Módulo de Taller
Confiabilidad de grafos con dos terminales con fallas en vértices

Guillermo Arlotto, Santiago Márquez

Facultad de Ingeniería, Universidad de la República

25 de mayo de 2026

Resumen

Un grafo con dos terminales G es un grafo simple con vértices terminales especificados s y t. Definimos $T^d_{n,m}$ como el conjunto de grafos con dos terminales que tienen n vértices no terminales, m aristas y cuya distancia entre sus terminales es igual a d. Sea $G \in T^d_{n,m}$ un grafo con dos terminales, y sea $p \in [0,1]$. La confiabilidad de G, $NR_G(p)$, es la probabilidad de que exista un camino entre s y t de largo d cuando cada vértice de G falla independientemente con probabilidad $1-p$. Si para cada grafo H en $T^d_{n,m}$ y para todo $p \in [0,1]$ se cumple que $NR_G(p) \geq NR_H(p)$, decimos que G es uniformemente más confiable (UMR, por sus siglas en inglés).

En su trabajo Brown *et al.* [Networks 76(3):414-426, 2020] prueban que para cada par de enteros n y m tales que $n \geq 2$ y $m \leq \binom{n}{2} + n$ existe UMR en $T^3_{n,m}$ si y solo si se cumple exactamente una de las tres condiciones: $m \leq 8$, $n = 7$ y $m = 12$, o $m \geq \lfloor \frac{(n+1)^2}{4} \rfloor$. Los autores conjeturan que si $d \geq 4$, $n \geq 2(d-1)$ y $2d \leq m \leq (d-2)\lfloor \frac{n}{d-1} \rfloor^2 + \lfloor \frac{n}{d-1} \rfloor - 1$ entonces no existe UMR en $T^d_{n,m}$. Por su parte, Miranda [Tesis de grado, Universidad de la República (Uruguay). Facultad de Ingeniería, 2025] demostró que se cumple la conjetura de Brown para $d = 4$, desarrollando una estrategia de demostración a partir de la cual basamos nuestro trabajo actual.

En este trabajo estudiamos la inexistencia de grafos UMR cuando $d = 5$ mediante tres enfoques complementarios: (i) desarrollando una estrategia de demostración basada en técnicas de optimización matricial y transformaciones sobre grafos, (ii) aplicando un método de optimización basado en *Variable Neighborhood Search (VNS)*, y (iii) proporcionando verificación computacional de la conjetura para familias finitas de grafos, mediante un algoritmo de búsqueda local VNS sobre distintas familias de grafos $T^5_{n,m}$ y otro enfoque de búsquedas exhaustivas en las familias de grafos $T^5_{18,22}$, $T^5_{18,24}$ y $T^5_{12,22}$.

Palabras clave: Teoría de grafos, Confiabilidad de redes, VNS, Grafo con dos terminales, Modelo de fallas en vértices.

Índice

1. Introducción

La confiabilidad de un grafo con dos terminales es un área de estudio relevante en teoría de grafos y redes [4], con aplicaciones en diseño de redes de comunicación, infraestructura y sistemas distribuidos. En particular, el estudio de grafos con dos terminales donde los vértices pueden fallar de manera aleatoria permite modelar escenarios de redes vulnerables a fallas de nodos. Existe el deseo de diseñar el sistema perfecto utilizando componentes imperfectos [4].

Un grafo con dos terminales es un grafo G con dos vértices distinguidos s y t, que llamaremos terminales. Sea $T^d_{n,m}$ el conjunto de grafos con dos terminales que tienen n vértices no terminales, m aristas, y la distancia entre sus terminales es igual a d. Para cada G en $T^d_{n,m}$ y cada $p \in [0,1]$ definimos la confiabilidad de G evaluada en p como la probabilidad de que exista un camino entre s y t de largo d cuando cada vértice falla independientemente con probabilidad $1-p$. Denotaremos la confiabilidad de G evaluada en p mediante $NR_G(p)$.

Diremos que un grafo G en $T^d_{n,m}$ es uniformemente más confiable (UMR por sus siglas inglesas) cuando para cada grafo H en $T^d_{n,m}$ y para todo p en $[0,1]$ se cumple que $NR_G(p) \geq NR_H(p)$.

Para cada par de enteros positivos k y d denotaremos D^d_k al grafo con dos terminales que posee exactamente k caminos internamente disjuntos de largo d cuyos extremos son s y t.

Brown *et al.* en 2020 exploraron la existencia de grafos uniformemente más confiables con dos terminales. Primeramente prueban que cada grafo en $T^2_{n,m}$ que incluye a D^2_k es uniformemente más confiable. Luego prueban que en $T^3_{n,m}$ existe grafo uniformemente más confiable si y solo si se cumple exactamente una de las 3 condiciones: $m \leq 8$, $n = 7$ y $m = 12$, o $m \geq \lfloor \frac{(n+1)^2}{4} \rfloor$.

La conjetura de Brown *et al.*, formulada en 2020, propone que en cada una de las clases de grafos $T^d_{n,m}$ tales que $d \geq 4$, $n \geq 2(d-1)$ y $2d \leq m \leq (d-2)\lfloor \frac{n}{d-1} \rfloor^2 + \lfloor \frac{n}{d-1} \rfloor - 1$ no existe un grafo que sea uniformemente más confiable [3].

Recientemente, Miranda [5] demostró que la conjetura de Brown es cierta cuando $d = 4$. La esencia de la demostración consiste en mostrar que cada uno de los grafos en $T^4_{n,m}$ que son localmente más confiables cerca de $p = 1$ no son localmente más confiables cerca de $p = 0$. Puesto que los grafos uniformemente más confiables cerca de $p = 1$ en $T^4_{n,m}$ deben maximizar la cantidad de caminos de largo 4

entre los terminales de G, Miranda incluye en [5] un problema de optimización no lineal de variables enteras que tiene como objetivo la maximización de caminos de largo 4 dentro de todos los grafos en $T^4_{n,m}$ como un programa de optimización no lineal entera. Luego determina clases de soluciones de dicho problema que se corresponden con grafos con dos terminales, y mediante una relación de dominación de clases de soluciones factibles finalmente demuestra que ningún grafo en $T^4_{n,m}$ (con las elecciones de enteros n y m definidos en la conjetura de Brown) puede ser simultáneamente localmente más confiable cerca de $p = 0$ y de $p = 1$.

En este trabajo nos enfocamos en el caso $d = 5$, donde la conjetura de Brown afirma que en cada clase $T^5_{n,m}$ tal que $n \geq 8$ y $10 \leq m \leq 3\lfloor \frac{n}{4} \rfloor^2 + \lfloor \frac{n}{4} \rfloor - 1$ no existe un grafo que sea uniformemente más confiable [3]. Estudiamos la conjetura de Brown de inexistencia mediante tres enfoques complementarios: (i) extendemos las técnicas de Miranda mediante generalizaciones del problema de optimización a distancias mayores que 4, (ii) desarrollamos un método de búsqueda local basado en Vecindarios Variables (VNS) que explora transformaciones de grafos para maximizar $N_4(G)$, y (iii) proporcionamos validación computacional exhaustiva para familias pequeñas de grafos.

El presente trabajo se organiza de la siguiente manera. En la sección 2 presentamos los conceptos básicos y antecedentes necesarios, incluyendo las definiciones fundamentales de confiabilidad en grafos con dos terminales, las condiciones necesarias de Miranda para confiabilidad local, la estructura de capas y la conjetura de Brown junto con algunos conceptos presentados por Miranda en [5] para su prueba de inexistencia de UMR cuando $d = 4$. En la sección 3 planteamos el problema para $d = 5$, formulando la maximización de $N_4(G)$ como un programa de optimización no lineal entera y analizando la estructura de los grafos candidatos a ser UMR. En la sección 4 enunciamos la conjetura de inexistencia para $d = 5$ y desarrollamos heurísticas de búsqueda local basada en VNS. Finalmente, en la sección 5 presentamos los resultados computacionales obtenidos, que validan la conjetura para familias finitas de grafos.

2. Conceptos básicos y antecedentes

En esta sección presentamos las definiciones fundamentales relacionadas con la teoría de confiabilidad en grafos con dos terminales, adaptadas del trabajo de Miranda [5] y Brown *et al.* [3], introduciendo la notación que utilizaremos a lo largo del trabajo. Posteriormente, revisamos los resultados principales de ambos trabajos que motivan nuestro estudio.

Comenzamos en la sección 2.1 con las definiciones básicas del modelo de confiabilidad en grafos con dos terminales, estableciendo la notación que utilizaremos a lo largo del documento. En la sección 2.2 introducimos las condiciones necesarias de Miranda [5] para que un grafo sea localmente más confiable en los extremos del intervalo [0,1]. En la sección 2.3 describimos la estructura de capas de los grafos en $T^d_{n,m}$ y el concepto de elementos irrelevantes. En la sección 2.4 presentamos el Teorema de Menger y definimos el grafo D^d_k, herramientas fundamentales en la teoría [5]. Finalmente, en las Secciones 2.5 y 2.6 enunciamos la conjetura de Brown junto con algunos conceptos presentados por Miranda en [5] para su prueba de inexistencia de UMR cuando $d = 4$, y describimos la estrategia de demostración que motiva nuestro enfoque para $d = 5$.

2.1. Definiciones básicas

Definición 2.1 (*st*-pathset)**.** Sea G un grafo con dos terminales en $T^d_{n,m}$. Un conjunto $S \subseteq V(G) \setminus \{s, t\}$ es un *st-pathset* si el subgrafo inducido por $S \cup \{s, t\}$ contiene al menos un camino de s a t de largo d. Si $|S| = i$, decimos que S es un *st*-pathset de tamaño i.

