Diseños en tracks, patrones prohibidos y cotas de grado

Axel Fridman

December 10, 2025

Abstract

Estudiamos disposiciones ordenadas de un grafo sobre k "tracks" (pistas) en las que las aristas están restringidas por reglas de vecinos más cercanos. Esta noción viene de la formulación del Investigathon en términos de patrones coloreados prohibidos sobre tríos de vértices. Hacemos precisa la correspondencia entre las dos formalizaciones, definimos los *diseños triplemente-legales en k tracks* y después investigamos cuántos tracks (es decir, "colores") hacen falta para un grafo dado. Probamos cotas de grado, resultados de planaridad para $k \leq 2$ y ejemplos estructurales como ciclos y grafos obtenidos a partir de un camino agregando una arista extra. En el camino resaltamos exactamente cuándo es posible $k \leq 2$ ("menos de tres colores").

Contents

1 Configuración básica: tracks, orden y vecinos legales

A lo largo de todo el trabajo, $G = (V, E)$ es un grafo simple finito no dirigido.

1.1 Diseños ordenados en k tracks

Arrancamos con la estructura puramente combinatoria de los tracks y un orden global.

Definition 1.1 (Diseño ordenado en k tracks)**.** Sea $k \in \mathbb{N}$. Un *diseño ordenado en k tracks* de un grafo $G = (V, E)$ es un par

$$(\tau, p)$$

compuesto por

- una *asignación de track*
$$\tau : V \to \{1, \dots, k\},$$
- y una biyección de *posiciones*
$$p : V \to \{1, \dots, |V|\}.$$

Escribimos $x < y$ si y sólo si $p(x) < p(y)$ y pensamos a todos los vértices puestos de izquierda a derecha de acuerdo a p.

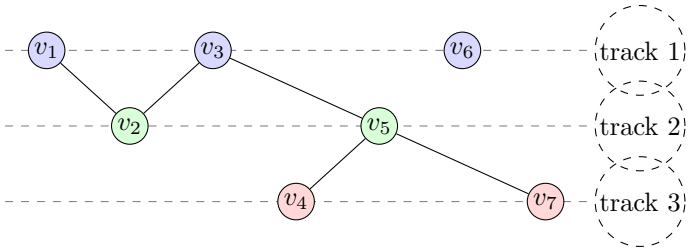

Figure 1: Ejemplo de un diseño ordenado en 3 tracks: el orden global es de izquierda a derecha, los colores indican la asignación de track τ.

Un ejemplo se muestra en la Figura 1.
Para cada track $t \in \{1, \ldots, k\}$ definimos el conjunto de vértices

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

ordenado por el $p(\cdot)$ creciente a lo largo de ese track.

Definition 1.2 (Predecesor y sucesor en un track)**.** Dado un diseño en k tracks (τ, p), un vértice $v \in V$ y un track t, definimos:

$$\text{pred}_t(v) := \text{el vértice en } V_t \text{ con mayor } p(\cdot) \text{ estrictamente menor que } p(v) \text{ (si existe)},$$
$$\text{succ}_t(v) := \text{el vértice en } V_t \text{ con menor } p(\cdot) \text{ estrictamente mayor que } p(v) \text{ (si existe)}.$$

Si un vértice así no existe, el predecesor/sucesor se dice "inexistente". Intuitivamente, son los vecinos más cercanos de v sobre el track t a la izquierda/derecha en el orden global.

Ver la Figura 2 para una vista pictórica sobre un solo track.

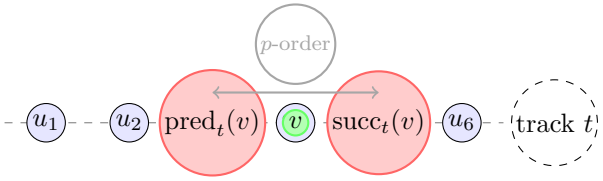

Figure 2: Predecesor y sucesor de un vértice v en un track t en el orden global.

Definition 1.3 (Conjunto de vecinos legales y grafo anfitrión)**.** Dado (τ, p), el *conjunto de vecinos legales* de $v \in V$ es

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

El *grafo anfitrión* correspondiente $H(\tau, p)$ sobre el conjunto de vértices V tiene conjunto de aristas

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

La Figura 3 muestra $N_{\text{legal}}(v)$ en tres tracks.

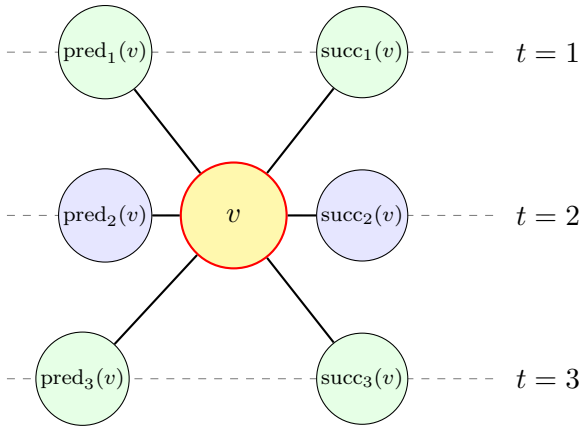

Figure 3: El conjunto de vecinos legales $N_{\text{legal}}(v)$.

Definition 1.4 (Grafo representable por vecinos más cercanos en k tracks)**.** Un grafo $G = (V, E)$ es *representable por vecinos más cercanos en k tracks* si existe un diseño ordenado en k tracks (τ, p) de V tal que

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

es decir, cada arista de G es una arista legal en el grafo anfitrión. Equivalentemente,

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

El menor k con esta propiedad (si existe) es el *número de tracks de vecinos más cercanos* de G.

La Figura 4 ilustra un grafo anfitrión típico cuando $k = 1$. La Figura 5 muestra un grafo anfitrión típico cuando $k = 2$.

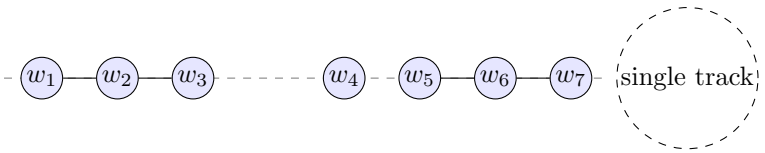

Figure 4: Con un solo track, el grafo anfitrión es una unión disjunta de caminos y vértices aislados (un bosque lineal).

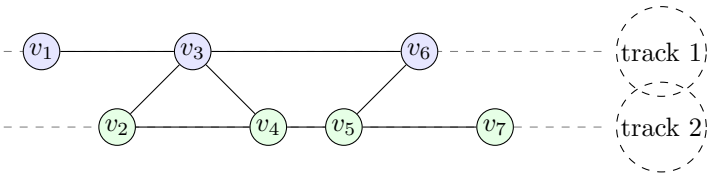

Figure 5: Con dos tracks, el grafo anfitrión puede tener tramos tipo camino en cada track más aristas cruzadas entre vecinos más cercanos en el otro track.

Remark 1.5 (Qué significa realmente $k = 1$). Cuando $k = 1$, todo vértice tiene a lo sumo un predecesor y un sucesor, de modo que el grafo anfitrión $H(\tau, p)$ es un subgrafo de un camino simple. Entonces cualquier grafo representable por vecinos más cercanos en un solo track es una

unión disjunta de caminos y vértices aislados (un *bosque lineal*). Esto va a ser importante cuando discutamos cuándo es posible "menos de dos colores" (tracks).

Remark 1.6 (Qué significa realmente $k = 2$). Cuando $k = 2$, cada vértice puede tener a lo sumo dos vecinos legales en su propio track (un predecesor y un sucesor) y a lo sumo dos vecinos legales en el otro track (de nuevo, un predecesor y un sucesor allí). Entonces cada vértice en el grafo anfitrión $H(\tau, p)$ tiene grado a lo sumo 4, y $H(\tau, p)$ se puede ver como dos capas tipo camino (una por track) con aristas cruzadas adicionales entre vecinos más cercanos de la otra capa, como en la Figura 5. En particular, cualquier grafo representable por vecinos más cercanos en 2 tracks es un subgrafo de un grafo anfitrión planar de este tipo "escalera". Esto será importante cuando discutamos cuándo es posible "menos de tres colores" (tracks).

2 De patrones coloreados prohibidos a restricciones en tríos

La formulación del Investigathon usa patrones *coloreados* prohibidos sobre triples de vértices. Ahora traducimos ese lenguaje a nuestro setting de diseños en k tracks.

2.1 Patrones coloreados en triples

Definition 2.1 (Patrón coloreado en tres vértices)**.** Fijamos el conjunto de índices $\{1, 2, 3\}$. Un *patrón* (a secas) es un par

$$(E_P, N_P),$$

donde $E_P, N_P \subseteq \{1, 2, 3\} \times \{1, 2, 3\}$ codifican aristas que deben estar presentes y aristas que deben estar ausentes entre las tres posiciones distinguidas.

Un *patrón coloreado* es un triple

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

donde (E_P, N_P) es un patrón como antes y $C_P \subseteq \{1, 2, 3\}$ es el conjunto de índices que deben quedar en una misma clase de color.

En el setting del Investigathon, el input es:

- un orden lineal $<$ sobre $V(G)$, y
- una partición ("coloración") $\mathcal{S} = \{S_1, \ldots, S_k\}$ de $V(G)$ en k clases de color.

Definition 2.2 (Realizar un patrón coloreado)**.** Dados $(\mathcal{S}, <)$ como arriba y un patrón coloreado $P = (E_P, N_P, C_P)$, un triple ordenado de vértices distintos

$$v_1 < v_2 < v_3$$

realiza P si:

(i) para todo $(i, j) \in E_P$ se cumple $\{v_i, v_j\} \in E(G)$;

(ii) para todo $(i, j) \in N_P$ se cumple $\{v_i, v_j\} \notin E(G)$;

(iii) todos los v_i con $i \in C_P$ pertenecen a una misma clase de color en $\mathcal{S}$.

Decimos que $(\mathcal{S}, <)$ *evita* P si no hay ningún triple $v_1 < v_2 < v_3$ que realice P.

