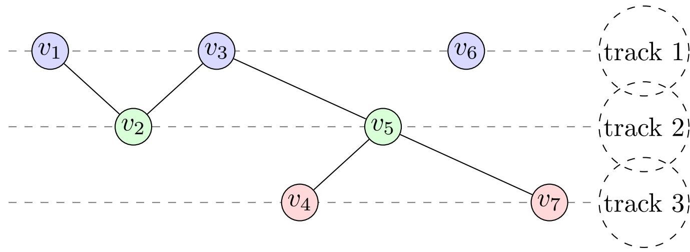

Figure 1: An example of a 3-track ordered layout: global order is left-to-right, colors indicate the track assignment τ.

See Figure 2 for a pictorial view along a single track.

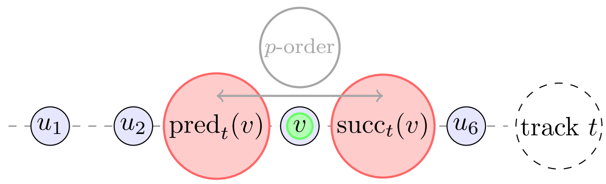

Figure 2: Predecessor and successor of a vertex v on a single track t in the global order.

Definition 1.3 (Legal neighbor set and host graph)**.** Given (τ, p), the *legal neighbor set* of $v \in V$ is

$$N_{\text{legal}}(v) := \big\{\mathrm{pred}_t(v), \mathrm{succ}_t(v) : t = 1, \dots, k\big\} \setminus \{\text{nonexistent}\}.$$

The corresponding *host graph* $H(\tau, p)$ on vertex set V has edge set

$$E\big(H(\tau, p)\big) := \big\{\{u, v\} : u \in N_{\text{legal}}(v)\big\}.$$

Figure 3 shows $N_{\text{legal}}(v)$ on three tracks.

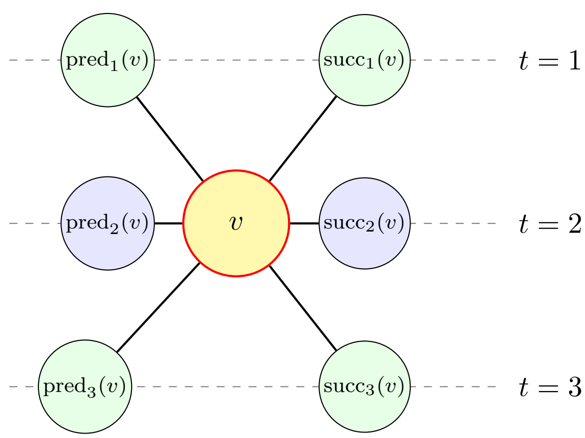

Figure 3: The legal neighbor set $N_{\text{legal}}(v)$.

Definition 1.4 (k-track nearest-neighbor representable graph)**.** A graph $G = (V, E)$ is *k-track nearest-neighbor representable* if there exists a k-track ordered layout (τ, p) of V such that

$$E \subseteq E\big(H(\tau, p)\big),$$

i.e. every edge of G is a legal edge in the host graph. Equivalently,

$$\forall v \in V, \quad N_G(v) \subseteq N_{\text{legal}}(v).$$

The smallest such k (if it exists) is the *nearest-neighbor track number* of G.

Figure 4 illustrates a typical host graph when $k = 1$. Figure 5 shows a typical host graph when $k = 2$.

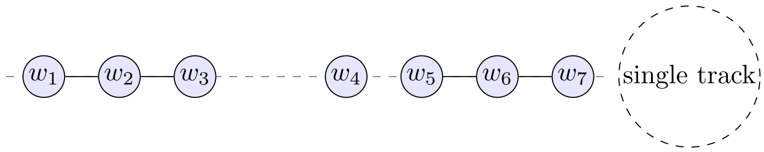

Figure 4: With a single track, the host graph is a disjoint union of paths and isolated vertices (a linear forest).

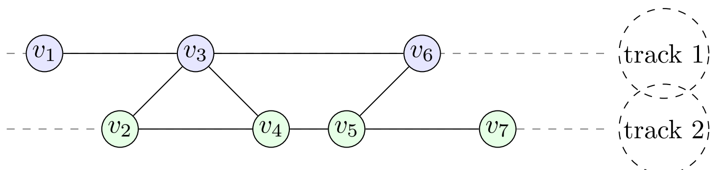

Figure 5: With two tracks, the host graph may have path-like pieces on each track plus cross-track edges between nearest neighbors on the other track.

Remark 1.5 (What $k = 1$ really means)**.** When $k = 1$, every vertex has at most one predecessor and one successor, so the host graph $H(\tau, p)$ is a subgraph of a simple path. Thus any 1-track nearest-neighbor representable graph is a disjoint union of paths and isolated vertices (a *linear forest*). This will be important when we discuss whether "less than two colors" (tracks) is possible.

Remark 1.6 (What $k = 2$ really means)**.** When $k = 2$, each vertex can have at most two legal neighbors on its own track (one predecessor and one successor) and at most two legal neighbors on the other track (again, one predecessor and one successor there). Thus every vertex in the host graph $H(\tau, p)$ has degree at most 4, and $H(\tau, p)$ can be viewed as two path-like layers (one per track) with additional nearest-neighbor cross edges between the layers, as in Figure 5. In particular, any 2-track nearest-neighbor representable graph is a subgraph of such a "ladder-like" planar host graph. This will be important when we discuss when "less than three colors" (tracks) is possible.

2 From Forbidden Colored Patterns to Triplet Constraints

The Investigathon formulation uses forbidden *colored patterns* on triples of vertices. We now translate that language into our k-track layout setting.

2.1 Colored patterns on triples

Definition 2.1 (Colored pattern on three vertices)**.** Fix the index set $\{1, 2, 3\}$. A (plain) *pattern* is a pair

$$(E_P, N_P),$$

where $E_P, N_P \subseteq \{1,2,3\} \times \{1,2,3\}$ encode edges that must be present and edges that must be absent between the three distinguished positions.

A *colored pattern* is a triple

$$P = (E_P, N_P, C_P),$$

where (E_P, N_P) is a pattern as above and $C_P \subseteq \{1,2,3\}$ is the set of indices that are required to lie in the same color class.

In the Investigathon setting, the input is:

- a linear order $<$ on $V(G)$, and
- a partition ("coloring") $\mathcal{S} = \{S_1, \dots, S_k\}$ of $V(G)$ into k color classes.

Definition 2.2 (Realizing a colored pattern)**.** Given $(\mathcal{S}, <)$ as above and a colored pattern $P = (E_P, N_P, C_P)$, an ordered triple of distinct vertices

$$v_1 < v_2 < v_3$$

realizes P if:

(i) for every $(i,j) \in E_P$ we have $\{v_i, v_j\} \in E(G)$;

(ii) for every $(i,j) \in N_P$ we have $\{v_i, v_j\} \notin E(G)$;

(iii) all v_i with $i \in C_P$ lie in a common color class in $\mathcal{S}$.

We say that $(\mathcal{S}, <)$ *avoids* P if no triple $v_1 < v_2 < v_3$ realizes P.

Given a finite family Π of colored patterns, $(\mathcal{S}, <)$ *avoids* Π if it avoids every $P \in \Pi$.

2.2 The Investigathon patterns

In our case, the forbidden family Π consists of the two patterns

$$\begin{aligned} P^{(1)}: &\quad E_{P^{(1)}} = \{(1,3)\}, \quad N_{P^{(1)}} = \varnothing, \quad C_{P^{(1)}} = \{1,2\}, \\ P^{(2)}: &\quad E_{P^{(2)}} = \{(1,3)\}, \quad N_{P^{(2)}} = \varnothing, \quad C_{P^{(2)}} = \{2,3\}. \end{aligned}$$

Intuitively:

- $P^{(1)}$ forbids triples $x < y < z$ with $\{x,z\} \in E(G)$ and x, y of the same color;
- $P^{(2)}$ forbids triples $x < y < z$ with $\{x,z\} \in E(G)$ and y, z of the same color.

See Figure 6 and 7.

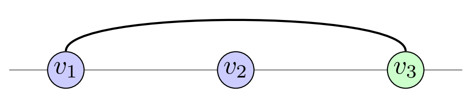

Figure 6: Forbidden pattern $P^{(1)}$: $v_1 < v_2 < v_3$ with $\{v_1, v_3\} \in E(G)$ and v_1, v_2 in the same color class.