Definición 2.2 (Polinomio de confiabilidad)**.** Sea $G \in T^d_{n,m}$. Para cada $i \in \{0, 1, \ldots, n\}$, sea $N_i(G)$ el número de *st*-pathsets de tamaño i. El *polinomio de confiabilidad* de G se define como:

$$NR_G(p) = \sum_{i=0}^{n} N_i(G) p^i (1-p)^{n-i}$$

donde $1 - p$ es la probabilidad de falla de cada vértice no terminal de G.

Definición 2.3 (Grafo uniformemente más confiable (UMR))**.** Un grafo $G \in T^d_{n,m}$ es *uniformemente más confiable* (UMR) si para todo $H \in T^d_{n,m}$ y todo $p \in [0, 1]$ se

cumple:

$$NR_G(p) \geq NR_H(p).$$

Definición 2.4 (Grafo localmente más confiable)**.** Un grafo $G \in T^d_{n,m}$ es *localmente más confiable en un entorno de* $p_0 \in [0,1]$ si existe $\delta > 0$ tal que para todo $H \in T^d_{n,m}$ y todo $p \in (p_0 - \delta, p_0 + \delta) \cap [0,1]$ se cumple:

$$NR_G(p) \geq NR_H(p).$$

2.2. Condiciones necesarias para confiabilidad local

Miranda [5] proporciona condiciones necesarias para que un grafo con dos terminales sea localmente más confiable en los extremos del intervalo $[0,1]$. Estas condiciones se basan en las siguientes secuencias de conjuntos anidados.

Definición 2.5 ([5])**.** Para cada clase no vacía $T^d_{n,m}$ se define la secuencia de conjuntos anidados ${}_dL^0_{n,m}, {}_dL^1_{n,m}, \ldots, {}_dL^n_{n,m}$ de la siguiente manera:

1. ${}_dL^0_{n,m} = T^d_{n,m}$
2. Para cada $i \in \{1, \ldots, n\}$:

$${}_dL^i_{n,m} = \left\{ G \in {}_dL^{i-1}_{n,m} : N_i(G) = \max_{H \in {}_dL^{i-1}_{n,m}} N_i(H) \right\}$$

Lema 2.1 ([6])**.** *Un grafo $G \in T^d_{n,m}$ es localmente más confiable en un entorno de $p = 0$ si y solo si $G \in {}_dL^n_{n,m}$.*

Definición 2.6 ([5])**.** Para cada clase no vacía $T^d_{n,m}$ se define la secuencia de conjuntos anidados ${}_dU^n_{n,m}, {}_dU^{n-1}_{n,m}, \ldots, {}_dU^0_{n,m}$ de la siguiente manera:

1. ${}_dU^n_{n,m} = T^d_{n,m}$
2. Para cada $i \in \{n-1, n-2, \ldots, 0\}$:

$${}_dU^i_{n,m} = \left\{ G \in {}_dU^{i+1}_{n,m} : N_i(G) = \max_{H \in {}_dU^{i+1}_{n,m}} N_i(H) \right\}$$

Lema 2.2 ([5])**.** *Un grafo $G \in T^d_{n,m}$ es localmente más confiable en un entorno de $p = 1$ si y solo si $G \in {}_dU^0_{n,m}$.*

2.3. Estructura de capas y elementos irrelevantes

Para analizar la estructura de los grafos en $T^d_{n,m}$, es conveniente particionar los vértices en capas según su distancia a los terminales.

Definición 2.7 ([5])**.** Para cada grafo $G \in T^d_{n,m}$ y cada $j \in \{1, \ldots, d-1\}$, definimos la *capa* j de G, denotada $V_j(G)$, como:

$V_j(G) = \{v \in V(G) : d(s, v) = j$ y v pertenece a algún camino de largo d entre s y $t\}$.

Definición 2.8 ([5])**.** Sea $G \in T^d_{n,m}$. Un vértice o arista de G es *irrelevante* si no pertenece a ningún camino de largo d que une los terminales de G. Denotamos al conjunto de vértices y aristas irrelevantes de G como $V^*(G)$ y $E^*(G)$, respectivamente.

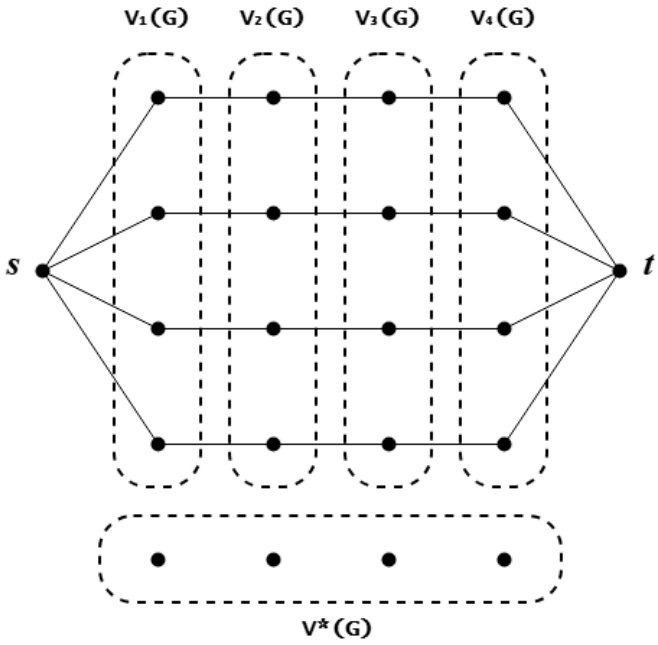

Figura 1: Capas $V_j(G)$ y vértices irrelevantes

2.4. Conectividad y el grafo D^d_k

Una herramienta fundamental en el análisis de confiabilidad es el Teorema de Menger, que relaciona la conectividad con el número de caminos disjuntos.

Teorema 2.3 (Teorema de Menger [2])**.** *Para todo grafo G con terminales s y t no adyacentes, la conectividad $\kappa(G)$ es igual a la máxima cantidad de caminos internamente disjuntos de s a t.*

La siguiente definición introduce el grafo D_k^d, que consiste en k caminos disjuntos de largo d entre los terminales.

Definición 2.9. Para cada entero positivo k y cada entero $d \geq 2$, definimos D_k^d como el grafo con dos terminales tal que:

$$V(D_k^d) = \{s, t\} \cup \bigcup_{i=1}^{d-1} \{v_{1,i}, v_{2,i}, \ldots, v_{k,i}\}$$

$$E(D_k^d) = \{sv_{j,1}, v_{j,1}v_{j,2}, \ldots, v_{j,d-2}v_{j,d-1}, v_{j,d-1}t : j \in \{1, \ldots, k\}\}$$

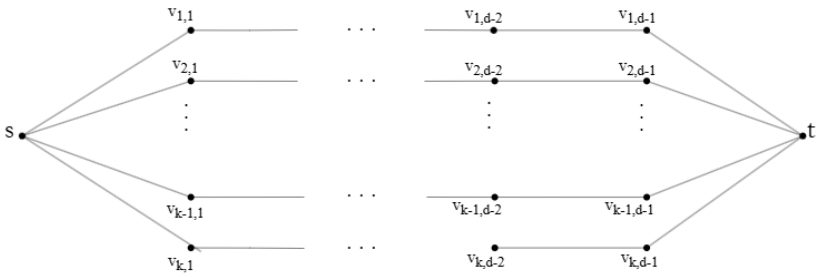

Figura 2: Grafo D_k^d

Observación 2.1. El grafo D_k^d consiste en k caminos disjuntos de largo d conectando los terminales s y t. Por el Teorema de Menger, $\kappa(D_k^d) = k$.

Definimos la función $k_d(n, m)$ que representa la máxima cantidad de caminos disjuntos de largo d que un grafo en $T_{n,m}^d$ puede contener.

Definición 2.10. Para cada dupla (n, m) con $n, m \in \mathbb{Z}^+$, definimos:

$$k_d(n, m) = \text{mín} \left\{ \left\lfloor \frac{n}{d-1} \right\rfloor, \left\lfloor \frac{m}{d} \right\rfloor \right\}$$

A partir de esta función, podemos establecer una cota superior para la conectividad de los grafos en $T_{n,m}^d$ y caracterizar aquellos que alcanzan dicha cota.

Teorema 2.4 ([5]). *Para cada $G \in T^d_{n,m}$ se cumple que $\kappa(G) \leq k_d(n,m)$, y la igualdad ocurre si y solo si G incluye a $D^d_{k_d(n,m)}$ como subgrafo.*

Como se mencionó en la sección anterior, los grafos en ${}_dU^1_{n,m}$ son aquellos que son uniformemente más confiables para $p = 1$. En su trabajo, Miranda demuestra que estos grafos deben tener máxima conectividad.

Teorema 2.5 ([5]). *Todo grafo con dos terminales en ${}_dU^1_{n,m}$ tiene máxima conectividad.*

Este resultado, junto con el anterior, nos lleva a la conclusión de que los grafos en ${}_dU^1_{n,m}$ deben contener a $D^d_{k_d(n,m)}$ como subgrafo. Esto motiva la definición del siguiente conjunto de grafos.

Definición 2.11. Para cada $n, m, d \in \mathbb{Z}^+$, $d \geq 2$, definimos el conjunto $\mathcal{M}^d_{n,m}$ como $\mathcal{M}^d_{n,m} := \{G \in T^d_{n,m} : G \text{ incluye a } D^d_{k_d(n,m)} \text{ como subgrafo}\}$

Finalmente, podemos enunciar el teorema que establece que los grafos localmente más confiables cerca de $p = 1$ deben pertenecer a $\mathcal{M}^d_{n,m}$.