Dada una familia finita Π de patrones coloreados, $(\mathcal{S}, <)$ *evita* Π si evita a cada $P \in \Pi$.

2.2 Los patrones del Investigathon

En nuestro caso, la familia prohibida Π consiste en dos patrones:

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

Intuitivamente:

- $P^{(1)}$ prohíbe triples $x < y < z$ con $\{x, z\} \in E(G)$ y x, y del mismo color;
- $P^{(2)}$ prohíbe triples $x < y < z$ con $\{x, z\} \in E(G)$ y y, z del mismo color.

Ver Figuras 6 y 7.

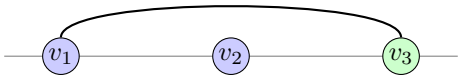

Figure 6: Patrón prohibido $P^{(1)}$: $v_1 < v_2 < v_3$ con $\{v_1, v_3\} \in E(G)$ y v_1, v_2 en la misma clase de color.

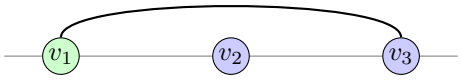

Figure 7: Patrón prohibido $P^{(2)}$: $v_1 < v_2 < v_3$ con $\{v_1, v_3\} \in E(G)$ y v_2, v_3 en la misma clase de color.

Una *solución con a lo sumo k colores* en el sentido del Investigathon es exactamente un par $(\mathcal{S}, <)$ con $|\mathcal{S}| \leq k$ que evita $P^{(1)}$ y $P^{(2)}$.

2.3 Tracks como clases de color

Dado un diseño en k tracks (τ, p), la asignación de tracks induce una partición en clases de color

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

y el orden global $<$ está definido por p.

A la inversa, dada una partición $\mathcal{S} = \{S_1, \ldots, S_k\}$ y un orden lineal $<$, obtenemos un diseño en k tracks etiquetando las partes $S_1, \ldots, S_k$ y fijando $\tau(v) = i$ para $v \in S_i$, y tomando un p cualquiera biyección compatible con $<$.

Así, hay una correspondencia uno a uno entre:

- soluciones $(\mathcal{S}, <)$ con a lo sumo k colores, y
- diseños ordenados en k tracks (τ, p).

2.4 Formulación como restricción en tríos

Reexpresamos ahora la evitación de $P^{(1)}$ y $P^{(2)}$ de forma compacta.

Lemma 2.3 (Patrones coloreados vs. restricción de tríos)**.** *Sea $G = (V, E)$ un grafo y sea (τ, p) un diseño ordenado en k tracks con el orden asociado $<$ y la partición inducida $\mathcal{S}$ como arriba. Entonces son equivalentes:*

(1) $(\mathcal{S}, <)$ evita ambos patrones coloreados $P^{(1)}$ y $P^{(2)}$.

(2) Para todo trío $x, y, z \in V$ con

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

se cumple

$$\tau(y) \neq \tau(x) \quad y \quad \tau(y) \neq \tau(z).$$

Proof. (1) $\Rightarrow$ (2): Supongamos que existen x, y, z con $p(x) < p(y) < p(z)$ y $\{x, z\} \in E(G)$ tales que $\tau(y) = \tau(x)$ o $\tau(y) = \tau(z)$.

Escribimos $v_1 := x$, $v_2 := y$, $v_3 := z$. Entonces $\{v_1, v_3\} \in E(G)$. Si $\tau(y) = \tau(x)$, entonces v_1 y v_2 están en la misma clase de color y el trío (v_1, v_2, v_3) realiza $P^{(1)}$, contradiciendo que se evita $P^{(1)}$. Si $\tau(y) = \tau(z)$, entonces (v_1, v_2, v_3) realiza $P^{(2)}$, otra contradicción. Por lo tanto, $\tau(y)$ es distinto tanto de $\tau(x)$ como de $\tau(z)$.

(2) $\Rightarrow$ (1): Asumamos que vale (2). Supongamos que algún triple $v_1 < v_2 < v_3$ realiza $P^{(1)}$. Entonces $\{v_1, v_3\} \in E(G)$ y v_1, v_2 están en la misma clase de color, o sea $\tau(v_1) = \tau(v_2)$, lo que contradice (2) con $(x, y, z) = (v_1, v_2, v_3)$. Un argumento idéntico, permutando los roles de las posiciones $\{1, 2\}$ y $\{2, 3\}$, muestra que ningún triple realiza $P^{(2)}$ tampoco. □

Definition 2.4 (Diseño triplemente-legal en k tracks)**.** Un diseño ordenado en k tracks (τ, p) de G es *triplemente-legal* si para todo $x, y, z \in V$ con

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

se cumple

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

Por el Lema 2.3, los diseños triplemente-legales son exactamente los diseños que corresponden a soluciones del Investigathon que evitan $P^{(1)}$ y $P^{(2)}$.

La Figura 8 contrasta un triple prohibido y dos triples legales, uno usando menos colores que el otro.

Remark 2.5 (Vecinos más cercanos vs. triplemente-legal). La representabilidad por vecinos más cercanos es una condición *más fuerte* que ser triplemente-legal: en un diseño por vecinos más cercanos, cada arista debe ser entre vecinos más cercanos en algún track; en un diseño triplemente-legal sólo prohibimos ciertos patrones coloreados en triples. En la próxima sección mostramos que los diseños de vecinos más cercanos satisfacen automáticamente la restricción de tríos.

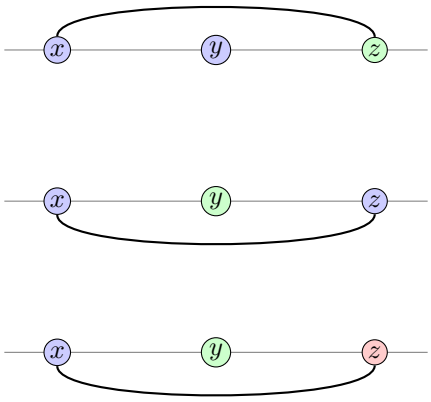

Figure 8: Arriba: un triple prohibido donde y comparte track (color) con x. Medio: un triple legal donde x comparte track (color) con z. Abajo: un triple legal donde el vértice del medio está en un tercer track.

3 Número cromático vs. "colores" basados en tracks

El índice de track $\tau(v)$ se comporta un poco como un color, pero *no* es una coloración propia en el sentido clásico (se permiten aristas dentro de un mismo track). Sólo necesitaremos dos hechos estándar del colorido clásico para la analogía:

- Si $G_1, \ldots, G_r$ son las componentes conexas de G, entonces
$$\chi(G) \;=\; \max_{1 \le i \le r} \chi(G_i).$$
- Si G contiene un clique de tamaño r, entonces $\chi(G) \ge r$; en particular un K_4 fuerza al menos cuatro colores.

En el setting de vecinos más cercanos, las etiquetas de track juegan el rol de "colores", y vamos a ver análogos de ambas afirmaciones para el número de tracks.

3.1 Número de tracks y componentes conexas

Definition 3.1 (Número de tracks de vecinos más cercanos)**.** El *número de tracks de vecinos más cercanos* de un grafo G, denotado por $\mathrm{tn}(G)$, es el mínimo $k \in \mathbb{N}$ tal que G es representable por vecinos más cercanos en k tracks.

Proposition 3.2 (Número de tracks y componentes conexas)**.** *Sean $G_1, \ldots, G_r$ las componentes conexas de G. Entonces*
$$\mathrm{tn}(G) \;=\; \max_{1 \le i \le r} \mathrm{tn}(G_i).$$

Remark 3.3. La Proposición 3.2 es directamente análoga al hecho clásico de que $\chi(G) = \max_i \chi(G_i)$ para el número cromático: en ambos settings, la cantidad de "colores" que se necesitan para todo el grafo es el máximo sobre sus componentes conexas.

Proof. Cota inferior. Cualquier diseño que prueba que G es representable por vecinos más cercanos en k tracks se restringe a un diseño para cada G_i, de modo que $\mathrm{tn}(G_i) \le k$. Tomando el mínimo sobre todos esos k se obtiene $\max_i \mathrm{tn}(G_i) \le \mathrm{tn}(G)$.

Cota superior. Sea $k_i = \mathrm{tn}(G_i)$ y $k := \max_i k_i$. Para cada componente G_i elegimos un diseño en k_i tracks (τ_i, p_i) que pruebe la representabilidad por vecinos más cercanos. Renombramos los tracks de forma que τ_i tome valores en $\{1, \ldots, k\}$ (simplemente no usamos todos los labels cuando $k_i < k$).

Ahora ponemos las componentes una detrás de la otra en el orden global: definimos p apilando los órdenes $p_1, \ldots, p_r$ con rangos disjuntos, y fijamos $\tau(v) = \tau_i(v)$ para $v \in V(G_i)$. Esto da un diseño en k tracks (τ, p) de G en el que cada arista dentro de cada componente sigue siendo legal. No hay aristas entre componentes, así que no hay nada más que chequear. Por lo tanto, G es representable por vecinos más cercanos en k tracks y $\mathrm{tn}(G) \leq k$. □

La Figura 9 muestra cómo los diseños de tracks para componentes diferentes se pueden apilar en el orden global reutilizando el mismo conjunto de labels de track.

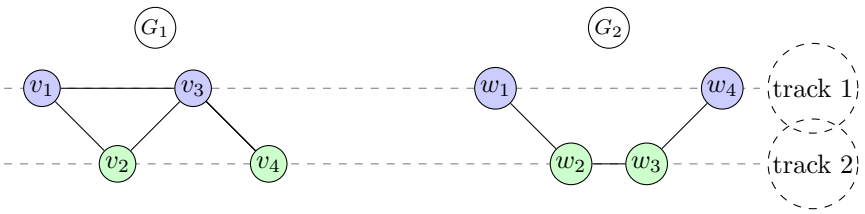

Figure 9: Los diseños para distintas componentes comparten los mismos labels de track. El orden global apila las componentes una tras otra.

Remark 3.4 (Cuándo alcanza con menos de 2 tracks). De la Sección 1, un grafo anfitrión con 1 track es siempre una unión disjunta de caminos y vértices aislados (un bosque lineal). Entonces

$$\mathrm{tn}(G) = 1 \quad \Longleftrightarrow \quad G \text{ es un bosque lineal,}$$

y todo grafo que contenga un ciclo requiere al menos dos tracks. La Figura 10 ilustra por qué un ciclo no puede vivir en un solo track.