A *solution with at most k colors* in the Investigathon sense is precisely a pair $(\mathcal{S}, <)$ with $|\mathcal{S}| \leq k$ that avoids $P^{(1)}$ and $P^{(2)}$.

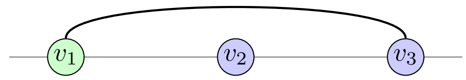

Figure 7: Forbidden pattern $P^{(2)}$: $v_1 < v_2 < v_3$ with $\{v_1, v_3\} \in E(G)$ and v_2, v_3 in the same color class.

2.3 Tracks as color classes

Given a k-track layout (τ, p), the track assignment induces a partition into color classes

$$S_i := \{v \in V(G) : \tau(v) = i\}, \qquad \mathcal{S} = \{S_1, \dots, S_k\},$$

and the global order $<$ is defined by p.

Conversely, given a partition $\mathcal{S} = \{S_1, \dots, S_k\}$ and a linear order $<$, we obtain a k-track layout by labeling the parts $S_1, \dots, S_k$ and setting $\tau(v) = i$ for $v \in S_i$, and letting p be any bijection consistent with $<$.

Thus there is a one-to-one correspondence between:

- solutions $(\mathcal{S}, <)$ with at most k colors, and
- k-track ordered layouts (τ, p).

2.4 Triplet constraint formulation

We now restate avoidance of $P^{(1)}$ and $P^{(2)}$ in a compact way.

Lemma 2.3 (Colored patterns vs. triplet constraint)**.** *Let $G = (V, E)$ be a graph and let (τ, p) be a k-track ordered layout with associated order $<$ and induced partition $\mathcal{S}$ as above. Then the following are equivalent:*

(1) $(\mathcal{S}, <)$ avoids both colored patterns $P^{(1)}$ and $P^{(2)}$.

(2) For every triple $x, y, z \in V$ with

$$p(x) < p(y) < p(z) \quad \text{and} \quad \{x, z\} \in E(G),$$

we have

$$\tau(y) \neq \tau(x) \quad \text{and} \quad \tau(y) \neq \tau(z).$$

Proof. (1) $\Rightarrow$ (2): Suppose there exist x, y, z with $p(x) < p(y) < p(z)$ and $\{x, z\} \in E(G)$ such that $\tau(y) = \tau(x)$ or $\tau(y) = \tau(z)$.

Write $v_1 := x$, $v_2 := y$, $v_3 := z$. Then $\{v_1, v_3\} \in E(G)$. If $\tau(y) = \tau(x)$ then v_1 and v_2 lie in the same color class and (v_1, v_2, v_3) realizes $P^{(1)}$, contradicting avoidance of $P^{(1)}$. If $\tau(y) = \tau(z)$ then (v_1, v_2, v_3) realizes $P^{(2)}$, a contradiction. Thus $\tau(y)$ differs from both $\tau(x)$ and $\tau(z)$.

(2) $\Rightarrow$ (1): Assume (2) holds. Suppose some triple $v_1 < v_2 < v_3$ realizes $P^{(1)}$. Then $\{v_1, v_3\} \in E(G)$ and v_1, v_2 lie in the same color class, i.e. $\tau(v_1) = \tau(v_2)$, contradicting (2) with $(x, y, z) = (v_1, v_2, v_3)$. An identical argument with roles of positions $\{1, 2\}$ and $\{2, 3\}$ exchanged shows that no triple realizes $P^{(2)}$ either. $\square$

Definition 2.4 (k-track triplet-legal layout)**.** A k-track ordered layout (τ, p) of G is *triplet-legal* if for every $x, y, z \in V$ with

$$p(x) < p(y) < p(z) \quad \text{and} \quad \{x, z\} \in E(G),$$

we have

$$\tau(y) \neq \tau(x) \quad \text{and} \quad \tau(y) \neq \tau(z).$$

By Lemma 2.3, triplet-legal layouts are exactly the layouts corresponding to Investigathon solutions avoiding $P^{(1)}$ and $P^{(2)}$.

Figure 8 contrasts a forbidden and two legal triples, one using fewer colors than the other.

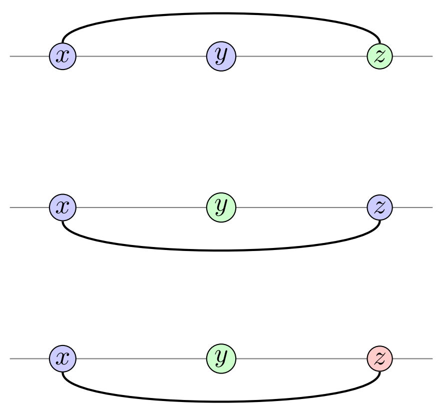

Figure 8: Top: a forbidden triple where y shares a track (color) with x. Middle: a legal triple where x shares a track (color) with z. Bottom: a legal triple where the middle vertex lies on a third track.

Remark 2.5 (Nearest-neighbor vs. triplet-legal)*.* Nearest-neighbor representability is a *stronger* requirement than being triplet-legal: in a nearest-neighbor layout, every edge must be between nearest neighbors on some track; in a triplet-legal layout we only forbid certain colored patterns on triples. In the next section we show that nearest-neighbor layouts automatically satisfy the triplet constraint.

3 Chromatic Number vs. Track-Based "Colors"

The track index $\tau(v)$ behaves a bit like a color, but it is *not* a proper graph coloring in the classical sense (edges within a track are allowed). We only need two standard facts from ordinary graph coloring for analogy:

- If $G_1, \ldots, G_r$ are the connected components of G, then

$$\chi(G) \;=\; \max_{1 \leq i \leq r} \chi(G_i).$$

- If G contains a clique of size r, then $\chi(G) \geq r$; in particular a K_4 forces at least four colors.

In the nearest-neighbor setting, track labels play the role of "colors", and we will see analogues of both statements for the track number.

3.1 Track number and connected components

Definition 3.1 (Nearest-neighbor track number)**.** The *nearest-neighbor track number* of a graph G, denoted $\operatorname{tn}(G)$, is the smallest $k \in \mathbb{N}$ for which G is k-track nearest-neighbor representable.

Proposition 3.2 (Track number and connected components)**.** *Let $G_1, \ldots, G_r$ be the connected components of G. Then*

$$\operatorname{tn}(G) = \max_{1 \le i \le r} \operatorname{tn}(G_i).$$

Remark 3.3. Proposition 3.2 is directly analogous to the classical fact that $\chi(G) = \max_i \chi(G_i)$ for the chromatic number: in both settings, the number of "colors" needed for the whole graph is the maximum over its connected components.

Proof. Lower bound. Any layout witnessing G as k-track nearest-neighbor representable restricts to a layout for each G_i, so $\operatorname{tn}(G_i) \le k$. Taking the minimum over all such k gives $\max_i \operatorname{tn}(G_i) \le \operatorname{tn}(G)$.

Upper bound. Let $k_i = \operatorname{tn}(G_i)$ and $k := \max_i k_i$. For each component G_i choose a k_i-track layout (τ_i, p_i) witnessing nearest-neighbor representability. Relabel tracks so that τ_i takes values in $\{1, \ldots, k\}$ (we simply do not use all labels when $k_i < k$).

Now place the components one after another in the global order: define p by stacking the orders $p_1, \ldots, p_r$ with disjoint ranges, and set $\tau(v) = \tau_i(v)$ for $v \in V(G_i)$. This gives a k-track layout (τ, p) of G in which every edge inside each component remains legal. There are no edges between components, so nothing else needs to be checked. Hence G is k-track nearest-neighbor representable and $\operatorname{tn}(G) \le k$. $\square$

Figure 9 shows how track layouts for components can be stacked in the global order while reusing the same set of track labels.

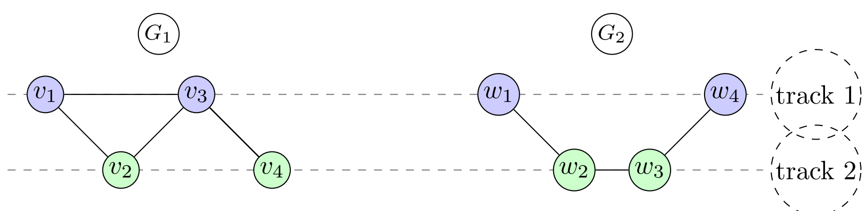

Figure 9: Layouts for different components share the same track labels. The global order stacks the components one after another.