Teorema 2.6. *Para cada $n, m, d \in \mathbb{Z}^+$, $d \geq 2$, se cumple que ${}_dU^1_{n,m} \subseteq \mathcal{M}^d_{n,m}$.*

2.5. La conjetura de Brown y el resultado de Miranda

Conjetura 1 (Brown, [3]). *Para cada terna de enteros n, m y d tales que $d \geq 4$, $n \geq 2(d-1)$ y $2d \leq m \leq (d-2)\lfloor \frac{n}{d-1} \rfloor^2 + \lfloor \frac{n}{d-1} \rfloor - 1$ no existe grafo UMR en $T^d_{n,m}$.*

Teorema 2.7 (Miranda, [5]). *La conjetura de Brown es verdadera para $d = 4$.*

2.6. Estrategia de demostración de Miranda

La demostración de Miranda para $d = 4$ se basa en mostrar que los conjuntos ${}_4L^n_{n,m}$ (grafos localmente más confiables cerca de $p = 0$) y ${}_4U^1_{n,m}$ (grafos localmente más confiables cerca de $p = 1$) son disjuntos para aquellas duplas de enteros n y m del enunciado de la conjetura de Brown. La estrategia central consiste en:

1. **Estructura de grafos localmente más confiables cerca de** $p = 1$: Mostrar que todo grafo localmente más confiable cerca de $p = 1$ debe contener a $D^d_{k_d(n,m)}$ como subgrafo, y por lo tanto, pertenece a $\mathcal{M}^d_{n,m}$.

2. **Maximizar** N_{d-1}: Formular el problema de maximizar $N_{d-1}(G)$ sobre $T^d_{n,m}$ como un problema de optimización no lineal, aprovechando la estructura de capas.

3. **Incompatibilidad de** ${}_dL^d_{n,m}$ **y** ${}_dU^1_{n,m}$: Se construye una cadena de conjuntos anidados que, partiendo de $\mathcal{M}^d_{n,m}$, permite demostrar que ${}_dL^n_{n,m}$ y ${}_dU^1_{n,m}$ son disjuntos. Por detalles de esta construcción recomendamos consultar [5].

Para $d = 4$, Miranda formula el problema de maximizar $N_3(G)$ aprovechando que los st-pathsets de tamaño 3 corresponden a ternas de vértices en las capas intermedias que forman caminos de s a t. Esta formulación se extiende naturalmente a $d = 5$ considerando 4-tuplas, lo que motiva nuestro enfoque en la siguiente sección.

3. Planteo del problema de optimización para $d = 5$

Dado el problema de demostrar incompatibilidad entre los grafos localmente más confiables en 1 y los grafos localmente más confiables en 0, es natural preguntarse qué estructura debe tener un grafo $G \in T^5_{n,m}$ que es localmente más confiable en $p = 1$. Por la condición necesaria de Miranda, sabemos que tal grafo debe contener a $D^5_{k_5(n,m)}$ como subgrafo, es decir, debe tener al menos $k_5(n, m)$ caminos s-t disjuntos.

Luego, una pregunta central es: ¿puede un grafo que contiene a $D^5_{k_5(n,m)}$ ser simultáneamente localmente más confiable cerca de $p = 0$ y de $p = 1$? Para responder esto, es necesario mostrar si existen o no grafos en $T^5_{n,m}$ que no contienen a $D^5_{k_5(n,m)}$ pero que logran un valor mayor de $N_4(G)$ que cualquier grafo en $T^5_{n,m}$ que contiene a $D^5_{k_5(n,m)}$. Formalizamos esta idea en la siguiente proposición, cuyo enunciado y demostración provienen de adaptar resultados del Proyecto de Grado de Miranda [5]. La proposición establece un criterio para demostrar que no existen grafos simultáneamente más confiables en ambos extremos del intervalo $[0, 1]$.

Proposición 3.1 (Criterio de inexistencia de grafos UMR en $T^5_{n,m}$)**.** *Sea* (n, m) *una dupla tal que* $T^5_{n,m} \neq \emptyset$*. Si para todo* $G \in \mathcal{M}^5_{n,m}$ *existe* $H \in T^5_{n,m}$ *con*

$$N_4(H) > N_4(G),$$

entonces no existe grafo UMR en $T^5_{n,m}$.

Demostración. Sean n y m dos enteros cualquiera tales que $T^5_{n,m}$ es no vacío. Supongamos que G un grafo uniformemente más confiable en $T^5_{n,m}$ que cumple con todas las condiciones del enunciado. Como G es uniformemente más confiable, G es localmente más confiable tanto en $p = 0$ como en $p = 1$. Por un lado, como G es localmente más confiable en $p = 1$, por el Teorema 2.4 tenemos que G pertenece al conjunto $\mathcal{M}^5_{n,m}$. Por otro lado, como G es localmente más confiable en $p = 0$, entonces por el Lema 2.1 y la Definición 2.5, G maximiza N_4 dentro de $T^5_{n,m}$. Pero la hipótesis asegura que todo grafo de $\mathcal{M}^5_{n,m}$ es estrictamente superado en N_4 por algún grafo de $T^5_{n,m}$. Hemos obtenido una contradicción, que proviene de haber supuesto inicialmente que G es un grafo uniformemente más confiable en $T^5_{n,m}$ que cumple con todas las condiciones del enunciado. Por lo tanto, bajo las condiciones del enunciado no existe ningún grafo que sea uniformemente más confiable en $T^5_{n,m}$, que es lo que queríamos demostrar. □

La Proposición 3.1 nos permite ver que es de interés formular el problema de maximizar N_4 dentro de una familia de grafos $T^5_{n,m}$, puesto que dicho problema encapsula el mecanismo mediante el cual podríamos encontrar grafos en $T^5_{n,m}$ que alcanzan valores de N_4 superiores a los de cualquier grafo en $\mathcal{M}^5_{n,m}$. En tal caso, estaríamos en condiciones de aplicar la Proposición 3.1 para concluir que no existen grafos UMR en $T^5_{n,m}$, lo que constituiría evidencia hacia la conjetura de Brown en la distancia $d = 5$.

3.1. Motivación y estrategia

Siguiendo el enfoque de Miranda, el objetivo de esta sección es formular el problema de maximizar $N_4(G)$ dentro de la clase $T^5_{n,m}$ como un problema de optimización no lineal. Recordemos que maximizar $N_4(G)$ se corresponde a maximizar la cantidad de st-pathsets de tamaño 4, lo que es una condición necesaria para que un grafo sea localmente más confiable en un entorno de $p = 0$. Por otro lado, los grafos localmente más confiables en un entorno de $p = 1$ deben contener a $D^5_{k_5(n,m)}$ como subgrafo, lo que implica una conectividad mínima entre los terminales. Formulando el problema de maximizar $N_4(G)$, encapsulamos el mecanismo a través del cual podría demostrarse la incompatibilidad entre los conjuntos $L^n_{n,m}$ y $U^1_{n,m}$

para $d = 5$, lo que a su vez contribuiría a la demostración de la conjetura de Brown para esta distancia.

Para un grafo $G \in T^5_{n,m}$, los st-pathsets de tamaño 4 corresponden a conjuntos de 4 vértices no terminales que forman un camino de largo exactamente 5 entre los terminales. Debido a la distancia fija $d(s,t) = 5$, estos 4 vértices deben estar distribuidos en las capas $V_1(G), V_2(G), V_3(G)$ y $V_4(G)$ respectivamente, donde $V_i(G)$ denota el conjunto de vértices a distancia i de s que pertenecen a algún camino de largo 5.

La estrategia consiste en:

1. Particionar los vértices según las capas definidas por su distancia a s.
2. Contar el número de st-pathsets de tamaño 4 mediante variables que representan las conexiones entre capas consecutivas.
3. Expresar la suma total como un producto de vectores y matrices que capturan la estructura del grafo.
4. Formular restricciones sobre el número total de vértices y aristas disponibles.

3.2. Formulación del problema de optimización

A continuación, traducimos la estrategia anterior en variables matemáticas concretas. Introducimos un sistema de partición de vértices en capas y definimos variables que capturan las conectividades entre dichas capas. Estas variables nos permitirán expresar $N_4(G)$ como un producto de vectores y matrices, facilitando la formulación del problema de optimización.

Consideremos un grafo $G \in T^5_{n,m}$ y particionemos sus vértices no terminales en cuatro capas:

- $V_1(G) = \{v_{1,1}, v_{2,1}, \ldots, v_{n_1,1}\}$: vértices a distancia 1 de s
- $V_2(G) = \{v_{1,2}, v_{2,2}, \ldots, v_{n_2,2}\}$: vértices a distancia 2 de s
- $V_3(G) = \{v_{1,3}, v_{2,3}, \ldots, v_{n_3,3}\}$: vértices a distancia 3 de s
- $V_4(G) = \{v_{1,4}, v_{2,4}, \ldots, v_{n_4,4}\}$: vértices a distancia 4 de s

Todos los vértices en $V_i(G)$ están conectados a t mediante al menos un camino de largo $5-i$ (es decir, pertenecen a algún camino s-t de largo 5).

Sea $i \in \{1, ..., |V_2|\}$, para cada vértice $v_{i,2} \in V_2(G)$, definimos:

$$y_i = |\{v \in V_1(G) : vv_{i,2} \in E(G)\}|$$

Es decir, y_i es el número de vecinos en $V_1(G)$ de $v_{i,2}$.