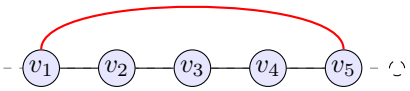

Figure 10: En un solo track sólo pueden ser legales aristas entre vértices consecutivos. La arista $\{v_1, v_5\}$ necesaria para completar un ciclo salta tres vértices y no se puede realizar.

4 Los diseños de vecinos más cercanos cumplen la restricción de tríos

Ahora conectamos la condición de vecinos más cercanos con la restricción de tríos del Lema 2.3.

Lemma 4.1 (Restricción de tríos para diseños de vecinos más cercanos)**.** *Sea (τ, p) un diseño ordenado en k tracks de un grafo $G = (V, E)$ y supongamos que G es representable por vecinos más cercanos en k tracks respecto de (τ, p), es decir*

$$N_G(v) \subseteq N_{\mathrm{legal}}(v) \quad \textit{para todo } v \in V.$$

Sean $x, y, z \in V$ tales que

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

y supongamos que $\{x, z\} \in E$. Entonces

$$\tau(y) \neq \tau(x) \quad y \quad \tau(y) \neq \tau(z).$$

En palabras: cualquier vértice estrictamente entre x y z en el orden global debe estar en un track distinto tanto de $\tau(x)$ como de $\tau(z)$.

La Figura 11 muestra la situación del Lema 4.1.

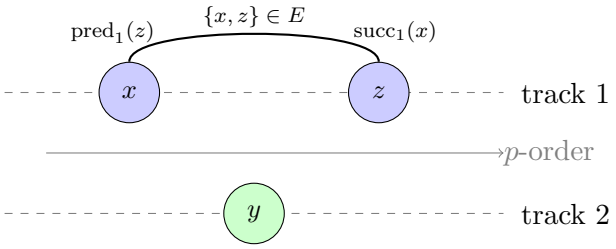

Figure 11: Si $\{x, z\}$ es una arista que viene de vecinos más cercanos en el track 1, entonces no puede haber otro vértice en el track 1 entre ellos en el orden global. Por lo tanto cualquier vértice del medio y debe vivir en otro track.

Proof. Como $\{x, z\} \in E$ y G es representable por vecinos más cercanos, tenemos

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

Paso 1: z es el sucesor de x en el track $\tau(z)$.

Por definición de $N_{\text{legal}}(x)$, existe un track t tal que

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

Si $z = \text{pred}_t(x)$ entonces $p(z) < p(x)$, contradiciendo $p(x) < p(z)$; luego

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

Además, $\text{succ}_t(x)$ está en el track t, así que $\tau(z) = t$ y

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

Por definición de sucesor, no hay vértices w en el track $\tau(z)$ con posición global estrictamente entre $p(x)$ y $p(z)$, es decir

$$\text{no hay } w \text{ tal que } \tau(w) = \tau(z) \text{ y } p(x) < p(w) < p(z).$$

Como $p(x) < p(y) < p(z)$, no puede pasar que $\tau(y) = \tau(z)$.

Paso 2: x es el predecesor de z en el track $\tau(x)$.

Análogamente, como $x \in N_{\text{legal}}(z)$ existe un track s tal que

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

Como $p(x) < p(z)$, x no puede ser sucesor de z, así que

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

Luego $\tau(x) = s$ y

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

De nuevo por definición de predecesor, no hay vértices w en el track $\tau(x)$ con $p(x) < p(w) < p(z)$. En particular, $\tau(y) \neq \tau(x)$.

Combinando ambos pasos, obtenemos $\tau(y) \neq \tau(x)$ y $\tau(y) \neq \tau(z)$. □

Corollary 4.2. *Todo diseño de vecinos más cercanos en k tracks (τ, p) es un diseño triplemente-legal en k tracks. En particular, cualquier solución de vecinos más cercanos evita automáticamente los patrones del Investigathon $P^{(1)}$ y $P^{(2)}$.*

Proof. Aplicar el Lema 4.1 y luego el Lema 2.3. □

5 Cotas de grado y restricciones de cliques

Ahora cuantificamos cómo el número de tracks k controla los posibles grados y cliques en un grafo representable por vecinos más cercanos. Esto apunta a una forma natural de *acotar por abajo* la cantidad de "colores" (tracks) necesarios.

5.1 Un K_4 fuerza al menos tres tracks

Theorem 5.1 (Un clique de tamaño 4 fuerza $k \geq 3$)**.** *Sea $G = (V, E)$ un grafo representable por vecinos más cercanos en k tracks para algún $k \in \mathbb{N}$. Si G contiene un clique de tamaño 4, entonces $k \geq 3$. Equivalentemente, ningún diseño en 1 o 2 tracks puede representar un 4-clique.*

Proof. Sea $H \subseteq G$ un subgrafo isomorfo a K_4 con conjunto de vértices $\{a, b, c, d\}$. Sea (τ, p) un diseño ordenado en k tracks que pruebe la representabilidad por vecinos más cercanos de G.

Renombramos a, b, c, d de forma que

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

Como H es completo, cada par entre $\{a, b, c, d\}$ es una arista.

Usando el Lema 4.1, aplicamos la restricción de tríos a todos los triples (x, y, z) con $x < y < z$ donde $\{x, z\}$ es una arista (siempre cierto en K_4). Los triples relevantes y sus consecuencias son:

$$\tau(b) \neq \tau(a), \quad \tau(b) \neq \tau(c), \qquad \text{a partir de } (a, b, c) \text{ y la arista } \{a, c\}, \tag{1}$$
$$\tau(b) \neq \tau(a), \quad \tau(b) \neq \tau(d), \qquad \text{a partir de } (a, b, d) \text{ y la arista } \{a, d\}, \tag{2}$$
$$\tau(c) \neq \tau(a), \quad \tau(c) \neq \tau(d), \qquad \text{a partir de } (a, c, d) \text{ y la arista } \{a, d\}, \tag{3}$$
$$\tau(c) \neq \tau(b), \quad \tau(c) \neq \tau(d), \qquad \text{a partir de } (b, c, d) \text{ y la arista } \{b, d\}. \tag{4}$$

De (1) y (2) vemos que

$$\tau(b) \notin \{\tau(a), \tau(c), \tau(d)\},$$

y de (3) y (4) que

$$\tau(c) \notin \{\tau(a), \tau(b), \tau(d)\}.$$

Supongamos $k \leq 2$. Entonces los tracks son $\{1, 2\}$. Sin pérdida de generalidad, tomemos $\tau(a) = 1$.

Caso 1: $\tau(d) = 1$. Entonces (2) implica $\tau(b) \neq \tau(a)$ y $\tau(b) \neq \tau(d)$, así que $\tau(b) \neq 1$ y por lo tanto $\tau(b) = 2$. Análogamente, (3) implica $\tau(c) \neq 1$, luego $\tau(c) = 2$. Pero entonces (1) exige $\tau(b) \neq \tau(c)$, imposible.

Caso 2: $\tau(d) = 2$. Entonces (2) dice que $\tau(b) \neq 1$ y $\tau(b) \neq 2$, lo cual es imposible con sólo dos tracks.

En todos los casos llegamos a una contradicción, así que $k \geq 3$. □

La Figura 12 muestra un K_4 cuyos vértices no pueden asignarse a sólo dos tracks sin violar el Lema 4.1.

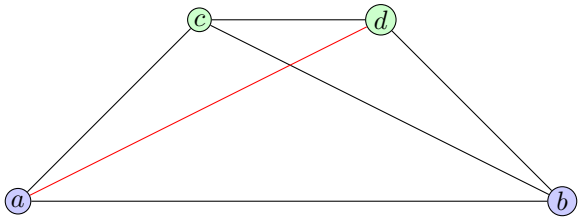

Figure 12: Un K_4 dibujado con dos "colores de track". Ningún orden de a, b, c, d puede evitar crear un triple $x < y < z$ con $\{x, z\} \in E$ donde y comparta track con x o z, así que hacen falta al menos tres tracks.

5.2 Una cota de grado: $\Delta(G) \leq 2k$

Lemma 5.2 (Cota de grado para diseños en k tracks)**.** *Sea $G = (V, E)$ un grafo simple no dirigido representable por vecinos más cercanos en k tracks. Es decir, existe un diseño (τ, p) con*

$$\tau : V \to \{1, \ldots, k\}, \qquad p : V \to \{1, \ldots, |V|\}$$

tal que cada arista es legal:

$$\forall \{u, v\} \in E, \quad v \in N_{\text{legal}}(u) \quad \textit{(equivalentemente } u \in N_{\text{legal}}(v)\textit{)}.$$

Entonces para todo vértice $v \in V$ se cumple

$$\deg_G(v) \leq 2k.$$

En particular, el grado máximo $\Delta(G)$ satisface

$$\Delta(G) \leq 2k.$$

Proof. Fijamos $v \in V$. Para cada track t, sea $V_t = \{x : \tau(x) = t\}$ y definimos $\text{pred}_t(v), \text{succ}_t(v)$ como antes. El conjunto de vecinos legales es

$$N_{\text{legal}}(v) = \{\text{pred}_1(v), \text{succ}_1(v), \ldots, \text{pred}_k(v), \text{succ}_k(v)\} \setminus \{\text{inexistente}\}.$$

Por representabilidad por vecinos más cercanos,

$$N_G(v) \subseteq N_{\text{legal}}(v),$$

y entonces

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

Para cada track t, a lo sumo dos vértices pueden aparecer en $N_{\text{legal}}(v)$ provenientes del track t, a saber $\text{pred}_t(v)$ y $\text{succ}_t(v)$ si existen. Definimos

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

entonces $|S_t(v)| \leq 2$ y

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

Por lo tanto,

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

Así, $\deg_G(v) \leq 2k$ para todo v, y tomando el máximo se obtiene $\Delta(G) \leq 2k$. □

Corollary 5.3 (Cota inferior de tracks basada en el grado)**.** *Si G es representable por vecinos más cercanos en k tracks, entonces*

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

La Figura 13 muestra un vértice de grado 5, lo que fuerza al menos tres tracks por el Corolario 5.3.