Remark 3.4 (When is fewer than 2 tracks enough?). From Section 1, a 1-track host graph is always a disjoint union of paths and isolated vertices (a linear forest). Thus

$$\operatorname{tn}(G) = 1 \quad \Longleftrightarrow \quad G \text{ is a linear forest},$$

and any graph containing a cycle requires at least two tracks. Figure 10 illustrates why a cycle cannot live on a single track.

4 Nearest-Neighbor Layouts Satisfy the Triplet Constraint

We now connect the nearest-neighbor condition back to the triplet constraint of Lemma 2.3.

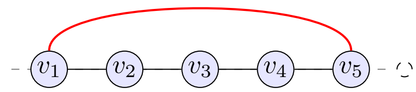

Figure 10: On a single track, only edges between consecutive vertices can be legal. The edge $\{v_1, v_5\}$ needed to complete a cycle skips three vertices and cannot be realized.

Lemma 4.1 (Triplet constraint for nearest-neighbor layouts)**.** *Let (τ, p) be a k-track ordered layout of a graph $G = (V, E)$ and suppose that G is k-track nearest-neighbor representable with respect to (τ, p), i.e.*

$$N_G(v) \subseteq N_{\text{legal}}(v) \quad \textit{for all } v \in V.$$

Let $x, y, z \in V$ with

$$p(x) < p(y) < p(z),$$

and suppose $\{x, z\} \in E$. Then

$$\tau(y) \neq \tau(x) \quad \textit{and} \quad \tau(y) \neq \tau(z).$$

In words: any vertex strictly between adjacent vertices x and z in the global order must lie on a track different from both $\tau(x)$ and $\tau(z)$.

Figure 11 shows the situation in Lemma 4.1.

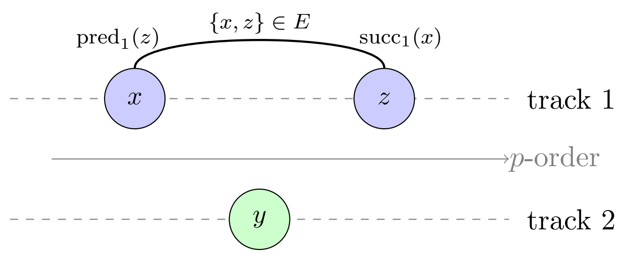

Figure 11: If $\{x, z\}$ is an edge coming from nearest neighbors on track 1, then no other vertex on track 1 can lie between them in the global order. Hence any middle vertex y must live on a different track.

Proof. Since $\{x, z\} \in E$ and G is nearest-neighbor representable, we have

$$z \in N_{\text{legal}}(x) \quad \text{and} \quad x \in N_{\text{legal}}(z).$$

Step 1: z is the successor of x on track $\tau(z)$.

By definition of $N_{\text{legal}}(x)$, there exists a track t such that

$$z = \text{pred}_t(x) \quad \text{or} \quad z = \text{succ}_t(x).$$

If $z = \text{pred}_t(x)$ then $p(z) < p(x)$, contradicting $p(x) < p(z)$; hence

$$z = \text{succ}_t(x).$$

Moreover, $\text{succ}_t(x)$ lies on track t, hence $\tau(z) = t$ and

$$z = \text{succ}_{\tau(z)}(x).$$

By definition of successor, there is no vertex w on track $\tau(z)$ with global position strictly between $p(x)$ and $p(z)$, i.e.

$$\text{no } w \text{ satisfies } \tau(w) = \tau(z) \text{ and } p(x) < p(w) < p(z).$$

Since $p(x) < p(y) < p(z)$, we cannot have $\tau(y) = \tau(z)$.

Step 2: x is the predecessor of z on track $\tau(x)$.

Similarly, since $x \in N_{\text{legal}}(z)$ there exists some track s such that

$$x = \text{pred}_s(z) \quad \text{or} \quad x = \text{succ}_s(z).$$

Because $p(x) < p(z)$, x cannot be a successor of z, so

$$x = \text{pred}_s(z).$$

Hence $\tau(x) = s$ and

$$x = \text{pred}_{\tau(x)}(z).$$

Again by definition of predecessor, there is no vertex w on track $\tau(x)$ with $p(x) < p(w) < p(z)$. In particular, $\tau(y) \neq \tau(x)$.

Combining both steps, we obtain $\tau(y) \neq \tau(x)$ and $\tau(y) \neq \tau(z)$. □

Corollary 4.2. *Every k-track nearest-neighbor layout (τ, p) is a k-track triplet-legal layout. In particular, any nearest-neighbor solution automatically avoids the Investigathon patterns $P^{(1)}$ and $P^{(2)}$.*

Proof. Apply Lemma 4.1 and then use Lemma 2.3. □

5 Degree Bounds and Clique Constraints

We now quantify how the number of tracks k controls the possible degrees and cliques in a nearest-neighbor representable graph. This addresses one natural way to *lower-bound* the number of "colors" (tracks) required.

5.1 A K_4 forces at least three tracks

Theorem 5.1 (A 4-clique forces $k \geq 3$)**.** *Let $G = (V, E)$ be a graph that is k-track nearest-neighbor representable for some $k \in \mathbb{N}$. If G contains a clique of size 4, then $k \geq 3$. Equivalently, no 1- or 2-track layout can represent a 4-clique.*

Proof. Let $H \subseteq G$ be a subgraph isomorphic to K_4 with vertex set $\{a, b, c, d\}$. Let (τ, p) be a k-track ordered layout witnessing nearest-neighbor representability of G.

Rename a, b, c, d so that

$$p(a) < p(b) < p(c) < p(d).$$

Since H is complete, every pair among $\{a, b, c, d\}$ is an edge.

Using Lemma 4.1, we apply the triplet constraint to all triples (x, y, z) with $x < y < z$ where $\{x, z\}$ is an edge (always true in K_4). The relevant triples and consequences are:

$$\tau(b) \neq \tau(a), \quad \tau(b) \neq \tau(c), \qquad \text{from } (a, b, c) \text{ and edge } \{a, c\}, \tag{1}$$
$$\tau(b) \neq \tau(a), \quad \tau(b) \neq \tau(d), \qquad \text{from } (a, b, d) \text{ and edge } \{a, d\}, \tag{2}$$
$$\tau(c) \neq \tau(a), \quad \tau(c) \neq \tau(d), \qquad \text{from } (a, c, d) \text{ and edge } \{a, d\}, \tag{3}$$
$$\tau(c) \neq \tau(b), \quad \tau(c) \neq \tau(d), \qquad \text{from } (b, c, d) \text{ and edge } \{b, d\}. \tag{4}$$

From (1) and (2) we see that
$$\tau(b) \notin \{\tau(a), \tau(c), \tau(d)\},$$
and from (3) and (4) that
$$\tau(c) \notin \{\tau(a), \tau(b), \tau(d)\}.$$

Suppose $k \leq 2$. Then tracks are $\{1, 2\}$. Without loss of generality, let $\tau(a) = 1$.

Case 1: $\tau(d) = 1$. Then (2) implies $\tau(b) \neq \tau(a)$ and $\tau(b) \neq \tau(d)$, so $\tau(b) \neq 1$ and hence $\tau(b) = 2$. Similarly, (3) implies $\tau(c) \neq 1$, so $\tau(c) = 2$. But then (1) requires $\tau(b) \neq \tau(c)$, impossible.

Case 2: $\tau(d) = 2$. Then (2) says $\tau(b) \neq 1$ and $\tau(b) \neq 2$, which is impossible with only two tracks.

In all cases we get a contradiction, so $k \geq 3$. □

Figure 12 shows a K_4 whose vertices cannot be assigned to only two tracks without violating Lemma 4.1.

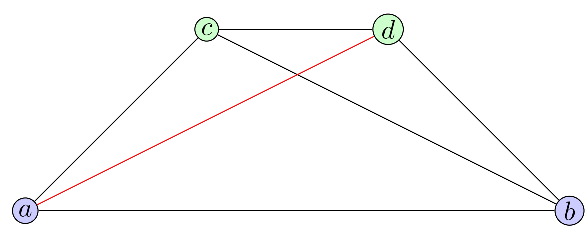

Figure 12: A K_4 drawn with two "track colors". No ordering of a, b, c, d can avoid creating a triple $x < y < z$ with $\{x, z\} \in E$ where y shares a track with x or z, so at least three tracks are needed.