Para cada par de vértices $v_{i,2} \in V_2(G)$ y $v_{j,3} \in V_3(G)$, definimos la variable binaria:

$$x_{ij} = \begin{cases} 1 & \text{si } v_{i,2}v_{j,3} \in E(G) \\ 0 & \text{en caso contrario} \end{cases}$$

La matriz $X = (x_{ij})$ es de tamaño $n_2 \times n_3$ y captura la estructura de aristas entre las capas 2 y 3.

Sea $j \in \{1, ..., |V_3|\}$, para cada vértice $v_{j,3} \in V_3(G)$, definimos:

$$z_j = |\{v \in V_4(G) : v_{j,3}v \in E(G)\}|$$

Es decir, z_j es el número de vecinos en $V_4(G)$ de $v_{j,3}$.

El número total de st-pathsets de tamaño 4 es:

$$N_4(G) = \sum_{i=1}^{n_2} \sum_{j=1}^{n_3} y_i x_{ij} z_j = \mathbf{y}^T X \mathbf{z}$$

donde $\mathbf{y} = (y_1, \ldots, y_{n_2})^T \in \mathbb{R}^{n_2}$ y $\mathbf{z} = (z_1, \ldots, z_{n_3})^T \in \mathbb{R}^{n_3}$ son vectores de cardinalidades, y $X \in \{0,1\}^{n_2 \times n_3}$ es la matriz de adyacencia entre capas.

3.3. Interpretación de variables y dominio

Contemplamos los siguientes coeficientes, adaptados del trabajo de Miranda [5]:

- n_1, n_2, n_3, n_4: número de vértices en cada capa relevante.
- n^*: número de vértices irrelevantes (aquellos que no pertenecen a ningún camino s-t de largo 5).
- m^*: número de aristas irrelevantes (aquellas que no pertenecen a ningún camino s-t de largo 5).

- $\mathbf{y} = (y_1, \ldots, y_{n_2})$: vector de conectividad entre capas 1 y 2.
- X: matriz de adyacencia entre capas 2 y 3.
- $\mathbf{z} = (z_1, \ldots, z_{n_3})$: vector de conectividad entre capas 3 y 4.

3.4. Restricciones del problema

El problema de optimización busca maximizar $N_4(G)$ sujeto a restricciones sobre el número total de vértices y aristas. Formalmente:

$$\begin{aligned}
&\max_{G \in T^5_{n,m}} \quad N_4(G) = \mathbf{y}^T X \mathbf{z} \\
&\text{s.t.:} \\
&(1) \quad n_1 + n_2 + n_3 + n_4 + n^* = n \\
&(2) \quad n_1 + \sum_{i=1}^{n_2} y_i + \sum_{i=1}^{n_2} \sum_{j=1}^{n_3} x_{ij} + \sum_{j=1}^{n_3} z_j + n_4 + m^* = m \\
&(3) \quad x_{ij} \in \{0,1\}, \quad \forall i \in \{1, \ldots, n_2\}, j \in \{1, \ldots, n_3\} \\
&(4) \quad y_i \in \{1, \ldots, n_1\}, \quad \forall i \in \{1, \ldots, n_2\} \\
&(5) \quad z_j \in \{1, \ldots, n_4\}, \quad \forall j \in \{1, \ldots, n_3\} \\
&(6) \quad n_1, n_2, n_3, n_4, n^*, m^* \in \mathbb{N}
\end{aligned} \tag{1}$$

La restricción (1) asegura que todos los vértices no terminales están contabilizados: aquellos en las cuatro capas relevantes más aquellos irrelevantes.

La restricción (2) contabiliza las aristas: aquellas entre capas consecutivas (desde s a V_1, V_1 a V_2, de V_2 a V_3, de V_3 a V_4, y de V_4 a t), más aquellas que son irrelevantes.

La restricción (3) definen el dominio binario de la matriz X.

Las restricciones (4)-(5) definen que cada vértice en las capas intermedias debe tener al menos una conexión con sus capas adyacentes para ser relevante.

4. Conjetura de inexistencia para d = 5

En esta sección enunciamos una conjetura de inexistencia de grafos UMR en $T^5_{n,m}$ y desarrollamos un algoritmo de búsqueda local basado en Variable Neighborhood Search (VNS) con el que buscamos verificar la conjetura computacionalmente para distintas familias $T^5_{n,m}$.

En la sección 4.1 presentamos y desarrollamos la intuición para ciertas heurísticas de mejora para el problema de optimización 1, describiendo en alto nivel las transformaciones estructurales que realizan sobre un grafo $G \in T^5_{n,m}$ y cómo estas transformaciones contribuyen a aumentar el valor de $N_4(G)$. Luego, en la sección 4.2 presentamos el algoritmo VNS propiamente dicho, definiendo los vecindarios utilizados, la función de evaluación y el criterio de parada. Finalmente, en la sección 4.3 ilustramos el funcionamiento del algoritmo sobre el caso concreto $T^5_{12,20}$, explicitando qué vecindario actúa en cada etapa y cuál es el efecto de la transformación realizada sobre el grafo.

Sea $k = \lfloor \frac{n}{4} \rfloor$. Enunciamos la siguiente conjetura.

Conjetura 2. *Sean n y m enteros positivos tales que $n > 8$ y $10 \leq m \leq 3k^2 + 2k - 3$, donde $k = \lfloor \frac{n}{4} \rfloor$. Entonces no existe grafo UMR en $T^5_{n,m}$.*

Observación 4.1. Notamos que como para todo $n \geq 8$ se cumple que $3k^2+2k-3 \geq 3k^2 + k - 1$, entonces una demostración de la conjetura anterior implica también que la conjetura de Brown se cumple cuando $d = 5$.

Definición 4.1. Definimos el conjunto I de las duplas (n, m) que cumplen las hipótesis de la Conjetura 2 de la siguiente manera:

$$I = \{(n, m) \in \mathbb{Z}^2 : n > 8,\ 10 \leq m \leq 3k^2 + 2k - 3,\ k = \lfloor \frac{n}{4} \rfloor\}$$

4.1. Heurísticas para aumento de $N_4(G)$

Basados en los conjuntos anidados definidos por Miranda [5, sección 3.3], proponemos un conjunto de heurísticas de mejora local dirigidas a maximizar $N_4(G)$ en grafos pertenecientes a $T^5_{n,m}$ donde (n, m) pertenece a I. Recordemos que el valor de $N_4(G)$ se calcula como la suma de los productos $y_i x_{ij} z_j$ sobre todos los pares de vértices $(v_{i,2}, v_{j,3})$ de las capas 2 y 3, es decir, $N_4(G) = \sum_{i=1}^{n_2} \sum_{j=1}^{n_3} y_i x_{ij} z_j$.

Cada término $y_i x_{ij} z_j$ de esta suma representa la cantidad de caminos de largo 5 que pasan a través de la arista que conecta el vértice $v_{i,2}$ de la capa 2 con el vértice $v_{j,3}$ de la capa 3 (si es que tal arista existe, i.e., $x_{ij} = 1$).

Antes de desarrollar las heurísticas que buscan maximizar esta suma mediante transformaciones en la estructura del grafo, introducimos un concepto de interés.

Definición 4.2 (Dupla justa)**.** Una dupla $(x, y) \in \mathbb{Z}^2$ es justa si y solo si $|x-y| \leq 1$.

Heurísticas

1. **Aprovechamiento de aristas irrelevantes (Algoritmo 2, Aprovechar Aristas Irrelevantes):** Las aristas irrelevantes, por definición, no forman parte de ningún camino s-t de largo 5 y, por lo tanto, no contribuyen al valor de $N_4(G)$. Esta heurística reubica estas aristas entre las capas relevantes del grafo. Al agregar una arista entre la capa 1 y la capa 2, por ejemplo, se puede incrementar el valor de algún y_i, lo que a su vez aumenta los términos $y_i x_{ij} z_j$ para todos los j a los que el vértice i está conectado. De manera similar, agregar aristas entre las capas 2 y 3 (aumentando las entradas de X) o entre las capas 3 y 4 (aumentando algún z_j) crea nuevos sumandos o aumenta el valor de los existentes en la fórmula de $N_4(G)$.

2. **Balanceo de cantidad de vértices en capas externas (Algoritmo 3, Ajustar Capas Externas):** Los valores n_1 y n_4 (tamaños de las capas 1 y 4) actúan como un límite superior para los grados y_i y z_j, respectivamente. Es decir, $y_i \leq n_1$ y $z_j \leq n_4$. Para que los productos $y_i z_j$ sean grandes, es necesario que tanto y_i como z_j puedan tomar valores altos. Si una de las capas extremas es muy pequeña (e.g., n_1 es pequeño), todos los y_i estarán limitados, restringiendo el valor máximo de cada término en la suma de $N_4(G)$. Al balancear n_1 y n_4 y lograr que la dupla (n_1, n_2) sea justa, se busca crear un escenario donde tanto los y_i como los z_j tengan el potencial de ser grandes, permitiendo así mayores productos $y_i z_j$.

3. **Balanceo de tuplas conectivas (Algoritmo 4, Ajustar Entradas y Salidas):** Como se explicó, el valor de $N_4(G)$ es la suma de los términos $y_i x_{ij} z_j$. Para maximizar esta suma, es deseable que los productos individuales $y_i z_j$ sean lo más grandes posible para los pares (i, j) donde hay una conexión

($x_{ij} = 1$). Dado que el producto de dos números con suma fija se maximiza cuando ambos son iguales o casi iguales, esta heurística propone que para cada $i \in [n_2]$ exista un $j \in [n_3]$ tal que la dupla (y_i, z_j) sea justa y además $x_{ij} = 1$.