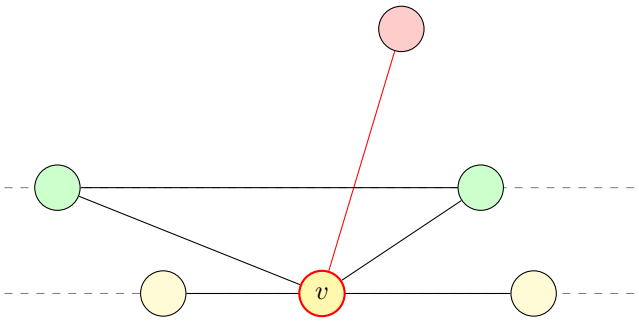

Figure 13: Un vértice de grado 5: por $\Delta(G) \leq 2k$, esto implica $k \geq 3$.

Proof. Sale directamente del Lema 5.2. □

5.3 Cota de aristas para dos tracks: a lo sumo $2n - 3$

Hasta ahora vimos que un diseño en k tracks para vecinos más cercanos fuerza $\Delta(G) \leq 2k$ (Lema 5.2). Para $k = 2$ podemos decir mucho más: no sólo el grado máximo es a lo sumo 4, sino que el *número total de aristas* es a lo sumo $2n - 3$ para un grafo de n vértices. Esto coincide con la cota extremal conocida para grafos outerplanos.

Trabajamos respecto de un diseño fijo en 2 tracks (τ, p) y su grafo anfitrión $H(\tau, p)$.

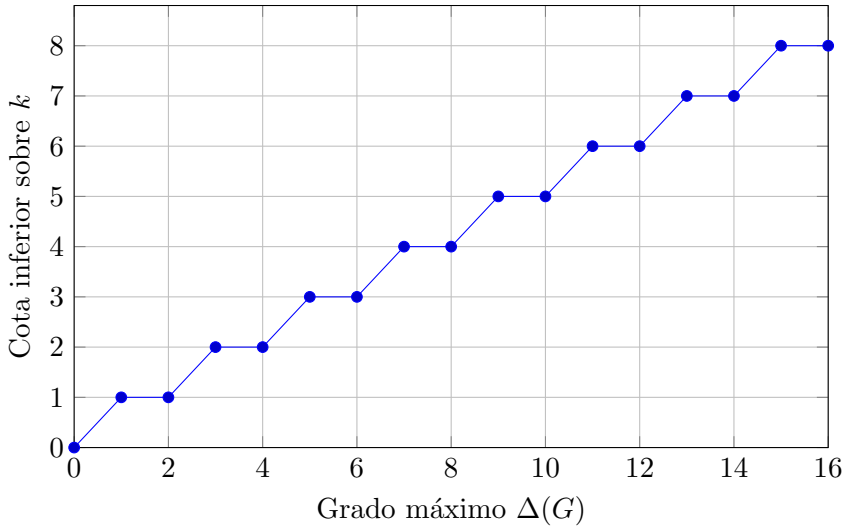

Figure 14: Cota inferior sobre el número de tracks: $k \geq \lceil \Delta(G)/2 \rceil$. En particular, $\Delta(G) \geq 5$ fuerza $k \geq 3$.

Definition 5.4 (Grado hacia la izquierda)**.** Dado un orden lineal $v_1, \ldots, v_n$ de $V(G)$ inducido por p, el *grado hacia la izquierda* de v_i se define como

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

Equivalentemente, $\deg^-(v_i)$ cuenta los vecinos de v_i estrictamente a su izquierda en el orden global.

Cada arista $\{u, v\}$ se cuenta exactamente una vez en la suma de grados hacia la izquierda, en su extremo derecho.

Lemma 5.5 (A lo sumo un predecesor por track)**.** *Sea G representable por vecinos más cercanos en k tracks respecto de (τ, p), y sea $v \in V(G)$. Entonces para cada track $t \in \{1, \ldots, k\}$, v tiene a lo sumo un vecino en el track t a su izquierda. En particular,*

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

para todo vértice v.

Proof. Fijamos v y un track t. Por definición, el único candidato a vecino a la izquierda de v en el track t que puede ser legal en el grafo anfitrión es $\mathrm{pred}_t(v)$ (si existe). No pueden existir dos vecinos distintos x, y en el track t con $p(x) < p(y) < p(v)$ y ambos adyacentes a v, porque sólo el más cercano a la izquierda puede ser un predecesor legal en ese track.

Por lo tanto, en cada track t hay a lo sumo un vecino de v a la izquierda. Sumando sobre los k tracks obtenemos $\deg^-(v) \leq k$. □

Theorem 5.6 (Cota de aristas para diseños en dos tracks)**.** *Sea G un grafo simple finito de n vértices representable por vecinos más cercanos en 2 tracks. Entonces*

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

Además, para todo $n \geq 2$ existe un grafo de este tipo con exactamente $2n - 3$ aristas.

Proof. Sea (τ, p) un diseño en 2 tracks y sea $H = H(\tau, p)$ su grafo anfitrión. Como G es subgrafo de H, alcanza con demostrar $|E(H)| \leq 2n - 3$.

Escribimos $V = \{v_1, \ldots, v_n\}$ en orden creciente de p, de forma que $p(v_i) = i$. Cada arista de H tiene un único extremo derecho, entonces

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

Ahora:

- Para v_1, no hay vértices a la izquierda, entonces $\deg_H^-(v_1) = 0$.
- Para v_2, el único vértice a la izquierda es v_1, así que $\deg_H^-(v_2) \leq 1$ en cualquier grafo simple.
- Para cada $i \geq 3$, el Lema 5.5 con $k = 2$ da $\deg_H^-(v_i) \leq 2$.

Por lo tanto,

$$|E(H)| = \sum_{i=1}^{n} \deg_H^-(v_i) \;\leq\; 0 + 1 + 2(n-2) = 2n - 3.$$

Como G es subgrafo de H, también vale $|E(G)| \leq |E(H)| \leq 2n - 3$.

Para ver que la cota es ajustada, fijemos $n \geq 2$ y construyamos un diseño en 2 tracks sobre los vértices $v_1, \ldots, v_{n-1}, w$ como sigue:

- Ponemos $v_1, \ldots, v_{n-1}$ en el track 1 en el orden $p(v_i) = i$ y conectamos todos los pares consecutivos $\{v_i, v_{i+1}\}$, formando un camino de longitud $n - 2$.
- Ponemos w en el track 2 a la extrema derecha, $p(w) = n$.

En el track 1 tenemos $n - 2$ aristas. Para los vecinos de track cruzado:

- Para cada v_i, w es el vértice más cercano en el track 2 a la derecha, luego $\{v_i, w\}$ es una arista cruzada legal.
- Hay $n - 1$ vértices v_i de este tipo.

Así, H tiene $(n - 2)$ aristas en el track 1 y $(n - 1)$ aristas cruzadas a w, sumando en total

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

aristas. Tomando $G = H$ vemos que la cota es ajustada para todo $n \geq 2$. □

La Figura 15 muestra la construcción extremal para $n = 6$, y la Figura 16 muestra cómo cada vértice nuevo puede aportar a lo sumo dos aristas nuevas cuando construimos un grafo de este tipo de izquierda a derecha en el orden global.

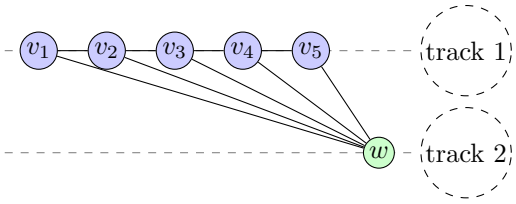

Figure 15: Un grafo anfitrión extremal en 2 tracks con $n = 6$ vértices: un camino $v_1 - \cdots - v_5$ en el track 1 y un único vértice w en el track 2, unido a todos los v_i. Esto tiene $(6-2)+(6-1) = 9 = 2n-3$ aristas.

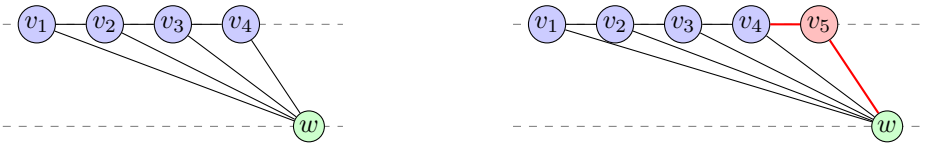

Figure 16: Vista incremental de la familia extremal. Cuando agregamos un vértice nuevo en el extremo derecho del track principal (acá v_5), podemos conectarlo a lo sumo con dos vecinos nuevos a su izquierda: el extremo anterior del camino y el vértice especial w del otro track. Esto aporta exactamente dos aristas nuevas, consistente con la cota $|E| \leq 2n - 3$.

5.4 Dos vértices de bajo grado en los extremos del orden

La prueba del Teorema 5.6 señala naturalmente a los primeros y últimos vértices en el orden global. Para $k = 2$ siempre tienen grado a lo sumo 2.

Lemma 5.7 (Extremos con grado a lo sumo dos)**.** *Sea G representable por vecinos más cercanos en 2 tracks con diseño (τ, p) sobre los vértices $v_1, \ldots, v_n$ en orden creciente de p. Entonces*

$$\deg_G(v_1) \leq 2 \quad y \quad \deg_G(v_n) \leq 2.$$

Proof. Por definición,

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

Para v_1 no hay vértices a la izquierda en ningún track, así que tanto $\text{pred}_1(v_1)$ como $\text{pred}_2(v_1)$ son inexistentes. Los únicos vecinos legales posibles de v_1 son sus dos sucesores, $\text{succ}_1(v_1)$ y $\text{succ}_2(v_1)$, a lo sumo uno en cada track. Entonces $|N_{\text{legal}}(v_1)| \leq 2$, y como G es un subgrafo del grafo anfitrión, $\deg_G(v_1) \leq |N_{\text{legal}}(v_1)| \leq 2$.