5.2 A degree bound: $\Delta(G) \leq 2k$

Lemma 5.2 (Degree bound for k-track nearest-neighbor layouts)**.** *Let $G = (V, E)$ be a simple undirected graph that is k-track nearest-neighbor representable. That is, there exists a layout (τ, p) with*
$$\tau : V \to \{1, \ldots, k\}, \qquad p : V \to \{1, \ldots, |V|\}$$
such that every edge is legal:
$$\forall \{u, v\} \in E, \quad v \in N_{\text{legal}}(u) \quad \textit{(equivalently } u \in N_{\text{legal}}(v)\textit{)}.$$
Then for every vertex $v \in V$ we have
$$\deg_G(v) \leq 2k.$$
In particular, the maximum degree $\Delta(G)$ satisfies
$$\Delta(G) \leq 2k.$$

Proof. Fix $v \in V$. For each track t, let $V_t = \{x : \tau(x) = t\}$ and define $\text{pred}_t(v), \text{succ}_t(v)$ as before. The legal neighbor set is
$$N_{\text{legal}}(v) = \{\text{pred}_1(v), \text{succ}_1(v), \ldots, \text{pred}_k(v), \text{succ}_k(v)\} \setminus \{\text{nonexistent}\}.$$

By nearest-neighbor representability,
$$N_G(v) \subseteq N_{\text{legal}}(v),$$

so

$$\deg_G(v) = |N_G(v)| \le |N_{\text{legal}}(v)|.$$

For each track t, at most two vertices can appear in $N_{\text{legal}}(v)$ from track t, namely $\text{pred}_t(v)$ and $\text{succ}_t(v)$ if they exist. Define

$$S_t(v) := \{\text{pred}_t(v), \text{succ}_t(v)\} \setminus \{\text{nonexistent}\},$$

so $|S_t(v)| \le 2$ and

$$N_{\text{legal}}(v) = \bigcup_{t=1}^{k} S_t(v).$$

Thus

$$|N_{\text{legal}}(v)| \le \sum_{t=1}^{k} |S_t(v)| \le \sum_{t=1}^{k} 2 = 2k.$$

Hence $\deg_G(v) \le 2k$ for all v, and taking the maximum gives $\Delta(G) \le 2k$. □

Corollary 5.3 (Degree-based lower bound on tracks)**.** *If G is k-track nearest-neighbor representable, then*

$$k \ \ge \ \left\lceil \frac{\Delta(G)}{2} \right\rceil.$$

Figure 13 shows a vertex of degree 5, which forces at least three tracks by Corollary 5.3

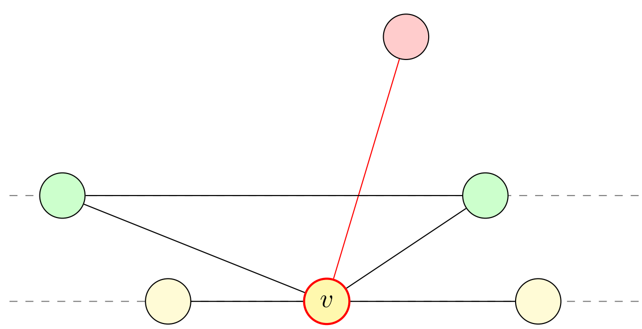

Figure 13: A vertex of degree 5: by $\Delta(G) \le 2k$, this implies $k \ge 3$.

Proof. Immediate from Lemma 5.2. □

5.3 Edge bounds for two tracks: at most $2n - 3$ edges

So far we have seen that a k-track nearest-neighbor layout forces $\Delta(G) \le 2k$ (Lemma 5.2). For $k = 2$ we can say much more: not only is the maximum degree at most 4, but the *total number of edges* is at most $2n - 3$ for an n-vertex graph. This matches the familiar extremal bound for outerplanar graphs.

We work relative to a fixed 2-track ordered layout (τ, p) and its host graph $H(\tau, p)$.

Definition 5.4 (Left-degree)**.** Given a linear order $v_1, \dots, v_n$ of $V(G)$ induced by p, the *left-degree* of v_i is

$$\deg^-(v_i) := \big|\{\, u \in N_G(v_i) : p(u) < p(v_i) \,\}\big|.$$

Equivalently, $\deg^-(v_i)$ counts neighbors of v_i strictly to the left of v_i in the global order.

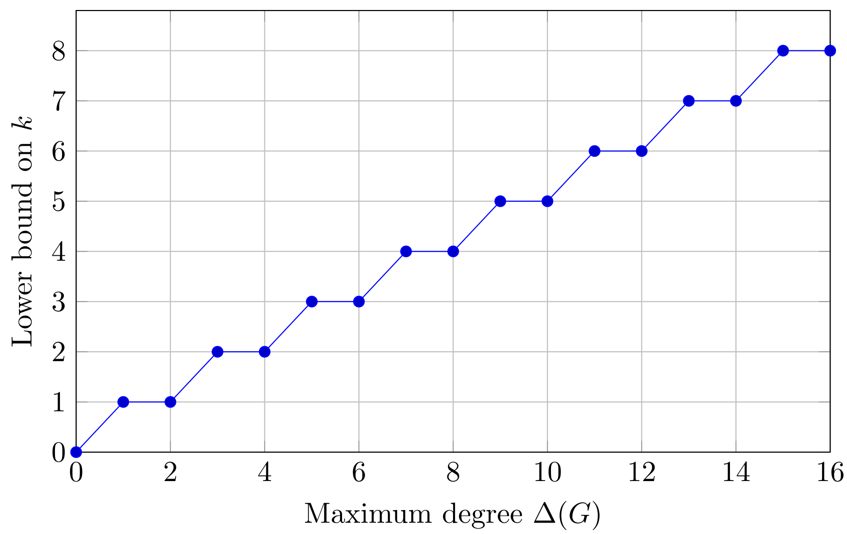

Figure 14: Lower bound on the number of tracks: $k \geq \lceil \Delta(G)/2 \rceil$. In particular, $\Delta(G) \geq 5$ forces $k \geq 3$.

Every edge $\{u, v\}$ is counted exactly once in the sum of left-degrees, at its right endpoint.

Lemma 5.5 (At most one predecessor per track)**.** *Let G be k-track nearest-neighbor representable with respect to (τ, p), and let $v \in V(G)$. Then for each track $t \in \{1, \ldots, k\}$, v has at most one neighbor on track t to its left. In particular,*

$$\deg^{-}(v) \;\leq\; k$$

for every vertex v.

Proof. Fix v and a track t. By definition, the only candidate left neighbor of v on track t that is legal in the host graph is $\mathrm{pred}_t(v)$ (if it exists). There cannot be two distinct neighbors x, y on track t with $p(x) < p(y) < p(v)$ and both adjacent to v, because only the closest one to the left can be a legal predecessor on that track.

Thus, on track t, there is at most one neighbor of v to the left. Summing over the k tracks gives $\deg^{-}(v) \leq k$. $\square$

Theorem 5.6 (Edge bound for two-track nearest-neighbor layouts)**.** *Let G be a finite simple graph on n vertices that is 2-track nearest-neighbor representable. Then*

$$|E(G)| \;\leq\; 2n - 3.$$

Moreover, for every $n \geq 2$ there exists such a graph with exactly $2n - 3$ edges.

Proof. Let (τ, p) be a 2-track layout and let $H = H(\tau, p)$ be its host graph. Since G is a subgraph of H, it suffices to prove $|E(H)| \leq 2n - 3$.

Write $V = \{v_1, \ldots, v_n\}$ in increasing order of p, so $p(v_i) = i$. Every edge of H has a unique right endpoint, so

$$|E(H)| = \sum_{i=1}^{n} \deg_H^{-}(v_i).$$

Now:

- For v_1, there are no vertices to the left, so $\deg^-_H(v_1) = 0$.
- For v_2, the only vertex to the left is v_1, so $\deg^-_H(v_2) \le 1$ in any simple graph.
- For each $i \ge 3$, Lemma 5.5 with $k = 2$ gives $\deg^-_H(v_i) \le 2$.