4. **Concentración selectiva de grados (Algoritmo 5, Min-Max):** Esta heurística busca explotar la estructura de la suma $N_4(G) = \sum_i y_i(\sum_j x_{ij} z_j)$. Si un vértice $v_{i,2}$ tiene un grado de entrada y_i particularmente alto, el impacto de ese y_i en la suma total se magnifica si se conecta a vértices $v_{j,3}$ que a su vez tienen grados de salida z_j altos. Al crear variabilidad en el vector $\mathbf{z}$ (algunos z_j grandes y otros pequeños), se puede "dirigir"la conectividad de los vértices con y_i altos hacia los vértices con z_j altos, maximizando así los productos $y_i z_j$ para esos pares y aumentando significativamente la suma total.

5. **Aumento de cantidad de vértices en capa 1 (Algoritmo 6, Aumentar Capa Uno):** Esta heurística busca aumentar el tamaño de la capa 1 hasta que $|V_1| \geq k_5(n, m) + 1$, tomando vértices poco acoplados de las capas 3 o 4 y reubicándolos en la capa 1, redistribuyendo sus aristas para maximizar las conexiones hacia la capa 2. Esto tiene un doble efecto: por un lado, elevar n_1 incrementa el límite superior de los valores y_i, lo que permite mayores productos $y_i z_j$ y por tanto un mayor $N_4(G)$; por otro, al dejar alguna capa con menos de $k_5(n, m)$ vértices, el grafo resultante deja de contener a $D^5_{k_5(n,m)}$ como subgrafo, con lo cual no puede ser localmente más confiable cerca de $p = 1$. La heurística no está orientada a producir mejoras inmediatas grandes en N_4, sino a deshacer estructuras que bloquean la acción de las demás heurísticas para que puedan continuar optimizando.

Complementando las heurísticas anteriores, introducimos un algoritmo adicional que opera de manera más global. A diferencia de las heurísticas previas que realizan mejoras locales incrementales, este algoritmo identifica la configuración óptima de la matriz X (aristas entre las capas 2 y 3) dada una cantidad fija de aristas entre dichas capas, y realiza una transformación directa hacia esa configuración óptima.

Sea $G \in T^5_{n,m}$ un grafo actual y sea m_{23} el número de aristas presentes entre las capas $V_2(G)$ y $V_3(G)$. El algoritmo **Optimizar Matriz X** opera de la siguiente manera:

1. **Extracción de parámetros:** Se extrae del grafo G el número de aristas m_{23}, así como los vectores de grado $\mathbf{y}$ y $\mathbf{z}$ en sus formas actuales.

2. **Identificación del óptimo local:** Dado m_{23}, se resuelve mediante búsqueda exhaustiva el subproblema de encontrar la matriz binaria $X^* \in \{0,1\}^{n_2 \times n_3}$ que maximiza:

$$\mathbf{y}^T X^* \mathbf{z}$$

   sujeto a que la suma de todas las entradas de X^* es igual a m_{23}.

3. **Generación del grafo vecino:** Se construye un nuevo grafo $G' \in T^5_{n,m}$ que preserva la estructura de capas y los vectores $\mathbf{y}$ y $\mathbf{z}$, pero reemplaza la matriz de adyacencia entre capas 2 y 3 por X^*. Esto implica eliminar todas las aristas entre $V_2(G)$ y $V_3(G)$ y agregar nuevas aristas de acuerdo a la matriz X^*.

Algoritmo 1: Optimizar Matriz X

Entrada: Grafo $G \in T^5_{n,m}$

Salida : Grafo $G' \in T^5_{n,m}$

Calcular vectores y, z y matriz X, del grafo G;

$m' \leftarrow$ cantidad de aristas entre $V_2(G)$ y $V_3(G)$;

`// Ordenar todas las duplas por su producto`

$P \leftarrow \{(i, j, y_i \cdot z_j) : i \in |V_2(G)|, j \in |V_3(G)|\}$;

Ordenar P en orden decreciente por la tercera componente;

`// Crear nueva matriz X asignando aristas a los mejores productos`

$X' \leftarrow$ matriz de ceros de tamaño $|V_2(G)| \times |V_3(G)|$;

for $k \leftarrow 1$ **to** m' **do**

  $(i, j, \_) \leftarrow P[k]$;

  $X'[i][j] \leftarrow 1$;

end

`// Reconstruir grafo con la nueva matriz X'`

Grafo $G' \leftarrow$ Copia de grafo G;

Eliminar todas las aristas entre $V_2(G')$ y $V_3(G')$;

for $i \in |V_2(G)|, j \in |V_3(G)|$ **do**

  if $X'[i][j] = 1$ **then**

    Agregar arista $(V_2(G)[i], V_3(G)[j])$ a G';

  end

end

return grafo modificado G';

4.2. Estrategia de búsqueda local mediante Vecindarios Variables (VNS)

Integramos las heurísticas anteriores en un algoritmo de búsqueda local basado en *Variable Neighborhood Search* (VNS). La estrategia consiste en:

1. **Definición de vecindarios:** Definimos un conjunto $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_6\}$ de vecindarios, donde cada $\mathcal{B}_i$ corresponde a una de las heurísticas propuestas

anteriormente. Para cada vecindario $\mathcal{B}_i$, se define una función $\phi_i : T^5_{n,m} \to T^5_{n,m}$ que genera un grafo vecino a partir del grafo actual.

2. **Función de evaluación:** Se implementa una función de evaluación $f(G) = N_4(G)$ que calcula el número de st-pathsets de tamaño 4 para un grafo dado $G \in T^5_{n,m}$.

3. **Búsqueda iterativa:** El algoritmo VNS procede de la siguiente manera: comenzando desde un grafo inicial $G_0 \in T^5_{n,m}$, se intenta mejorar secuencialmente visitando los vecindarios en orden. Para cada vecindario $\mathcal{B}_i$, se genera un grafo vecino $G' = \phi_i(G)$ y se evalúa si $f(G') \geq f(G)$. Si se encuentra una mejora (o equivalencia), se actualiza $G \leftarrow G'$ y se reinicia la búsqueda desde el primer vecindario. Si no se encuentra mejora en el vecindario actual, se avanza al siguiente vecindario $\mathcal{B}_{i+1}$.

4. **Criterio de parada:** El algoritmo termina cuando se ha visitado la totalidad de los vecindarios sin encontrar mejora alguna, lo que indica un óptimo local respecto de la estructura de vecindarios considerada.

5. **Flexibilidad en transformaciones:** A diferencia del algoritmo VNS encontrado en [1], permitimos que el algoritmo acepte transformaciones con $f(G') = f(G)$ (es decir, transformaciones que no incrementan estrictamente el valor de N_4). Esta estrategia favorece la exploración de diferentes regiones del espacio de soluciones y puede ayudar a escapar de óptimos locales estrictos, aumentando la diversidad de grafos explorados.

El objetivo de este procedimiento es demostrar que para los casos considerados, los grafos que contienen a $D^5_{k_5(n,m)}$ como subgrafo no son óptimos en $N_4(G)$. Para ello, se busca exhibir que para cada uno de estos grafos, existe otro grafo en $T^5_{n,m}$ con un valor de $N_4(G)$ superior. De esta forma, se prueba que ningún grafo localmente más confiable en $p = 1$ alcanza el valor máximo de N_4 dentro de su clase, y por tanto, ninguno de estos grafos es localmente más confiable en $p = 0$.

A continuación se presenta un diagrama que ilustra el flujo de ejecución del algoritmo VNS, seguido del Cuadro 1, que resume los vecindarios utilizados y el orden en que son visitados durante cada iteración.

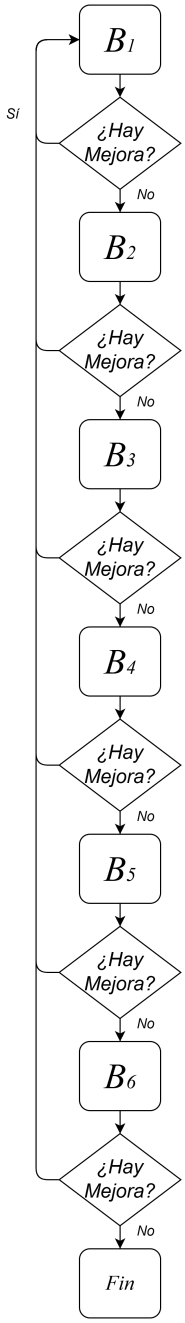

Figura 3: Diagrama de VNS

i	Vecindario
1	Aprovechar Aristas Irrelevantes
2	Ajustar Capas Externas
3	Ajustar Entradas y Salidas
4	Optimizar Matriz X
5	Min-Max
6	Aumentar Capa Uno

Cuadro 1: Orden de aplicación de vecindarios

4.3. Ejemplo: Aplicación de VNS al caso $T^5_{12,20}$

En esta subsección ilustramos el funcionamiento del algoritmo VNS sobre un ejemplo concreto. La idea no es solo mostrar el grafo inicial y el grafo final, sino también explicitar qué vecindario actúa en cada etapa y cuál es el efecto estructural de la transformación realizada.