Simétricamente, v_n no tiene vértices a la derecha en ningún track, entonces $\text{succ}_1(v_n)$ y $\text{succ}_2(v_n)$ son inexistentes, y sus únicos vecinos legales posibles son los dos predecesores $\text{pred}_1(v_n)$ y $\text{pred}_2(v_n)$. De nuevo, $|N_{\text{legal}}(v_n)| \leq 2$, luego $\deg_G(v_n) \leq 2$. □

En particular, todo grafo conexo representable por vecinos más cercanos en 2 tracks tiene al menos dos vértices de grado a lo sumo 2, ubicados en los extremos izquierdo y derecho del orden global. Esto es exactamente lo que hace falta para arrancar un argumento inductivo de "pelado": eliminar v_1 o v_n, aplicar la cota de aristas al grafo restante de $(n-1)$ vértices y luego agregar de vuelta un vértice de grado a lo sumo 2.

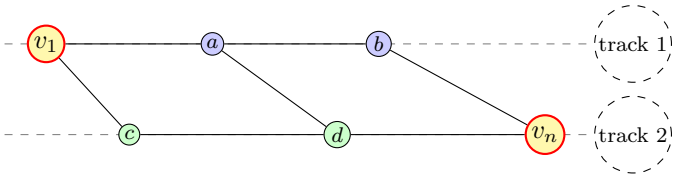

Figure 17: En un diseño de vecinos más cercanos en 2 tracks, el primer vértice v_1 sólo puede ver un sucesor en cada track, y el último vértice v_n sólo puede ver un predecesor en cada track. Por eso ambos tienen grado a lo sumo 2.

6 Ejemplos: ciclos y caminos con una arista extra

Ahora miramos familias concretas de grafos y determinamos sus números de tracks, respondiendo más en concreto cuándo alcanza con $k = 1$ versus cuándo hace falta $k = 2$.

6.1 Ciclos: C_n necesita exactamente dos tracks

Theorem 6.1 (Número de tracks de un ciclo)**.** *Para todo entero $n \geq 3$, el ciclo C_n es representable por vecinos más cercanos en $k = 2$ tracks, pero no en $k = 1$. En particular,*

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

Proof. Paso 1: C_n no es representable con $k = 1$.

Supongamos, para llegar a una contradicción, que C_n es representable por vecinos más cercanos en 1 track. Entonces existe un diseño (τ, p) en 1 track tal que todas las aristas de C_n son legales.

Como $k = 1$, todos los vértices están en el track 1, y podemos escribir el orden lineal como

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

En el único track, las únicas aristas legales posibles son entre vértices consecutivos:

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

así que $H(\tau, p)$ es un bosque (de hecho, un camino) y no contiene ciclos. Pero C_n es un ciclo, así que no puede ser subgrafo de $H(\tau, p)$, contradiciendo la representabilidad por vecinos más cercanos.

Luego C_n no es representable con $k = 1$.

Paso 2: diseño explícito en 2 tracks para C_n.

Etiquetamos los vértices de C_n como $v_1, \ldots, v_n$ con aristas

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

Orden global. Tomamos $p(v_i) = i$ para todo i.

Asignación de tracks. Usamos dos tracks $\{1, 2\}$ y definimos

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

Así, el track 1 tiene los vértices $\{v_1, v_n\}$ y el track 2 tiene $\{v_2, \ldots, v_{n-1}\}$.

Chequeamos ahora que cada arista de C_n es legal.

Aristas interiores en el track 2. Para $2 \leq i \leq n-2$, tanto v_i como v_{i+1} están en el track 2 y son consecutivos allí, entonces

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

Por lo tanto cada $\{v_i, v_{i+1}\}$ con $2 \leq i \leq n-2$ es legal.

Aristas $\{v_1, v_2\}$ *y* $\{v_{n-1}, v_n\}$. Estas involucran un vértice del track 1 y otro del track 2. Chequeando los vecinos más cercanos en el track opuesto vemos que:

- v_2 es el vértice más cercano del track 2 a la derecha de v_1, y v_1 es el vértice más cercano del track 1 a la izquierda de v_2, así que $\{v_1, v_2\}$ es legal.
- v_n es el vértice más cercano del track 1 a la derecha de v_{n-1} y v_{n-1} es el más cercano del track 2 a la izquierda de v_n, así que $\{v_{n-1}, v_n\}$ es legal.

Arista $\{v_n, v_1\}$. Ambos extremos están en el track 1 y son los únicos vértices de ese track, entonces

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

y $\{v_n, v_1\}$ es legal.

Por lo tanto (τ, p) es un diseño en 2 tracks en el que todas las aristas de C_n son legales, así que C_n es representable por vecinos más cercanos en 2 tracks.

Combinado con la no representabilidad para $k = 1$, obtenemos $\mathrm{tn}(C_n) = 2$. □

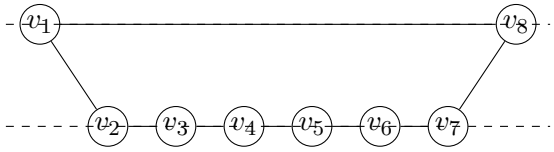

Figure 18: Un diseño en 2 tracks de C_8 con v_1, v_8 en el track 1 y $v_2, \ldots, v_7$ en el track 2.

6.2 Un camino más una arista extra

Mostramos ahora que un camino con una sola arista adicional siempre es representable en 2 tracks. Es un ejemplo simple de grafo que ya *no* es un bosque lineal pero igual necesita sólo $k = 2$ tracks.

Theorem 6.2 (Caminos con una arista extra)**.** *Sea G un grafo conexo simple no dirigido obtenido así:*

- $V = \{v_1, \ldots, v_n\}$ *para algún* $n \geq 2$,
- E *contiene todas las aristas del camino* $\{v_i, v_{i+1}\}$ *para* $i = 1, \ldots, n-1$,
- *y además una única arista extra* $e^\star = \{v_a, v_b\}$ *con* $1 \leq a < b \leq n$.

Entonces G es representable por vecinos más cercanos en 2 tracks.

La Figura 19 ilustra la construcción para una arista extra $\{v_3, v_7\}$. La misma idea funciona cuando uno o ambos extremos de la arista extra están en los extremos del camino. Como un grafo representable por vecinos más cercanos en 1 track es un bosque lineal (Remarca 1.5) y nuestros grafos contienen un ciclo, no pueden ser representables en 1 track. Junto con la construcción anterior, esto muestra que su número de tracks de vecinos más cercanos es exactamente 2.

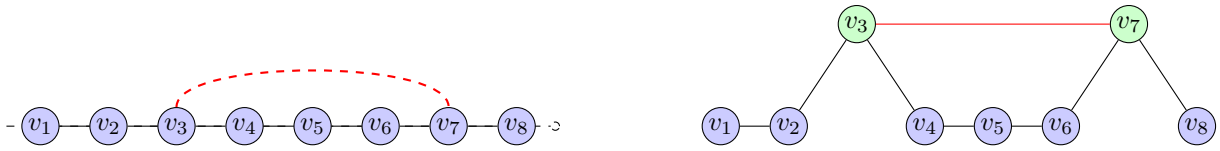

Figure 19: Un camino $v_1 - \cdots - v_8$ con una arista extra $\{v_3, v_7\}$. Izquierda: el diseño natural en 1 track del camino; la arista extra (roja, punteada) salta varios vértices y no es una arista de vecinos más cercanos. Derecha: moviendo v_3 y v_7 al track 2, obtenemos un diseño en 2 tracks en el que la arista extra (roja sólida) pasa a ser legal mientras que todas las aristas del camino siguen siendo legales.

Proof. Mantenemos el orden natural $p(v_i) = i$ y sólo modificamos la asignación de tracks.

Paso 1: arrancamos del camino. Para el camino puro P_n con aristas $\{v_i, v_{i+1}\}$, el diseño

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

es un diseño de vecinos más cercanos en 1 track: cada arista $\{v_i, v_{i+1}\}$ conecta vértices consecutivos en el track 1.

Paso 2: movemos los extremos de la arista extra al track 2. Ahora agregamos la arista extra $e^\star = \{v_a, v_b\}$ con $a < b$, mantenemos el mismo orden global p y definimos una nueva asignación de tracks τ por

$$\tau(v_i) = \begin{cases} 2, & \text{si } i \in \{a, b\}, \\ 1, & \text{en otro caso.} \end{cases}$$

Así, sólo v_a y v_b están en el track 2; todos los demás vértices están en el track 1.

Chequeamos que todas las aristas de G son legales bajo (τ, p).

Paso 3: aristas del camino no incidentes en v_a o v_b. Si $i \notin \{a-1, a, b-1, b\}$, entonces $\{v_i, v_{i+1}\}$ no es incidente ni en v_a ni en v_b y ambos extremos están en el track 1. Son consecutivos en el orden global y no hay otro vértice del track 1 entre ellos; por lo tanto

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

y $\{v_i, v_{i+1}\}$ es legal.

Paso 4: aristas del camino incidentes en v_a o v_b. Consideremos las aristas $\{v_{a-1}, v_a\}$ y $\{v_a, v_{a+1}\}$ (cuando esos índices existan). El vecino de v_a inmediatamente a la izquierda en el track 1 es v_{a-1} y el de la derecha es v_{a+1}, entonces

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

Así, las aristas $\{v_{a-1}, v_a\}$ y $\{v_a, v_{a+1}\}$ son legales.

El mismo argumento aplica a v_b: los vértices más cercanos en el track 1 a la izquierda y a la derecha son v_{b-1} y v_{b+1} (si existen), haciendo que $\{v_{b-1}, v_b\}$ y $\{v_b, v_{b+1}\}$ sean legales.

Paso 5: la arista extra $\{v_a, v_b\}$. En el track 2, v_a y v_b son los únicos vértices y $a < b$. Entonces

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

luego $v_b \in N_{\text{legal}}(v_a)$ y $v_a \in N_{\text{legal}}(v_b)$. Por lo tanto la arista extra es legal.

En conjunto, cada arista de G es legal, así que G es representable por vecinos más cercanos en 2 tracks. □

7 Planaridad para $k \leq 2$

Finalmente, mostramos que $k \leq 2$ impone restricciones estructurales fuertes: todo grafo de este tipo es planar.