Therefore,

$$|E(H)| = \sum_{i=1}^{n} \deg^-_H(v_i) \ \le\ 0 + 1 + 2(n-2) = 2n - 3.$$

Since G is a subgraph of H, we also have $|E(G)| \le |E(H)| \le 2n - 3$.

For tightness, fix $n \ge 2$ and construct a 2-track layout on vertices $v_1, \dots, v_{n-1}, w$ as follows:

- Put $v_1, \dots, v_{n-1}$ on track 1 in the order $p(v_i) = i$ and connect all consecutive pairs $\{v_i, v_{i+1}\}$, forming a path of length $n - 2$.
- Put w on track 2 at the far right, $p(w) = n$.

On track 1 we have $n - 2$ edges. For the cross-track neighbors:

- For each v_i, w is the nearest track-2 vertex to the right, so $\{v_i, w\}$ is a legal cross-track edge.
- There are $n - 1$ such vertices v_i.

Thus H has $(n - 2)$ edges along track 1 and $(n - 1)$ cross edges to w, for a total of

$$(n-2) + (n-1) = 2n - 3$$

edges. Taking $G = H$ shows that the bound is tight for every $n \ge 2$. □

Figure 15 shows the extremal construction for $n = 6$, and Figure 16 shows how each newly added vertex can contribute at most two new edges when we build such a graph from left to right in the global order.

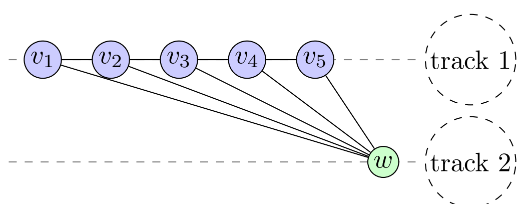

Figure 15: An extremal 2-track host graph on $n = 6$ vertices: a path $v_1 - \cdots - v_5$ on track 1 and a single vertex w on track 2, joined to every v_i. This has $(6 - 2) + (6 - 1) = 9 = 2n - 3$ edges.

5.4 Two low-degree vertices at the ends of the order

The proof of Theorem 5.6 naturally singles out the first and last vertices in the global order. For $k = 2$ they *always* have degree at most 2.

Lemma 5.7 (Endpoints have degree at most two)**.** *Let G be 2-track nearest-neighbor representable with layout (τ, p) on vertices $v_1, \dots, v_n$ in increasing p-order. Then*

$$\deg_G(v_1) \le 2 \quad \textit{and} \quad \deg_G(v_n) \le 2.$$

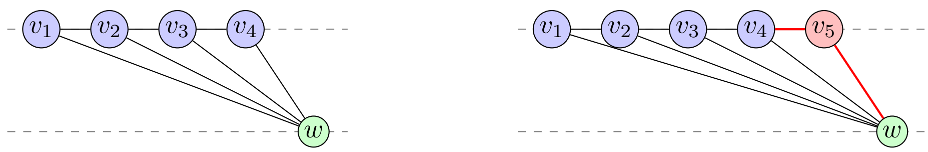

Figure 16: Incremental view of the extremal family. When we append a new vertex at the right end of the main track (here v_5), we can connect it to at most two new neighbors to its left: the previous endpoint of the path and the special vertex w on the other track. This contributes exactly two new edges, consistent with the bound $|E| \leq 2n - 3$.

Proof. By definition,

$$N_{\text{legal}}(v) = \{\text{pred}_1(v), \text{succ}_1(v), \text{pred}_2(v), \text{succ}_2(v)\} \setminus \{\text{nonexistent}\}.$$

For v_1 there is no vertex to the left on any track, so both $\text{pred}_1(v_1)$ and $\text{pred}_2(v_1)$ are nonexistent. The only possible legal neighbors of v_1 are its two successors, $\text{succ}_1(v_1)$ and $\text{succ}_2(v_1)$, one on each track if they exist. Thus $|N_{\text{legal}}(v_1)| \leq 2$, and since G is a subgraph of the host graph, $\deg_G(v_1) \leq |N_{\text{legal}}(v_1)| \leq 2$.

Symmetrically, v_n has no vertices to the right on any track, so both $\text{succ}_1(v_n)$ and $\text{succ}_2(v_n)$ are nonexistent, and its only possible legal neighbors are the two predecessors $\text{pred}_1(v_n)$ and $\text{pred}_2(v_n)$. Again $|N_{\text{legal}}(v_n)| \leq 2$, so $\deg_G(v_n) \leq 2$. □

In particular, every connected 2-track nearest-neighbor representable graph has at least two vertices of degree at most 2, sitting at the left and right ends of the global order. This is exactly what one needs to start an inductive "peeling" argument: delete v_1 or v_n, apply the edge bound to the remaining $(n-1)$-vertex graph, and then add back a vertex of degree at most 2.

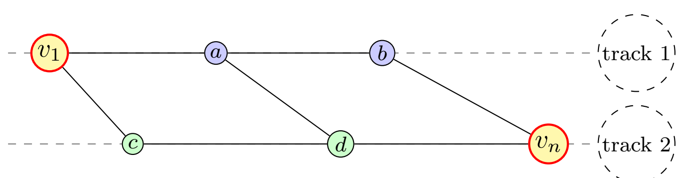

Figure 17: In a 2-track nearest-neighbor layout, the first vertex v_1 can only see one successor on each track, and the last vertex v_n can only see one predecessor on each track. Hence both have degree at most 2.

6 Examples: Cycles and Paths with One Extra Edge

We now look at explicit families of graphs and determine their track numbers, answering more concretely when $k = 1$ versus $k = 2$ is sufficient.

6.1 Cycles: C_n needs exactly two tracks

Theorem 6.1 (Track number of a cycle)**.** *For every integer $n \geq 3$, the cycle graph C_n is k-track nearest-neighbor representable for $k = 2$, but not for $k = 1$. In particular,*

$$\text{tn}(C_n) = 2.$$

Proof. Step 1: C_n is not representable with $k = 1$.

Assume, for contradiction, that C_n is 1-track nearest-neighbor representable. Then there exists a 1-track ordered layout (τ, p) such that every edge of C_n is legal.

Since $k = 1$, every vertex lies on track 1, and we can write the linear order as

$$v_1, v_2, \dots, v_n, \quad \text{where } p(v_i) = i.$$

On the single track, the only possible legal edges are between consecutive vertices:

$$E\big(H(\tau, p)\big) \subseteq \{\{v_i, v_{i+1}\} : 1 \le i \le n-1\},$$

so $H(\tau, p)$ is a forest (in fact a path) and contains no cycle. But C_n is a cycle, so it cannot be a subgraph of $H(\tau, p)$, contradicting nearest-neighbor representability.

Thus C_n is not 1-track representable.

Step 2: explicit 2-track layout for C_n.

Label the vertices of C_n as $v_1, \dots, v_n$ with edges

$$E(C_n) = \big\{\{v_i, v_{i+1}\} : 1 \le i \le n-1\big\} \cup \{\{v_n, v_1\}\}.$$

Global order. Set $p(v_i) = i$ for all i.

Track assignment. Use two tracks $\{1, 2\}$ and define

$$\tau(v_1) = \tau(v_n) = 1, \qquad \tau(v_i) = 2 \quad \text{for } 2 \le i \le n-1.$$

Thus track 1 has vertices $\{v_1, v_n\}$ and track 2 has $\{v_2, \dots, v_{n-1}\}$.

We now check that each edge of C_n is legal.

Interior edges on track 2. For $2 \le i \le n-2$, both v_i and v_{i+1} lie on track 2 and are consecutive there, so

$$\mathrm{succ}_2(v_i) = v_{i+1}, \quad \mathrm{pred}_2(v_{i+1}) = v_i.$$

Hence each $\{v_i, v_{i+1}\}$ with $2 \le i \le n-2$ is legal.

Edges $\{v_1, v_2\}$ and $\{v_{n-1}, v_n\}$. These involve one vertex on track 1 and one on track 2. Checking the nearest opposite-track neighbors shows:

- v_2 is the closest track-2 vertex to the right of v_1 and v_1 is the closest track-1 vertex to the left of v_2, so $\{v_1, v_2\}$ is legal.
- v_n is the closest track-1 vertex to the right of v_{n-1} and v_{n-1} is the closest track-2 vertex to the left of v_n, so $\{v_{n-1}, v_n\}$ is legal.