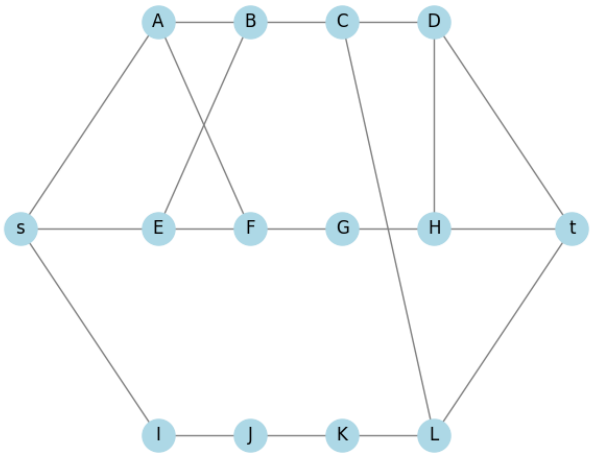

Figura 4: Grafo inicial, $N_4 = 7$

Mejora 1, aprovechamiento de una arista irrelevante: Partimos de un grafo $G \in T^5_{12,20}$ con una arista irrelevante. El vecindario *Aprovechar Aristas Irrelevantes* (ver Anexo 2) la reubica para conectar los vértices I (capa 1) y B (capa 2). El resultado es un nuevo grafo $G_1 \in T^5_{12,20}$ con un valor de N_4 mayor.

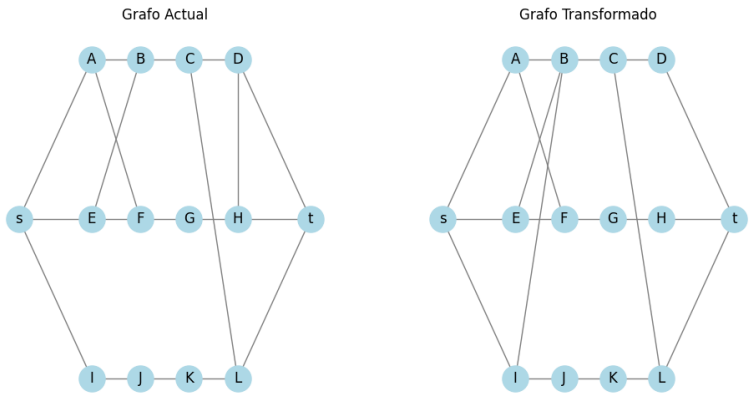

Figura 5: Vecindario Aprovechar Aristas Irrelevantes, $N_4 = 9$

Mejora 2, optimizar el uso de aristas entre capa 2 y capa 3: A partir de G_1, los primeros vecindarios no introducen cambios. El vecindario *Optimizar X* (ver Algoritmo 1) redistribuye las aristas entre las capas 2 y 3 para maximizar los productos $y_i z_j$, resultando en el grafo G_2 con una mejora en N_4.

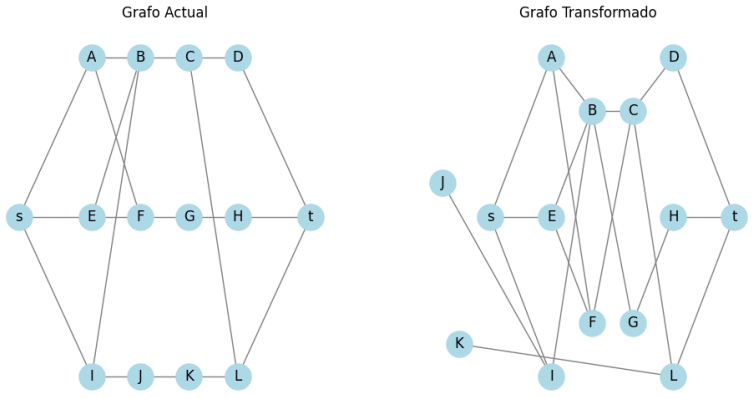

Figura 6: Vecindario Optimizar X, $N_4 = 13$

Mejora 3, aprovechamiento de una arista irrelevante: De forma análoga a la primera mejora, el vecindario *Aprovechar Aristas Irrelevantes* utiliza una

arista irrelevante para agregar la arista (I,F), generando el grafo G_3 y aumentando nuevamente el valor de N_4.

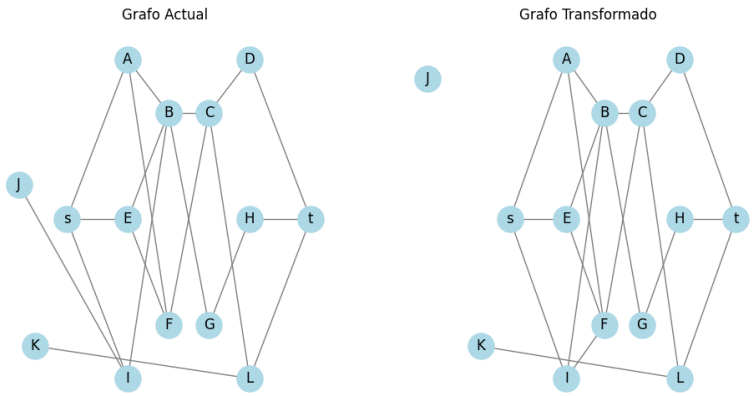

Figura 7: Vecindario Aprovechar Aristas Irrelevantes, $N_4 = 15$

Mejora 4, aprovechamiento de una arista irrelevante: El proceso se repite, aplicando una vez más *Aprovechar Aristas Irrelevantes* para obtener el grafo G_4 con un N_4 aún mayor.

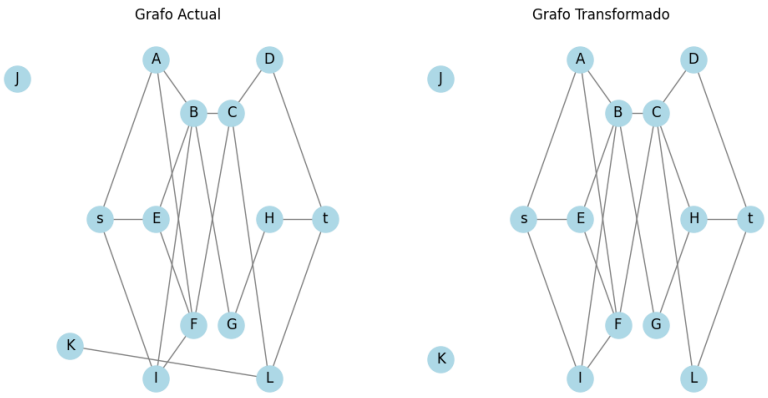

Figura 8: Vecindario Aprovechar Aristas Irrelevantes, $N_4 = 21$

Mejora 5, aumentar la cantidad de vértices en capa 1: Sobre el grafo G_4, los vecindarios anteriores no aplican. El vecindario *Aumentar Capa Uno* (ver Anexo 6) mueve un vértice de la capa 4 a la capa 1. Aunque esta transformación es posible, en este caso no mejora el valor de N_4, por lo que el algoritmo VNS finaliza y se rechaza el cambio.

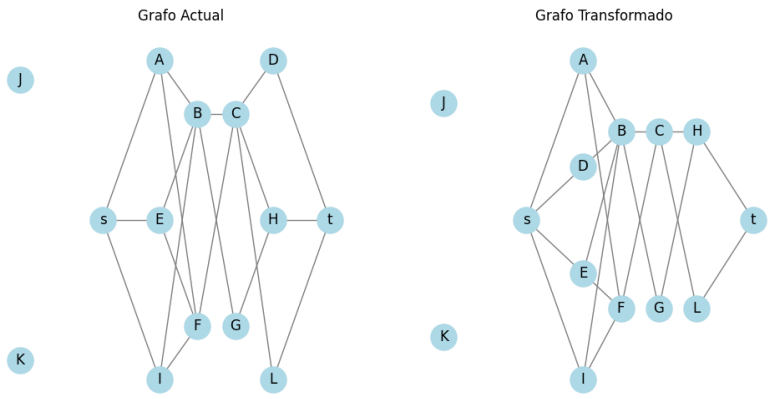

Figura 9: Vecindario Aumentar Capa Uno, $N_4 = 21$

Resultado: El valor de la función objetivo en el grafo original es $N_4(G) = 7$ mientras que el valor obtenido luego de aplicar VNS es $N_4(G_4) = 21$, pasadas 13 iteraciones.

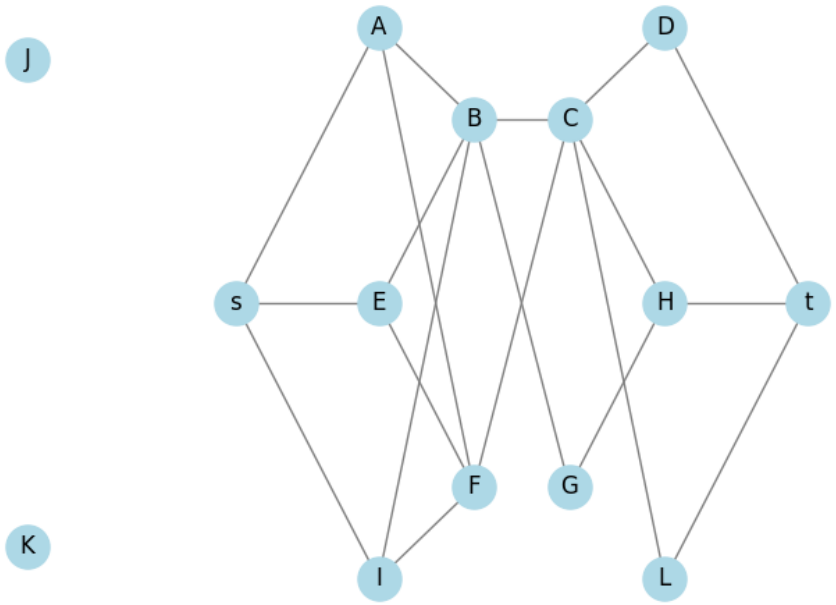

Figura 10: Resultado, $N_4 = 21$

5. Resultados computacionales

En esta sección nos dedicamos a comprobar la Conjetura 2 para familias finitas de grafos en $T^5_{n,m}$ mediante un enfoque computacional.