Theorem 7.1 (Planaridad para $k \leq 2$)**.** *Sea G un grafo simple no dirigido representable por vecinos más cercanos en (k) tracks con $k \leq 2$. Entonces G es planar.*

Proof. Tratamos por separado los casos $k = 1$ y $k = 2$.

Caso $k = 1$. Cuando hay un solo track, cada vértice tiene a lo sumo un predecesor y un sucesor, y toda arista legal conecta vértices consecutivos en el orden del track. Por lo tanto, cada componente conexa de G es un camino (o un vértice aislado). Un grafo así es una unión disjunta de caminos y vértices aislados, luego planar.

Caso $k = 2$: reducción a grafos anfitriones. Fijamos un diseño en 2 tracks (τ, p) sobre V y sea $H(\tau, p)$ su grafo anfitrión con todas las aristas legales. Como G es un subgrafo de $H(\tau, p)$, alcanza con ver que $H(\tau, p)$ es planar.

Ponemos los vértices sobre una recta horizontal en el orden global $v_1, \ldots, v_n$ con $p(v_i) = i$, y escribimos $\tau(v_i) \in \{1, 2\}$. Particionamos las aristas de $H(\tau, p)$ en cuatro clases:

- $E^{(1)}$: aristas entre vértices consecutivos en el track 1,
- $E^{(2)}$: aristas entre vértices consecutivos en el track 2,
- E_R: aristas cruzadas donde cada extremo es el vecino más cercano en el track opuesto hacia la derecha,
- E_L: aristas cruzadas donde cada extremo es el vecino más cercano en el track opuesto hacia la izquierda.

La Figura 20 ilustra esquemáticamente el dibujo planar usado en la prueba.

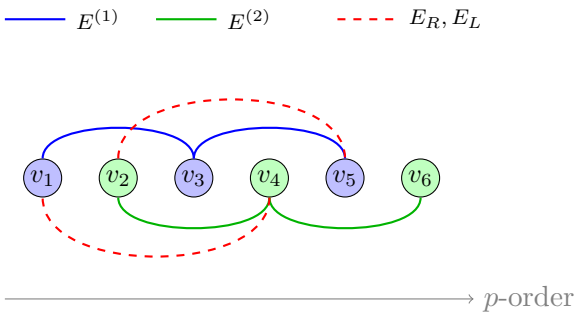

Figure 20: Un embedding planar esquemático para un grafo anfitrión en 2 tracks: las aristas en el track 1 y las aristas cruzadas hacia la derecha se dibujan arriba de la recta, mientras que las aristas en el track 2 y las cruzadas hacia la izquierda se dibujan abajo.

Dibujamos:

- todos los vértices $v_1, \ldots, v_n$ sobre el eje x en $(i, 0)$,
- todas las aristas en $E^{(1)} \cup E_R$ como arcos x-monótonos en el semiplano superior,

- todas las aristas en $E^{(2)} \cup E_L$ como arcos x-monótonos en el semiplano inferior.

Es un argumento estándar de órdenes interválicos (omitimos los detalles de rutina) que:

- las aristas en $E^{(1)}$ forman una familia sin cruces en el semiplano superior,
- las aristas en E_R forman una familia sin cruces en el semiplano superior,
- ninguna arista de $E^{(1)}$ se cruza con una arista de E_R en el semiplano superior,
- por simetría, lo mismo vale para $E^{(2)}$ y E_L en el semiplano inferior.

Como las aristas dibujadas arriba y abajo sólo se encuentran sobre la recta horizontal en sus extremos, esto da un embedding planar de $H(\tau, p)$. Luego $H(\tau, p)$ es planar, y cualquier subgrafo G suyo también lo es. □

Esta restricción de planaridad, junto con las cotas de grado y aristas probadas antes, se va a aprovechar en la Sección 9 para diseñar un algoritmo de reconocimiento para diseños de vecinos más cercanos en 2 tracks.

8 Complejidad: reconocer diseños de vecinos más cercanos en 2 tracks

8.1 El problema de decisión 2TNN-Layout

Fijamos $k = 2$ a lo largo de esta sección. Recordemos que un diseño de vecinos más cercanos en 2 tracks de un grafo $G = (V, E)$ consiste en:

- una asignación de track $\tau : V \to \{1, 2\}$,
- y una biyección $p : V \to \{1, \ldots, |V|\}$,

tal que cada arista $\{u, v\} \in E$ es legal en el sentido de la Sección 1: cada extremo es el predecesor o sucesor del otro en algún track (el propio o el opuesto).

Definition 8.1 (2TNN-Layout)**.** El problema de decisión 2TNN-Layout es:

Input: Un grafo simple finito no dirigido $G = (V, E)$.
Pregunta: ¿Existe un diseño ordenado en 2 tracks (τ, p) de G tal que G sea representable por vecinos más cercanos en 2 tracks respecto de (τ, p), es decir

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

Equivalentemente, ¿se cumple que $\text{tn}(G) \leq 2$ en el sentido de vecinos más cercanos?

8.2 Pertenencia a NP

Proposition 8.2 (2TNN-Layout está en NP)**.** *El problema* 2TNN-Layout *pertenece a la clase de complejidad NP.*

Proof. Un certificado consiste en un par (τ, p) con $\tau : V \to \{1, 2\}$ y una biyección $p : V \to \{1, \ldots, n\}$. A partir de (τ, p) podemos computar todos los predecesores y sucesores a lo largo de cada track en tiempo lineal y, con eso, los conjuntos de vecinos legales $N_{\text{legal}}(v)$. Luego verificamos en tiempo $O(|E|)$ que cada arista de G sea legal en ambos extremos. Por lo tanto 2TNN-Layout está en NP. □

9 Un algoritmo de reconocimiento para diseños en dos tracks

Apoyándonos en la mirada de complejidad de la sección anterior, describimos ahora un procedimiento concreto de decisión para $\mathrm{tn}(G) \leq 2$.

Por la Proposición 3.2 podemos trabajar componente conexa por componente, ya que $\mathrm{tn}(G) = \max_i \mathrm{tn}(G_i)$.

En toda esta sección, para una componente H escribimos

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

Nuestro algoritmo tiene dos capas:

1. una *capa de filtros rápidos* con tests estructurales que en tiempo $O(n+m)$ o bien certifican *SÍ* o bien *NO* para la mayoría de los grafos, y
2. una *capa de backtracking* sobre los núcleos "difíciles" que queden, basada en los cuatro slots de vecinos más cercanos que cada vértice puede usar cuando $k = 2$.

Organizamos el procedimiento de reconocimiento como una secuencia de "cortes" estructurales que explicitamente realizan las condiciones necesarias simples desarrolladas en las secciones previas, seguidos de una búsqueda exponencial sobre el núcleo restante.

9.1 Decisiones inmediatas por tamaño y densidad

Fijamos una componente conexa H.

Corte 0: grafos muy chicos. Si $n_H \leq 3$ entonces H tiene a lo sumo tres aristas y no contiene un K_4, y uno puede escribir fácilmente a mano un diseño de vecinos más cercanos en 2 tracks. Tomamos entonces

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

como un *SÍ automático*. Esto se chequea en tiempo $O(1)$ por componente.

Corte 1: cota superior de densidad de aristas. Por el Teorema 5.6, todo grafo anfitrión en 2 tracks sobre n_H vértices tiene a lo sumo $2n_H - 3$ aristas. Por lo tanto, cualquier componente con $m_H > 2n_H - 3$ no puede tener $\mathrm{tn}(H) \leq 2$.

9.2 Filtros estructurales lineales

Una vez que $n_H \geq 4$ y $m_H \leq 2n_H - 3$, aplicamos tres cortes más, cada uno justificado por resultados previos.

Corte 2: grado máximo. Por el Lema 5.2, un diseño de vecinos más cercanos en k tracks satisface $\Delta(H) \leq 2k$. Para $k \leq 2$ esto implica

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

Así que si alguna vez vemos $\Delta(H) \geq 5$ podemos concluir de inmediato que $\mathrm{tn}(H) \geq 3$ y devolver *NO*. Computar todos los grados lleva tiempo $O(n_H + m_H)$.

Corte 3: subgrafo K_4. El Teorema 5.1 muestra que un clique de tamaño 4 fuerza al menos tres tracks. Entonces si H contiene un K_4 debemos rechazar:

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

Dado que desde el Corte 2 tenemos $\Delta(H) \leq 4$, un K_4 se puede detectar en tiempo $O(n_H \Delta(H)^2) = O(n_H)$ intersectando conjuntos de vecinos.

Corte 4: planaridad. Por el Teorema 7.1, cualquier grafo con $\mathrm{tn}(H) \leq 2$ debe ser planar. Entonces corremos un test de planaridad en tiempo lineal (por ejemplo Hopcroft–Tarjan) y rechazamos si H no es planar:

$$H \text{ no planar} \quad \Longrightarrow \quad \mathrm{tn}(H) \geq 3.$$

Esto cuesta $O(n_H + m_H)$.

Luego de los Cortes 1–4, todas las componentes que quedan satisfacen simultáneamente

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

9.3 Casos de SÍ baratos

Antes de pasar al backtracking, también aprovechamos las construcciones explícitas de la Sección 5.

Corte 5: bosques lineales. Si cada componente de H es un camino o un vértice aislado (un bosque lineal), entonces $\mathrm{tn}(H) = 1$ por lo discutido en la Sección 1, y respondemos *SÍ*. En el caso conexo esto equivale a testear

$$\Delta(H) \leq 2 \quad \text{y } H \text{ acíclico}.$$

La aciclicidad y los grados se pueden chequear en tiempo $O(n_H + m_H)$.

Corte 6: un ciclo con grado máximo 2. La Sección 5 prueba que

- todo ciclo C_n tiene $\mathrm{tn}(C_n) = 2$ (Teorema 6.1), y
- cualquier grafo obtenido a partir de un camino agregando una arista extra también es representable por vecinos más cercanos en 2 tracks (Teorema 6.2).

Estos son precisamente los grafos conexos con

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

es decir, grafos con tantas aristas como vértices y grado máximo a lo suma dos.

Así que las componentes conexas con

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

son *SÍ automáticos* y nunca llegan a la capa de búsqueda exponencial. De nuevo, este test es $O(n_H + m_H)$.