Edge $\{v_n, v_1\}$. Both endpoints lie on track 1 and are the only vertices there, so

$$\mathrm{succ}_1(v_1) = v_n, \quad \mathrm{pred}_1(v_n) = v_1,$$

and $\{v_n, v_1\}$ is legal.

Thus (τ, p) is a 2-track layout in which all edges of C_n are legal, so C_n is 2-track nearest-neighbor representable.

Combined with the non-representability for $k = 1$, we obtain $\mathrm{tn}(C_n) = 2$. □

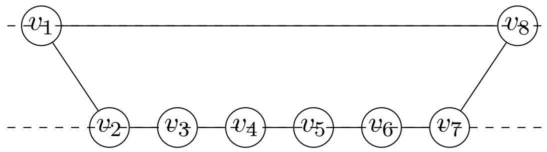

Figure 18: A 2-track layout of C_8 with v_1, v_8 on track 1 and $v_2, \dots, v_7$ on track 2.

6.2 A path plus one extra edge

We next show that a path with a single additional edge is always 2-track representable. This is a simple example of a graph that is *no longer* a linear forest but still only needs $k = 2$ tracks.

Theorem 6.2 (Paths with one extra edge)**.** *Let G be a connected simple undirected graph obtained as follows:*

- *$V = \{v_1, \dots, v_n\}$ for some $n \geq 2$,*
- *E contains all path edges $\{v_i, v_{i+1}\}$ for $i = 1, \dots, n-1$,*
- *and in addition a single extra edge $e^\star = \{v_a, v_b\}$ with $1 \leq a < b \leq n$.*

Then G is 2-track nearest-neighbor representable.

Figure 19 illustrates the construction in the case of an extra edge $\{v_3, v_7\}$. The same idea works when one or both endpoints of the extra edge are at the ends of the path. Since a 1-track nearest-neighbor representable graph is a linear forest (Remark 1.5) and our graphs contain a cycle, they cannot be 1-track representable. Combined with the construction above, this shows that their nearest-neighbor track number is exactly 2.

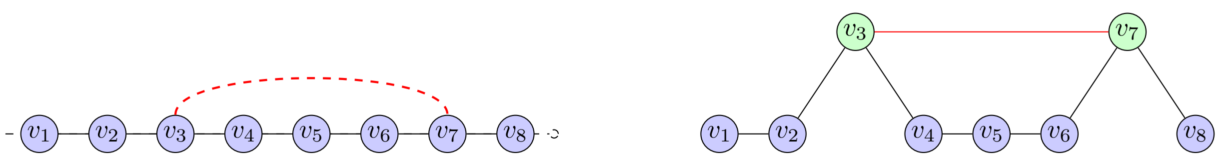

Figure 19: A path $v_1 - \cdots - v_8$ with an extra edge $\{v_3, v_7\}$. Left: the natural 1-track layout of the path; the extra edge (red, dashed) skips several vertices and is not a nearest-neighbor edge. Right: by moving v_3 and v_7 to track 2, we obtain a 2-track nearest-neighbor layout in which the extra edge (solid red) becomes legal while all path edges remain legal.

Proof. We keep the natural order $p(v_i) = i$ and modify only the track assignment.

Step 1: start with the path. For the bare path P_n with edges $\{v_i, v_{i+1}\}$, the layout

$$\tau^{(1)}(v_i) = 1 \quad \text{for all } i, \qquad p(v_i) = i,$$

is a 1-track nearest-neighbor layout: each $\{v_i, v_{i+1}\}$ connects consecutive vertices on track 1.

Step 2: move the extra-edge endpoints to track 2. Now add the extra edge $e^\star = \{v_a, v_b\}$ with $a < b$, keep the same global order p, and define a new track assignment τ by

$$\tau(v_i) = \begin{cases} 2, & \text{if } i \in \{a, b\}, \\ 1, & \text{otherwise.} \end{cases}$$

Thus only v_a and v_b are on track 2; every other vertex is on track 1.

We check that every edge of G is legal under (τ, p).

Step 3: path edges not incident with v_a or v_b. If $i \notin \{a-1, a, b-1, b\}$, then $\{v_i, v_{i+1}\}$ is not incident with v_a or v_b and both endpoints lie on track 1. They are consecutive in the global order, and no other track-1 vertex lies between them; hence

$$\mathrm{succ}_1(v_i) = v_{i+1}, \quad \mathrm{pred}_1(v_{i+1}) = v_i,$$

and $\{v_i, v_{i+1}\}$ is legal.

Step 4: path edges incident with v_a or v_b. Consider the edges $\{v_{a-1}, v_a\}$ and $\{v_a, v_{a+1}\}$ (when these indices exist). The neighbor of v_a immediately to the left on track 1 is v_{a-1} and immediately to the right is v_{a+1}, so

$$\mathrm{pred}_1(v_a) = v_{a-1} \text{ (if } a > 1), \quad \mathrm{succ}_1(v_a) = v_{a+1} \text{ (if } a < n).$$

Thus the edges $\{v_{a-1}, v_a\}$ and $\{v_a, v_{a+1}\}$ are legal.

The same argument applies to v_b: the nearest track-1 vertices to the left and right are v_{b-1} and v_{b+1} (when they exist), making $\{v_{b-1}, v_b\}$ and $\{v_b, v_{b+1}\}$ legal.

Step 5: the extra edge $\{v_a, v_b\}$. On track 2, v_a and v_b are the only vertices and $a < b$. Thus

$$\mathrm{succ}_2(v_a) = v_b, \qquad \mathrm{pred}_2(v_b) = v_a,$$

so $v_b \in N_{\mathrm{legal}}(v_a)$ and $v_a \in N_{\mathrm{legal}}(v_b)$. Hence the extra edge is legal.

Altogether, every edge of G is legal, so G is 2-track nearest-neighbor representable. □

7 Planarity for $k \leq 2$

Finally, we show that $k \leq 2$ imposes strong structural limitations: every such graph is planar.

Theorem 7.1 (Planarity for $k \leq 2$)**.** *Let G be a simple undirected graph that is (k)-track nearest-neighbor representable with $k \leq 2$. Then G is planar.*

Proof. We treat $k = 1$ and $k = 2$ separately.

Case $k = 1$. When there is only one track, each vertex has at most one predecessor and one successor, and every legal edge joins consecutive vertices in the single track order. Thus every connected component of G is a path (or an isolated vertex). Such a graph is a disjoint union of paths and isolated vertices, hence planar.

Case $k = 2$: reduction to host graphs. Fix a 2-track layout (τ, p) on V and let $H(\tau, p)$ be the host graph containing all legal edges. Since G is a subgraph of $H(\tau, p)$, it suffices to show that $H(\tau, p)$ is planar.

Place the vertices along a horizontal line in the global order $v_1, \ldots, v_n$ with $p(v_i) = i$, and write $\tau(v_i) \in \{1, 2\}$. We partition the edges of $H(\tau, p)$ into four classes:

- $E^{(1)}$: edges between consecutive vertices on track 1,
- $E^{(2)}$: edges between consecutive vertices on track 2,
- E_R: cross-track edges where each vertex is the nearest opposite-track neighbor to the right,

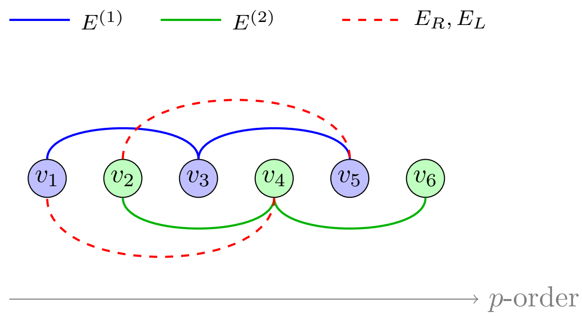

Figure 20: A schematic planar embedding for a 2-track host graph: edges on track 1 and right-going cross edges are drawn above the line, while edges on track 2 and left-going cross edges are drawn below.

- E_L: cross-track edges where each vertex is the nearest opposite-track neighbor to the left.

  Figure 20 schematically illustrates the planar drawing used in the proof.
  We draw:

- all vertices $v_1, \ldots, v_n$ on the x-axis at $(i, 0)$,
- all edges in $E^{(1)} \cup E_R$ as x-monotone arcs in the upper half-plane,
- all edges in $E^{(2)} \cup E_L$ as x-monotone arcs in the lower half-plane.