Los programas desarrollados en el transcurso de este proyecto se encuentran públicamente accesibles.[1]

Primero, en la sección 5.1, aplicamos el algoritmo de VNS sobre distintas familias de grafos en $T^5_{n,m}$, con el objetivo de encontrar para todo grafo $G \in \mathcal{M}^5_{n,m}$ un grafo $H \in T^5_{n,m}$ tal que $N_4(H) > N_4(G)$, así verificando las hipótesis de la Proposición 3.1 y por lo tanto comprobando que no existe grafo UMR para los valores de (n, m) explorados. Además, exploramos algunas familias parcialmente a través de un muestreo aleatorio de grafos, y no encontramos ningún contraejemplo para la Conjetura 2.

Luego, en la sección 5.2, calculamos el valor de N_4 de todos los grafos en tres

[1]Los programas desarrollados en el transcurso de este proyecto se encuentran públicamente accesibles en https://gitlab.fing.edu.uy/santiago.marquez.laguna/modulo-de-taller-2025-2s#.

familias $\mathcal{M}^5_{n,m}$ distintas, y encontramos para cada familia un grafo en $T^d_{n,m}$ tal que supera en N_4 a toda la familia $\mathcal{M}^5_{n,m}$, verificando entonces las hipótesis de la Proposición 3.1.

Por último, en la sección 5.3, justificamos la inclusión de cada vecindario en el algoritmo VNS implementado, observando una muestra de 1000 ejecuciones del algoritmo con grafos aleatoriamente generados en las clases $\mathcal{M}^5_{12,22}$ y $\mathcal{M}^5_{12,26}$, elegidas de forma arbitraria y con diferente cantidad de aristas, a modo de evidenciar cómo los resultados pueden variar de una clase a otra. Asimismo, discutimos la presencia de algunos vecindarios que pueden no parecer útiles en la práctica por la observación de los resultados, justificándola mediante ejemplos.

5.1. Aplicación de VNS a familias de grafos

En esta subsección se presentan los resultados obtenidos al aplicar el algoritmo de *Variable Neighborhood Search* (VNS) sobre distintas familias de grafos en $T^5_{n,m}$.

Para realizar esta exploración sobre las familias de grafos modificamos el algoritmo VNS con el fin de optimizar su tiempo de ejecución en cada familia de la siguiente forma: Se mantiene una referencia al mayor valor de N_4 encontrado a lo largo de la ejecución, y para cada grafo por explorar tal que su N_4 inicial es menor que el valor de N_4 guardado, se detiene la exploración para ese grafo y se avanza al siguiente, ya que se ha encontrado previamente un grafo con un valor mayor de N_4. En ciertos casos particulares se permite que avance la exploración sobre un grafo, para alcanzar valores mayores de N_4 los cuales permitirán ahorrar iteraciones a lo largo de la ejecución.

Se distinguen dos tipos de experimentos:

- Exploración exhaustiva de todas las combinaciones posibles para ciertos valores de (n, m).
- Exploración parcial (muestreo) para valores mayores de (n, m), debido a limitaciones computacionales.

5.1.1. Exploración exhaustiva para $n = 12$

Para $n = 12$ y valores de m entre 15 y 20, se generaron todas las combinaciones posibles de grafos en $\mathcal{M}^5_{n,m}$. En la columna *Mejora en VNS* indicamos si para cada

uno de los grafos generados se encontró un grafo que lo supere en N_4.

m	Grafos generados	Mejora en VNS
15	1	Sí
16	30	Sí
17	435	Sí
18	4060	Sí
19	27405	Sí
20	142506	Sí

En todos los casos analizados, el algoritmo VNS logró mejorar el valor objetivo N_4 para la totalidad de los grafos considerados. Por lo tanto, se proporciona evidencia computacional para la Conjetura 2 para $n = 12$ y m entre 15 y 20: en los casos recorridos se verifica la hipótesis de la Proposición 3.1, y en consecuencia no existe grafo UMR para dichos valores de (n, m).

5.1.2. Exploración parcial para $n = 12$

Para valores de $m \geq 21$, se realizó una exploración sobre subconjuntos de grafos de cada familia, manteniendo m por debajo de la cota dada por la Conjetura 2: $m \leq 3\lfloor \frac{12}{4} \rfloor^2 + 2\lfloor \frac{12}{4} \rfloor - 3 = 3 \cdot 9 + 2 \cdot 3 - 3 = 30$

m	Cantidad de grafos analizados	Mejora en VNS
21	55	Sí
22	50	Sí
23	46	Sí
24	42	Sí
25	39	Sí
26	36	Sí
27	33	Sí
28	31	Sí
29	29	Sí
30	27	Sí

5.1.3. Exploración parcial para $n \geq 13$

Se realizaron experimentos adicionales para $n = 13, 14, 15$ y valores de m entre 21 y 30, utilizando un número reducido de instancias por familia.

n	m	Cantidad de grafos analizados	Mejora en VNS
13	21→30	$27 \to 13$	Sí
14	21→30	$13 \to 6$	Sí
15	21→30	$6 \to 3$	Sí

En todos los casos considerados, el algoritmo VNS logró mejorar el valor objetivo en la totalidad de las instancias analizadas.

5.2. Búsqueda exhaustiva

En esta sección determinamos, en cada uno de los conjuntos $T^5_{n,m}$ tal que $(n,m) \in \{(18,22),(18,24),(12,22)\}$, el conjunto $\mathcal{M}^5_{n,m}$. Luego obtenemos un grafo H en $T^5_{n,m}$ que cumple que $N_4(H) > N_4(G)$ para cualquier elección de grafo G dentro de $\mathcal{M}^5_{n,m}$. Luego, por la Proposición 3.1 se sigue que no existe ningún grafo uniformemente más confiable en $T^5_{n,m}$ cuando $(n,m) \in \{(18,22),(18,24),(12,22)\}$, mostrando así una evidencia computacional de la validez de la Conjetura 2 para estos casos concretos.

5.2.1. Caso $T^5_{18,22}$

Para la familia $T^5_{18,22}$ donde $n = 18$ y $m = 22$, realizamos una búsqueda exhaustiva de grafos que contienen a $D^5_{k_5(18,22)}$ como subgrafo, donde $k_5(18,22) = \text{mín}\{\lfloor \frac{18}{4} \rfloor, \lfloor \frac{22}{5} \rfloor\} = \text{mín}\{4,4\} = 4$. Dichos grafos forman el conjunto $\mathcal{M}^5_{18,22}$ en su totalidad.

Calculamos $N_4(G)$ para cada $G \in \mathcal{M}^5_{18,22}$ y obtuvimos:

$$\max_{G \in \mathcal{M}^5_{18,22}} N_4(G) = 7$$

Sin embargo, por inspección, encontramos un grafo $H \in T^5_{18,22}$ que no pertenece a $\mathcal{M}^5_{18,22}$ tal que:

$$N_4(H) = 13 > 7$$

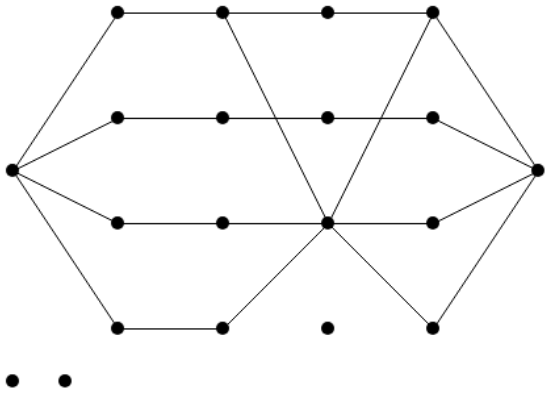

Figura 11: Grafo H

Este resultado proporciona evidencia computacional de que no existe grafo UMR en $T^5_{18,22}$: se verifica la hipótesis de la Proposición 3.1, ya que todo grafo de $\mathcal{M}^5_{18,22}$ queda estrictamente superado en N_4 por un grafo de $T^5_{18,22}$.

5.2.2. Casos $T^5_{18,24}$ y $T^5_{12,22}$

Bajo un procedimiento análogo al caso anterior, obtenemos los siguientes resultados:

$$\max_{G \in T^5_{18,24}} N_4(G) = 13$$

$$\max_{G \in T^5_{12,22}} N_4(G) = 23$$

Aplicando nuestro algoritmo de VNS, encontramos los siguientes grafos $H_1 \in T^5_{18,24}$ y $H_2 \in T^5_{12,22}$ con $N_4(H_1) = 36 > 13$ y $N_4(H_2) = 27 > 23$.

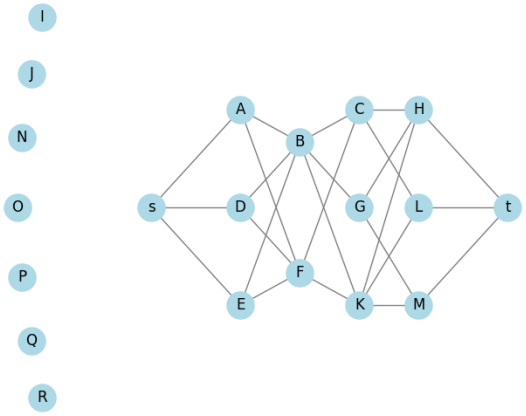

Figura 12: Grafo H_1

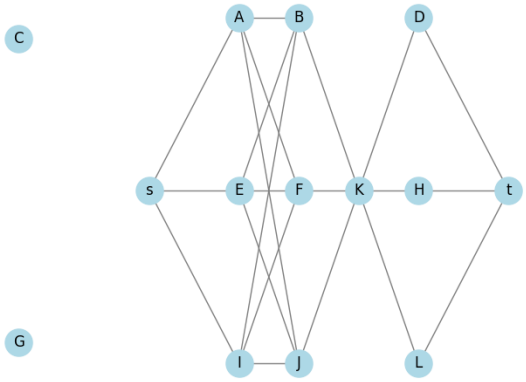

Figura 13: Grafo H_2

Por lo tanto, y aplicando nuevamente la Proposición 3.1, queda confirmada la Conjetura 2 para $(n, m) = (18, 24)$ y $(n, m) = (12, 22)$.