(Se puede extender esta familia si uno quiere, usando más teoremas estructurales, pero lo anterior ya cubre los casos unicíclcos escasos más comunes.)

9.4 Reducción a un núcleo planar de grado acotado

Después de los Cortes 0–6, sólo quedan componentes "realmente complicadas": grafos planares, libres de K_4, con $3 \le \delta(H) \le \Delta(H) \le 4$ y $m_H \le 2n_H - 3$ que no son ni bosques lineales ni ciclos simples ni caminos con una única arista extra.

En este punto contraemos H a un *núcleo* $H^\star$ más pequeño eliminando hojas innecesarias y vértices de grado 2 en caminos, al estilo de la kernelización estándar.

- Mientras H tenga una hoja v (grado 1), borramos v y su arista incidente. Por el clásico argumento de "volver a pegar la hoja", esto no cambia si existe o no un diseño de vecinos más cercanos en 2 tracks.
- Mientras H tenga un vértice de grado 2 cuyo par de vecinos x, y no están adyacentes, contraemos el camino $x - v - y$ en una única arista xy. Esto preserva la existencia de un diseño: en cualquier diseño del grafo contraído, v se puede reinsertar en el mismo track entre x e y.

Llamamos $H^\star$ al resultado, el *núcleo en dos tracks* de H. Es planar, libre de K_4, tiene $3 \le \delta(H^\star) \le \Delta(H^\star) \le 4$, y se puede computar en tiempo $O(n_H + m_H)$.

Si $H^\star$ está vacío, o es un único vértice, o es un único ciclo, ya sabemos cómo diseñar H a partir de los cortes previos; si no, $H^\star$ es la instancia sobre la que corremos la búsqueda exponencial.

9.5 La capa de backtracking

Sobre $H^\star$ corremos ahora un algoritmo de backtracking exacto pero exponencial en el tamaño de $H^\star$. La observación clave es que para $k = 2$ cada vértice v puede usar a lo sumo cuatro posiciones legales de vecinos más cercanos:

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

Codificamos estas posiciones como cuatro *slots* abstractos

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

y buscamos asignaciones de aristas incidentes a slots, sujetas a:

- en cada vértice, a lo sumo una arista usa cada slot (cota de grado),
- en cada arista $\{u, v\}$, el par de slots elegidos en u y v debe ser uno de los ocho pares compatibles que vienen de una relación predecesor/sucesor real (en el mismo track o cruzado),
- las restricciones inducidas de "estar a la izquierda de" entre vértices deben quedar acíclicas, para que se puedan realizar por algún orden global p.

Mantenemos asignaciones parciales de slots y sus implicancias como un CSP de dominios finitos, y ramificamos en el vértice con menos slots disponibles. Como $\deg(v) \le 4$ y hay cuatro slots, el factor de ramificación local es a lo sumo $4! = 24$, y normalmente mucho menor una vez que propagamos las restricciones.

Una asignación completa y consistente de slots produce un diseño de vecinos más cercanos en 2 tracks de $H^\star$ (y por lo tanto de H), mientras que el fracaso en todas las ramas prueba que no existe tal diseño. El argumento de corrección es estándar y sigue el mismo patrón que otros CSPs de diseño y orden; omitimos los detalles para no alargar aún más.

En el peor caso esta capa de backtracking corre en tiempo

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

para alguna constante $c < 24$ y $n^\star = |V(H^\star)|$, multiplicado por un factor polinomial correspondiente a la propagación de restricciones. Este comportamiento exponencial es esperable: ya el problema estándar de reconocer el track-number para $k = 2$ fijo es NP-completo (ver Dujmović et al. [1, 2]).

9.6 Algoritmo final y complejidad global

Podemos resumir ahora todo el procedimiento.

Algoritmo TwoTrackNN(G)

Input: grafo simple finito $G = (V, E)$.
Output: 1 si G tiene un diseño de vecinos más cercanos con a lo sumo dos tracks, 0 en otro caso.

1. Computar las componentes conexas $G_1, \ldots, G_r$ de G (tiempo $O(|V| + |E|)$).
2. Para cada componente G_i:
   (a) Sea $n_i = |V(G_i)|$, $m_i = |E(G_i)|$. Si $n_i \le 3$, marcar G_i como *SÍ* y seguir.
   (b) Si $m_i > 2n_i - 3$, devolver 0 (Corte 1).
   (c) Computar los grados. Si $\Delta(G_i) \ge 5$, devolver 0 (Corte 2).
   (d) Testear si hay un K_4; si aparece, devolver 0 (Corte 3).
   (e) Correr un test de planaridad en tiempo lineal; si G_i no es planar, devolver 0 (Corte 4).
   (f) Si G_i es un bosque lineal, o un ciclo, o un camino con una arista extra (Cortes 5 y 6), marcarlo como *SÍ* y seguir.
   (g) En otro caso, computar su núcleo $G_i^\star$ borrando hojas y suprimiendo vértices de grado 2 en caminos (Sección 9.4).
   (h) Correr la búsqueda de backtracking basada en slots sobre $G_i^\star$. Si falla, devolver 0; si tiene éxito, marcar G_i como *SÍ*.
3. Si todas las componentes están marcadas como *SÍ*, devolver 1.

Los pasos (1)–(2f) y la reducción al núcleo (2g) en conjunto toman

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

tiempo.

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 Intentos fallidos de cortes basados en BFS

Durante el diseño del algoritmo de reconocimiento en la Sección 9 es tentador buscar condiciones necesarias extra y "baratas" que vengan del comportamiento de un BFS (búsqueda en anchura). En este apéndice registramos dos ideas de ese estilo que terminaron siendo inviables: una es demasiado débil como para descartar algo más allá de la cota de grado, y la otra directamente es falsa (incluso para grafos muy chicos representables en 2 tracks).

A.1 Una cota demasiado débil: a lo sumo tres vértices nuevos por expansión

Una primera intuición era que, en un diseño de vecinos más cercanos en 2 tracks, una expansión de BFS nunca debería descubrir más de dos vértices nuevos a la vez. Eso es falso en general (un vértice puede tener grado 4), pero incluso la versión corregida

"toda expansión de BFS descubre a lo sumo tres vértices nuevos"

resulta demasiado débil para ser un corte útil.

De hecho, esta cota viene automáticamente de la restricción local de grado $\Delta(G) \leq 4$ (Lema 5.2). Si enraizamos un BFS en algún vértice r y expandimos los vértices uno por uno, para cualquier vértice $v \neq r$:

- una arista incidente va al padre de v en el BFS y no se cuenta como "nueva",
- las restantes a lo sumo $\deg(v) - 1 \leq 3$ aristas incidentes pueden llevar a vértices nuevos.

Así, la propiedad de "≤ 3 vértices nuevos por expansión" vale en todo grafo representable por vecinos más cercanos en 2 tracks y no agrega información más allá de la cota de grado; no puede filtrar instancias adicionales.

A.2 Una condición falsa de patrones: 2 nuevos y después 3 nuevos

Una idea más ambiciosa fue explotar la *secuencia* de tamaños de capas del BFS. Sea $L_0, L_1, L_2, \ldots$ las capas de un BFS a partir de alguna raíz elegida, donde L_i es el conjunto de vértices a distancia i de la raíz y $|L_i|$ es el número de vértices que se descubren *por primera vez* en esa capa.

La condición tentativa era, informalmente:

> *En un grafo representable por vecinos más cercanos en 2 tracks no debería existir un BFS en el que una capa con 3 vértices nuevos esté inmediatamente precedida por una capa con 2 vértices nuevos.*

La idea era usar esto como condición necesaria: si *algún* BFS desde *alguna* raíz exhibe el patrón

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

entonces el grafo se descartaría como candidato a tener un diseño en 2 tracks de vecinos más cercanos. Sin embargo, el siguiente ejemplo minúsculo muestra que este patrón sí aparece en un grafo perfectamente válido de ese tipo.

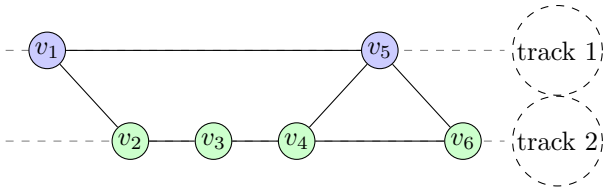

Figure 21: Un grafo representable por vecinos más cercanos en 2 tracks donde un BFS desde v_1 tiene $|L_0| = 1$, $|L_1| = 2$ y $|L_2| = 3$.

Example A.1 (Capas 2 y después 3 en un BFS de un grafo en 2 tracks). Consideremos el diseño en 2 tracks de la Figura 21 con orden global

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

asignación de tracks

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

y aristas

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

Cada arista es entre vecinos más cercanos en algún track (mismo track o track opuesto), así que es un diseño válido de vecinos más cercanos en 2 tracks.

Corremos un BFS desde la raíz v_1. Las capas son:

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

Efectivamente:

- Desde $L_0 = \{v_1\}$ descubrimos sus vecinos v_2 y v_5, así que $|L_1| = 2$.
- Desde L_1 descubrimos, en total, los vecinos v_3 (vía v_2) y v_4, v_6 (vía v_5), dando $L_2 = \{v_3, v_4, v_6\}$ y $|L_2| = 3$.

Así, este grafo exhibe el patrón

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

para un BFS completamente legítimo, contradiciendo la regla propuesta de que una capa de tamaño 3 no puede estar precedida por una capa de tamaño 2 en un grafo representable por vecinos más cercanos en 2 tracks.

El Ejemplo A.1 muestra que estos patrones de tamaños de capas de BFS son demasiado delicados como para usarlos como condiciones necesarias para diseños de vecinos más cercanos en 2 tracks: incluso grafos muy chicos que *sí* admiten tales diseños pueden tener secuencias de capas del tipo $2 \to 3$. Por esta razón abandonamos todos los cortes basados en "tamaños de capas de BFS" y nos quedamos sólo con las condiciones estructurales locales robustas (cotas de grado, planaridad y ausencia de K_4) en el algoritmo principal.