  It is a standard interval-order argument (we omit the routine details) that:

- edges in $E^{(1)}$ form a non-crossing family in the upper half-plane,
- edges in E_R form a non-crossing family in the upper half-plane,
- no edge in $E^{(1)}$ crosses any edge in E_R in the upper half-plane,
- by symmetry, the same holds for $E^{(2)}$ and E_L in the lower half-plane.

Since upper and lower half-plane edges only meet along the horizontal line at their endpoints, this gives a planar embedding of $H(\tau, p)$. Thus $H(\tau, p)$ is planar, and any subgraph G of it is planar as well. □

This planarity constraint, together with the degree and edge bounds proved earlier, will be exploited in Section 9 to design a recognition algorithm for 2-track nearest-neighbor layouts.

8 Complexity: Recognizing 2-Track Nearest-Neighbor Layouts

8.1 The decision problem 2TNN-Layout

We fix $k = 2$ throughout this section. Recall that a 2-track nearest-neighbor layout of a graph $G = (V, E)$ consists of:

- a track assignment $\tau : V \to \{1, 2\}$,
- and a bijection $p : V \to \{1, \ldots, |V|\}$,

such that every edge $\{u, v\} \in E$ is legal in the sense of Section 1: each endpoint is either the predecessor or successor of the other on some track (its own or the opposite track).

Definition 8.1 (2TNN-Layout)**.** The decision problem 2TNN-Layout is:

Input: A finite simple undirected graph $G = (V, E)$.

Question: Does there exist a 2-track ordered layout (τ, p) of G such that G is 2-track nearest-neighbor representable with respect to (τ, p), i.e.

$$\forall v \in V, \quad N_G(v) \subseteq N_{\text{legal}}(v)?$$

Equivalently, is $\text{tn}(G) \leq 2$ in the nearest-neighbor sense?

8.2 Membership in NP

Proposition 8.2 (2TNN-Layout is in NP)**.** *The problem* 2TNN-Layout *belongs to the complexity class NP.*

Proof. A certificate consists of a pair (τ, p) with $\tau : V \to \{1, 2\}$ and a bijection $p : V \to \{1, \ldots, n\}$. From (τ, p) we can compute all predecessors and successors along each track in linear time and thus the legal neighbor sets $N_{\text{legal}}(v)$. We then verify in $O(|E|)$ time that every edge of G is legal at both endpoints. Hence 2TNN-Layout is in NP. □

9 A Recognition Algorithm for Two-Track Layouts

Building on the complexity viewpoint of the previous section, we now describe a concrete decision procedure for $\text{tn}(G) \leq 2$.

By Proposition 3.2 we may work one connected component at a time, since $\text{tn}(G) = \max_i \text{tn}(G_i)$. Throughout this section, for a component H we write

$$n_H := |V(H)|, \qquad m_H := |E(H)|, \qquad \Delta(H) := \max_{v \in V(H)} \deg_H(v).$$

Our algorithm has two layers:

1. a *fast filter layer* of structural tests that in $O(n+m)$ time either certifies *YES* or *NO* for most graphs, and
2. a *backtracking layer* on the remaining "hard" cores, based on the four nearest-neighbor slots each vertex can use when $k = 2$.

We organize our recognition procedure into a sequence of structural "cuts" that explicitly realize the simple necessary conditions developed in the previous sections, followed by an exponential-time search on the remaining core.

9.1 Immediate decisions from size and density

Fix a connected component H.

Cut 0: tiny graphs. If $n_H \le 3$ then H has at most three edges and no K_4, and one easily writes down a 2-track nearest-neighbor layout by hand. We therefore treat

$$n_H \le 3 \quad \Longrightarrow \quad \mathrm{tn}(H) \le 2$$

as an *automatic YES*. This is checked in $O(1)$ per component.

Cut 1: edge-density upper bound. By Theorem 5.6, every 2-track nearest-neighbor host graph on n_H vertices has at most $2n_H - 3$ edges. Hence any component with $m_H > 2n_H - 3$ cannot have $\mathrm{tn}(H) \le 2$.

9.2 Linear-time structural filters

Once $n_H \ge 4$ and $m_H \le 2n_H - 3$, we apply three more structural cuts, each justified by earlier results.

Cut 2: maximum degree. By Lemma 5.2, a k-track nearest-neighbor layout satisfies $\Delta(H) \le 2k$. For $k \le 2$ this gives

$$\Delta(H) \le 4.$$

Thus if we ever see $\Delta(H) \ge 5$ we can immediately conclude $\mathrm{tn}(H) \ge 3$ and return *NO*. Computing all degrees takes $O(n_H + m_H)$ time.

Cut 3: K_4-subgraph. Theorem 5.1 shows that a 4-clique forces at least three tracks. So if H contains a K_4 we must reject:

$$K_4 \subseteq H \quad \Longrightarrow \quad \mathrm{tn}(H) \ge 3.$$

Given $\Delta(H) \le 4$ from Cut 2, K_4 can be detected in $O(n_H \Delta(H)^2) = O(n_H)$ time by intersecting neighbor sets.

Cut 4: planarity. By Theorem 7.1, any graph with $\mathrm{tn}(H) \le 2$ must be planar. We therefore run a linear-time planarity test (for example Hopcroft–Tarjan) and reject if H is nonplanar:

$$H \text{ nonplanar} \quad \Longrightarrow \quad \mathrm{tn}(H) \ge 3.$$

This is $O(n_H + m_H)$.

After Cuts 1–4 all remaining components satisfy simultaneously

$$n_H \ge 4, \quad m_H \le 2n_H - 3, \quad \Delta(H) \le 4, \quad H \text{ planar}, \quad K_4 \not\subseteq H.$$

9.3 Cheap YES cases

Before we resort to any backtracking, we also exploit explicit constructions from Section 5.

Cut 5: linear forests. If every component of H is a path or an isolated vertex (a linear forest), then $\mathrm{tn}(H) = 1$ by the discussion in Section 1, and we answer *YES*. In the connected case this is simply the test

$$\Delta(H) \le 2 \quad \text{and } H \text{ acyclic}.$$

Acyclicity and degrees can both be checked in $O(n_H + m_H)$.

Cut 6: one cycle with maximum degree 2. Section 5 proves that

- every cycle C_n has $\mathrm{tn}(C_n) = 2$ (Theorem 6.1), and
- any graph obtained from a path by adding one extra edge is also 2-track nearest-neighbor representable (Theorem 6.2).

These are precisely the connected graphs with

$$m_H = n_H, \quad \Delta(H) \leq 2,$$

i.e. graphs with as many edges as vertices and maximum degree at most two.

Thus connected components with

$$m_H = n_H \quad \text{and} \quad \Delta(H) \leq 2$$

are *automatic YES* and never reach the expensive search layer. Again, this test is $O(n_H + m_H)$.

(One can extend this family further if desired, using more structure theorems, but the above already covers the most common sparse unicyclic cases.)

9.4 Reducing to a bounded-degree planar core

After Cuts 0–6, only "genuinely complicated" components remain: planar, K_4-free graphs with $3 \leq \delta(H) \leq \Delta(H) \leq 4$ and $m_H \leq 2n_H - 3$ that are neither linear forests nor simple cycles nor paths-with-one-extra-edge.

At this point we shrink H to a smaller *core* $H^\star$ by deleting inessential leaves and degree-2 path vertices, in the spirit of standard kernelization.

- While H has a leaf v (degree 1), delete v and its incident edge. By the usual "attach-the-leaf-back" argument, this does not change whether a 2-track nearest-neighbor layout exists.
- While H has a degree-2 vertex v whose neighbors x, y are not adjacent, contract the path $x - v - y$ to a single edge xy. This preserves the existence of a layout as well: in any layout of the contracted graph, v can be reinserted along the track between x and y.

We call the result $H^\star$ the *two-track core* of H. It is planar, K_4-free, has $3 \leq \delta(H^\star) \leq \Delta(H^\star) \leq 4$, and can be computed in $O(n_H + m_H)$.

If $H^\star$ is empty, or a single vertex, or a single cycle, we already know how to layout H from previous cuts; otherwise, $H^\star$ is the instance on which we run the exponential search.

9.5 The backtracking layer

On $H^\star$ we now run a backtracking algorithm that is exact but exponential in the size of $H^\star$. The key observation is that for $k = 2$ every vertex v can use at most four legal nearest-neighbor positions:

$$\{\text{pred on track } 1,\ \text{succ on track } 1,\ \text{pred on track } 2,\ \text{succ on track } 2\}.$$

We encode these as four abstract *slots*

$$\mathsf{1L},\ \mathsf{1R},\ \mathsf{2L},\ \mathsf{2R},$$

and we search over assignments of incident edges to slots, subject to:

- at each vertex, at most one edge uses each slot (degree bound),
- at each edge $\{u, v\}$, the pair of slots chosen at u and v must be one of the eight compatible pairs coming from an actual predecessor/successor relation (same-track or cross-track), and
- the induced "to-the-left-of" constraints on vertices must stay acyclic, so that they can be realized by a global order p.

We maintain partial slot assignments and their implications as a standard finite-domain CSP, and we branch on a vertex with fewest available slots. Since $\deg(v) \leq 4$ and there are four slots, the local branching factor is at most $4! = 24$, and usually much smaller once constraints are propagated.

A complete, consistent assignment of slots yields a 2-track nearest-neighbor layout of $H^\star$ (and hence of H), while failure on all branches proves that no such layout exists. The correctness argument is routine and follows the same pattern as other layout/ordering CSPs; the details are omitted here for brevity.

In the worst case this backtracking layer runs in time

$$O\big(c^{n^\star}\big),$$

for some constant $c < 24$ and $n^\star = |V(H^\star)|$, multiplied by a polynomial factor for constraint propagation. This exponential behavior is expected: already the standard track-number recognition problem for fixed $k = 2$ is NP-complete (see Dujmović et al. [1, 2]).

9.6 Final algorithm and global complexity

We can now summarize the whole procedure.

Algorithm TwoTrackNN(G)

Input: finite simple graph $G = (V, E)$.
Output: 1 if G has a nearest-neighbor layout with at most two tracks, 0 otherwise.

1. Compute connected components $G_1, \ldots, G_r$ of G (time $O(|V| + |E|)$).
2. For each component G_i:
   (a) Let $n_i = |V(G_i)|$, $m_i = |E(G_i)|$. If $n_i \leq 3$, mark G_i as *YES* and continue.
   (b) If $m_i > 2n_i - 3$, return 0 (Cut 1).
   (c) Compute degrees. If $\Delta(G_i) \geq 5$, return 0 (Cut 2).
   (d) Test for a K_4; if found, return 0 (Cut 3).
   (e) Run a linear-time planarity test; if G_i is nonplanar, return 0 (Cut 4).
   (f) If G_i is a linear forest, or a cycle, or a path plus one extra edge (Cuts 5 and 6), mark it as *YES* and continue.
   (g) Otherwise compute its core $G_i^\star$ by repeatedly deleting leaves and suppressing degree-2 path vertices (Section 9.4).
   (h) Run the slot-based backtracking search on $G_i^\star$. If it fails, return 0; if it succeeds, mark G_i as *YES*.
3. If all components are marked *YES*, return 1.

Steps (1)–(2f) and the core reduction (2g) together take

$$O(|V| + |E|)$$

time.

References

[1] V. Dujmović, P. Morin, and D. R. Wood. Track layouts of graphs. *Discrete Mathematics & Theoretical Computer Science* 6(2):497–522, 2004.

[2] V. Dujmović and D. R. Wood. Track layouts, layered path decompositions, and leveled planarity. *Journal of Graph Theory*, to appear. Preprint available as arXiv:1506.09145.

[3] J. Hopcroft and R. Tarjan. Efficient planarity testing. *Journal of the ACM* 21(4):549–568, 1974.

[4] D. Eppstein. Queue number. Available at https://en.wikipedia.org/wiki/Queue_number.

A Failed BFS-Based Cut Attempts

During the design of the recognition algorithm in Section 9 it is tempting to look for extra "cheap" necessary conditions coming from the behavior of breadth-first search (BFS). In this appendix we record two such ideas that turned out to be unusable: one is too weak to rule out anything beyond the degree bound, and the other is simply false (even for very small 2-track nearest-neighbor graphs).

A.1 A too-weak bound: at most three new vertices per expansion

A first thought was that, in a 2-track nearest-neighbor layout, a BFS expansion might never discover more than two new vertices at a time. This is false in general (a vertex can have degree 4), but even the corrected statement

"every BFS expansion discovers at most three new vertices"

turns out to be too weak to serve as a useful cut.

Indeed, this bound is automatically implied by the local degree constraint $\Delta(G) \leq 4$ (Lemma 5.2). If we root a BFS at some vertex r and expand vertices one by one, then for any vertex $v \neq r$:

- one incident edge goes to the BFS parent of v and hence is not counted as "new",
- the remaining at most $\deg(v) - 1 \leq 3$ incident edges can lead to new vertices.

Thus the "≤ 3 new vertices per expansion" property holds in every 2-track nearest-neighbor graph and adds no information beyond the degree bound; it cannot filter any additional instances.

A.2 A false pattern condition: 2 new then 3 new

A more ambitious idea was to exploit the *sequence* of BFS layer sizes. Write $L_0, L_1, L_2, \dots$ for the BFS layers from some chosen root, where L_i is the set of vertices at distance i from the root and $|L_i|$ is the number of vertices *first* discovered at that layer.

The attempted condition was informally:

> *In a 2-track nearest-neighbor graph there should not be a BFS in which a layer with 3 new vertices is immediately preceded by a layer with 2 new vertices.*

The hope was to use this as a necessary condition: if *some* BFS on *some* root exhibits the pattern

$$|L_i| = 2, \quad |L_{i+1}| = 3,$$

then the graph would be ruled out as a 2-track nearest-neighbor candidate. However, the following tiny example shows that this pattern does occur in a perfectly valid 2-track nearest-neighbor graph.

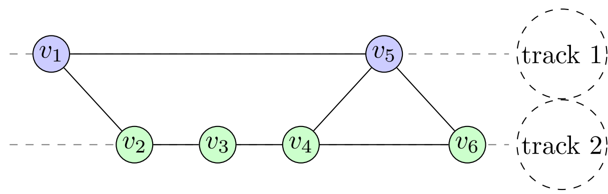

Figure 21: A 2-track nearest-neighbor graph in which a BFS from v_1 has $|L_0| = 1$, $|L_1| = 2$, and $|L_2| = 3$.

Example A.1 (A 2-then-3 BFS layering in a 2-track NN graph)*.* Consider the 2-track layout in Figure 21 with global order

$$v_1 < v_2 < v_3 < v_4 < v_5 < v_6,$$

track assignment

$$\tau(v_1) = \tau(v_5) = 1, \qquad \tau(v_2) = \tau(v_3) = \tau(v_4) = \tau(v_6) = 2,$$

and edges

$$\{v_1, v_5\},\ \{v_2, v_3\},\ \{v_3, v_4\},\ \{v_4, v_6\},\ \{v_1, v_2\},\ \{v_5, v_4\},\ \{v_5, v_6\}.$$

Every edge is between nearest neighbors on some track (either the same track or the opposite track), so this is a valid 2-track nearest-neighbor layout.

Run BFS from root v_1. The layers are:

$$L_0 = \{v_1\}, \quad L_1 = \{v_2, v_5\}, \quad L_2 = \{v_3, v_4, v_6\}.$$

Indeed:

- From $L_0 = \{v_1\}$ we discover its neighbors v_2 and v_5, so $|L_1| = 2$.
- From L_1 we discover, in total, the neighbors v_3 (via v_2) and v_4, v_6 (via v_5), giving $L_2 = \{v_3, v_4, v_6\}$ and $|L_2| = 3$.

Thus this graph exhibits the pattern

$$|L_0| = 1, \quad |L_1| = 2, \quad |L_2| = 3$$

for a perfectly legitimate BFS, contradicting the proposed rule that a layer of size 3 cannot be preceded by a layer of size 2 in any 2-track nearest-neighbor graph.

Example A.1 shows that such BFS layer-size patterns are too delicate to use as necessary conditions for 2-track nearest-neighbor layouts: even very small graphs that *do* admit such layouts can have BFS layer sequences of the form $2 \to 3$. For this reason we abandoned all BFS-based "layer-size" cuts and kept only the robust local structural conditions (degree bounds, planarity, and the absence of K_4) in the main algorithm.