B Bonus Track: Algoritmo para descubrir $k = \mathrm{tn}(G)$

Mientras que la Sección 9 resuelve el problema de *decisión* "¿Es $\mathrm{tn}(G) \leq 2$?", en esta sección abordamos el problema de *optimización*: dado un grafo G, calcular el valor exacto de $\mathrm{tn}(G)$.

El algoritmo sigue una estrategia de **Filtrado Progresivo y Descomposición**. En lugar de atacar el problema NP-hard directamente, simplificamos el grafo matemáticamente hasta reducirlo a sus componentes irreducibles.

> **Entrada:** Un grafo no dirigido $G = (V, E)$.
> **Salida:** El entero k mínimo tal que G admite un diseño de vecinos más cercanos en k tracks.

B.1 Cota inferior teórica (el piso)

Acción. Calculamos el grado máximo del grafo, $\Delta(G)$, y establecemos el valor inicial de búsqueda:

$$k_{\min} = \left\lceil \frac{\Delta(G)}{2} \right\rceil.$$

Justificación teórica. Por el Lema 5.2 (Sección 1), en un diseño de vecinos más cercanos cada vértice tiene a lo sumo 2 vecinos por track (un predecesor y un sucesor). Con k tracks, un vértice puede gestionar como máximo $2k$ conexiones. Esto implica

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

Es *físicamente imposible* usar menos tracks que $k_{\min}$.

B.2 Detección de estructuras triviales (atajos $O(n + m)$)

Acción. Verificamos si el grafo carece de ciclos (es un bosque).

- Si es un **bosque lineal** ($\Delta \leq 2$, sin ciclos): retornamos $k = 1$.
- Si es un **árbol/bosque general** ($\Delta > 2$, sin ciclos): retornamos $k = k_{\min}$.

Justificación teórica. Esta es una contribución clave sobre árboles. Sin ciclos no hay *obstrucciones topológicas*: la única razón para necesitar más "colores" que los que dicta la capacidad ($\Delta/2$) es la presencia de ciclos que se cierran sobre sí mismos.

Proposition B.1 (Track number de árboles)**.** *Sea T un árbol (o bosque). Entonces*

$$\mathrm{tn}(T) = \max\left\{1, \left\lceil \frac{\Delta(T)}{2} \right\rceil\right\}.$$

Idea de la demostración. Fijamos una raíz r y recorremos el árbol en orden BFS o DFS. En cada paso, al agregar un vértice v con padre u, asignamos v a un track y una posición que hagan legal la arista $\{u, v\}$. Como no hay ciclos, nunca se genera un conflicto entre las restricciones de distintas aristas. La única limitante es la capacidad de cada vértice, que requiere exactamente $\lceil \deg(v)/2 \rceil$ tracks distintos para sus vecinos. □

En ausencia de ciclos, la cota inferior de capacidad es también una cota superior suficiente. *No hace falta búsqueda*; la fórmula es exacta.

B.3 Kernelización (poda de hojas)

Acción. Eliminamos recursivamente todos los vértices de grado 1 hasta que no quede ninguno.

- Si el grafo se vacía: retornamos $k_{\min}$.
- Si quedan nodos: obtenemos el **núcleo topológico** G_{core}.

Justificación teórica (propiedad de inserción libre). Un vértice hoja v conectado a un único vecino u siempre puede ser insertado en un diseño válido existente de $G - \{v\}$ sin violar ninguna regla ni requerir tracks adicionales (siempre que $k \geq 1$), colocándolo adyacente a u en el orden lineal del track apropiado.

Lemma B.2 (Inserción de hojas)**.** *Sea G un grafo y $v \in V(G)$ un vértice de grado 1 con único vecino u. Entonces*

$$\text{tn}(G) = \text{tn}(G - v).$$

Proof. La dirección $\text{tn}(G) \geq \text{tn}(G-v)$ es trivial (un subgrafo no requiere más tracks). Para la otra dirección, dado un diseño (τ, p) de $G - v$ en k tracks, extendemos a G colocando v en el mismo track que u, inmediatamente a la derecha (o izquierda) de u en el orden global. Esto hace que $\{u, v\}$ sea una arista legal sin afectar las demás aristas. □

Esta propiedad nos permite ignorar las "ramas" de los árboles y concentrar el poder computacional solo en la parte difícil: los ciclos.

B.4 Descomposición biconexa (divide y vencerás)

Acción. Usamos DFS para encontrar puntos de articulación en el núcleo G_{core} y dividirlo en una lista de bloques biconexos $(B_1, B_2, \ldots, B_m)$.

Justificación teórica (propiedad de localidad del track number).

Theorem B.3 (Localidad del track number)**.** *Sea G un grafo conexo con bloques biconexos $B_1, \ldots, B_m$. Entonces*

$$\text{tn}(G) = \max_{i=1}^{m} \text{tn}(B_i).$$

Idea de la demostración. Dos bloques que comparten un único vértice (punto de articulación) son casi independientes. Dado un diseño válido para cada bloque B_i, podemos "pegar" los diseños identificando el punto de articulación y concatenando los órdenes apropiadamente. La única restricción es que el vértice compartido tenga suficiente capacidad total, lo cual ya está cubierto por la cota global de grado. □

Esta fase transforma la complejidad del problema de **multiplicativa** $O(2^{\sum_i n_i})$ a **aditiva** $O(\sum_i 2^{n_i})$, haciendo tratables los grafos grandes con muchos bloques pequeños.

B.5 Solver exacto con heurística de grado (el motor)

Acción. Para cada bloque B_i, buscamos el k local mínimo usando backtracking optimizado (o un SAT solver).

Punto de partida. Para el bloque B_i, empezamos probando $k = \lceil \Delta(B_i)/2 \rceil$.

Ordenamiento (heurística fail-first). Dentro del solver, las variables (nodos) *no* se asignan al azar. Se ordenan por **grado descendente**: los nodos de alto grado son los más restringidos (*most constrained variables*).

Remark B.4 (Efecto de la heurística de grado). Al intentar colocar primero los nodos de alto grado, si no caben, el algoritmo detecta el conflicto en los niveles superiores del árbol de búsqueda. Esto permite una *poda masiva* (pruning) y evita explorar millones de ramas inútiles.

Búsqueda.

- Si el solver encuentra solución para k, guardamos ese valor y pasamos al siguiente bloque.
- Si no, probamos $k+1$ y repetimos.

Observamos empíricamente que en la práctica $\mathrm{tn}(G) \leq \lceil \Delta(G)/2 \rceil + 1$ para la mayoría de grafos, por lo que raramente probamos más de dos valores de k por bloque.

Nota: La Conjetura de Akiyama–Chvátal establece que la *arboricidad lineal* satisface $\mathrm{la}(G) \leq \lceil (\Delta(G)+1)/2 \rceil$ (se sabe que $\mathrm{la}(G) \geq \lceil \Delta(G)/2 \rceil$). Sin embargo, el track number $\mathrm{tn}(G)$ es un problema distinto: requiere que *todos* los bosques lineales compartan un orden global único, lo cual puede forzar $\mathrm{tn}(G) > \mathrm{la}(G)$. La relación exacta entre ambos invariantes sigue siendo un problema abierto.

Reducción a SAT. El problema de evitar patrones prohibidos es equivalente a satisfacer un conjunto de cláusulas lógicas. Para cada arista $\{u, v\}$ y cada posible asignación de tracks y posiciones relativas, codificamos las restricciones de vecinos más cercanos como cláusulas. Un SAT solver moderno puede resolver instancias de tamaño moderado ($n \leq 30$) en segundos.

B.6 Agregación de resultados

Acción. El resultado final del grafo completo es el máximo entre:

- la necesidad teórica global del grafo original (antes de podar), y
- la necesidad topológica encontrada en el bloque más difícil.

$$k_{\text{final}} = \max\left(\left\lceil \frac{\Delta(G_{\text{orig}})}{2} \right\rceil,\ \max_i \mathrm{tn}(B_i) \right).$$

B.7 Algoritmo final y complejidad

Resumimos el procedimiento completo.

Algoritmo ComputeTrackNumber(G)

Input: grafo simple finito $G = (V, E)$.
Output: $\mathrm{tn}(G)$, el track number de vecinos más cercanos.

1. **Cota global:** $k_{\text{global}} \leftarrow \lceil \Delta(G)/2 \rceil$.
2. **Atajo árboles:** Si G es un bosque (sin ciclos), retornar k_{global}.
3. **Kernelización:** $G_{\text{core}} \leftarrow$ PodaHojas(G). Si $G_{\text{core}} = \emptyset$, retornar k_{global}.
4. **Descomposición:** $(B_1, \ldots, B_m) \leftarrow$ BloquesBiconexos(G_{core}).

5. **Solver local:** $k_{\max} \leftarrow 0$.
   Para cada B_i:
   (a) Ordenar nodos de B_i por grado descendente.
   (b) $k \leftarrow \max(\lceil \Delta(B_i)/2 \rceil, k_{\max})$. *(Si $\lceil \Delta(B_i)/2 \rceil \leq k_{max}$, ya sabemos que necesitamos al menos k_{max} tracks; no tiene sentido probar valores menores.)*
   (c) **Mientras** no se encuentre solución:
   - Si SolverBacktracking(B_i, k) tiene éxito: $k_{\max} \leftarrow \max(k_{\max}, k)$; **break**.
   - Si no: $k \leftarrow k + 1$.
6. **Agregación:** Retornar $\max(k_{\text{global}}, k_{\max})$.

Complejidad.

- **Caso mejor:** Árboles y bosques se resuelven en $O(|V| + |E|)$.
- **Caso peor:** Grafos biconexos densos requieren backtracking exponencial $O(c^n)$ para alguna constante c, pero la descomposición en bloques y la heurística de grado reducen drásticamente el espacio de búsqueda en la práctica.
- **Observación empírica:** En la práctica, para la mayoría de grafos se cumple $\text{tn}(G) \leq \lceil \Delta(G)/2 \rceil + 1$, por lo que el solver raramente necesita probar más de dos valores de k por bloque. Conjeturamos que esta cota es válida en general, pero no está demostrada (notar que la Conjetura de Akiyama–Chvátal es sobre arboricidad lineal, un problema relacionado pero distinto).