5.3. Actividad de Vecindarios

En esta sección nos dedicamos a enseñar la actividad de los vecindarios definidos para el algoritmo implementado, a modo de justificación para su presencia. Para llevar a cabo esta tarea, mostramos a través de tablas la actividad de los distintos vecindarios para 1000 grafos de la clase $T^5_{12,20}$ y 1000 grafos de la clase $T^5_{12,26}$. Cabe observar que esto no es un criterio definitivo para juzgar si las heurísticas desarrolladas deben o no deben estar presentes en el algoritmo, dado que algunas de ellas aplican en situaciones muy particulares donde la cardinalidad de vértices y aristas es considerablemente grande, lo cual resulta ser un limitante en esta investigación debido a la capacidad de cómputo. Asimismo, solo observamos una cantidad limitada de casos por lo que el objetivo de esta sección solo es dar una intuición de la actividad y no dar pruebas certeras.

En las tablas la columna de *recorridos* indica la cantidad de veces que el vecindario fue visitado a lo largo de las ejecuciones del algoritmo; *cambios* indica cuántas veces ocurrieron modificaciones sobre el grafo actual y *mejora* cuántas veces esos cambios fueron aceptados debido a que introdujeron una mejora en la función objetivo —esto es, favorecieron a maximizar N_4—; *Mej/Rec* es una proporción de cuántas veces hubo una mejora frente a cuántas veces se visitó el vecindario; *Mej/Cam* es una proporción de cuántas veces hubo mejora respecto de las que hubo una modificación en el grafo, es decir cuántas veces el cambio fue tomado porque introdujo una mejora en la función objetivo.

Observando las tablas 2 y 3 podemos apreciar que varían los vecindarios cuya relación de mejora/rechazo es la mayor. Aun así, esto nos permite observar que hay vecindarios visitados muy frecuentemente —como el caso de Aprovechar Aristas Irrelevantes y Ajustar Capas Externas— y otros que lo son menos, pero esto se debe en parte al orden de ejecución de los vecindarios.

Cuadro 2: Efectividad por vecindario $T^5_{12,20}$

Vecindario	**Recorridos**	**Cambios**	**Mejoras**	**Mej/Rec**	**Mej/Cam**
aumentarCapaUno	3630	3613	2634	**0.7256**	0.7290
aprovecharAristasIrrelevantes	11795	3759	3759	**0.3187**	**1.00**
ajustarCapasExternas	8035	2568	2550	0.3174	0.9930
optimizarX	4683	1048	1048	0.2238	**1.00**
ajustarEntradasSalidas	5485	866	800	0.1459	0.9238
minMax	3635	5	5	∼0.00	1.00

Cuadro 3: Efectividad por vecindario $T^5_{12,26}$

Vecindario	**Recorridos**	**Cambios**	**Mejoras**	**Mej/Rec**	**Mej/Cam**
aumentarCapaUno	12227	12004	11245	**0.9197**	0.9368
ajustarCapasExternas	24798	11249	11245	**0.4535**	∼1.00
aprovecharAristasIrrelevantes	30216	5414	5414	0.1792	1.00
optimizarX	13324	1094	1094	0.0821	1.00
ajustarEntradasSalidas	13551	232	227	0.0168	0.9784
minMax	12229	0	0	**0.00**	-

El caso de MinMax es uno muy particular. Se trata de un vecindario que en los escenarios estudiados no suele introducir cambios. Aún así, de acuerdo a la intuición exhibida en la sección anterior, es razonable pensar que es un vecindario que podría introducir mejoras en casos particulares y esto puede verse con la situación que lo ejemplifica en la figura 14. Se procede según el algoritmo 5 detallado en los Anexos.

El vecindario procede dados los vectores y y z, y la matriz de adyacencia X iniciales:

$$y = (1, 1, 1), \quad z = (1, 1, 1)$$

$$\min_{j \in \{1,\dots,|V_3|\}} z_j = \max_{j \in \{1,\dots,|V_3|\}} z_j = 1$$

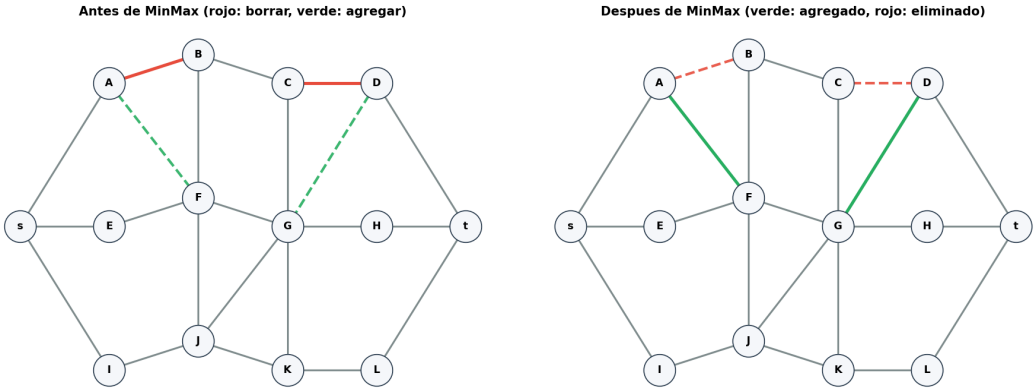

Figura 14: Ejemplo de mejora para vecindario MinMax

$$X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$N_4 = (1,1,1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = (1,2,1) \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 4$$

Nuevo estado:

$$y = (0,2,1), \quad z = (0,2,1)$$

$$X = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$N_4 = (0,2,1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} = (0,3,1) \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} = 7$$

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a Pablo Romero por brindarnos la oportunidad de colaborar en este proyecto. Su confianza, entusiasmo y dedicación desde el principio fueron fundamentales para que pudiéramos llevar a cabo esta iniciativa, logrando incentivarnos a explorar y descubrir una gran variedad de áreas de interés.

Asimismo, nos gustaría extender nuestro agradecimiento a Felipe Miranda por su constante acompañamiento durante el módulo de taller. Su apoyo, aportes y disposición marcaron una gran diferencia a lo largo de todo el proceso.

A ambos, les agradecemos por su presencia y por las valiosas contribuciones que hicieron, no solo en el ámbito académico, sino también en el personal.

Referencias

[1] Pierre Hansen y Nenad Mladenović. "Variable neighborhood search: Principles and applications". En: *European Journal of Operational Research* 130.3 (2001), págs. 449-467. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(00)00100-4. URL: https://www.sciencedirect.com/science/article/pii/S0377221700001004.

[2] Reinhard Diestel. *Graph Theory*. 5.ª ed. Vol. 173. Graduate Texts in Mathematics. Springer, 2017.

[3] Isaac Brown et al. "Most reliable two-terminal graphs with node failures". En: *Networks* 76.3 (2020), págs. 414-426. DOI: https://doi.org/10.1002/net.21968. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/net.21968. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21968.

[4] Pablo Romero. "Uniformly optimally reliable graphs: A survey". En: *Networks* 80.4 (2022), págs. 466-481.

[5] Felipe Miranda. "Grafos con dos terminales uniformemente más confiables bajo el modelo de falla de vértices : La conjetura de Brown es cierta cuando d=4". Tesis de grado. Universidad de la República (Uruguay). Facultad de Ingeniería, 2025.

[6] Pablo Romero. "Characterization of locally most split reliable graphs". En: *Theoretical Computer Science* 1047 (2025), pág. 115327. ISSN: 0304-3975.

Anexo A

A. Anexo 1

A.1. Algoritmos de Vecindarios

Algoritmo 2: Aprovechar Aristas Irrelevantes

```
1  Calcular vectores y, z y matriz X;
2  A_irr ← obtener aristas irrelevantes;
3  if A_irr = ∅ then
4  |  return sin cambios;
5  end
   // Saturar vector y (conexiones V_1 ↔ V_2)
6  for i ← 1 to |V_2| do
7  |  if y_i < |V_1| then
8  |  |  Buscar u ∈ V_1 tal que (u, V_2[i]) ∉ E(G);
9  |  |  if existe u then
10 |  |  |  Eliminar arista A_irr[0];
11 |  |  |  Agregar arista (u, V_2[i]);
12 |  |  |  return grafo modificado;
13 |  |  end
14 |  end
15 end
   // Saturar vector z (conexiones V_3 ↔ V_4)
16 for j ← 1 to |V_3| do
17 |  if z_j < |V_4| then
18 |  |  Buscar w ∈ V_4 tal que (V_3[j], w) ∉ E(G);
19 |  |  if existe tal w then
20 |  |  |  Eliminar arista A_irr[0];
21 |  |  |  Agregar arista (V_3[j], w);
22 |  |  |  return grafo modificado;
23 |  |  end
24 |  end
25 end
   // Saturar matriz X (conexiones V_2 ↔ V_3)
26 for i ∈ |V_2|, j ∈ |V_3| do
27 |  if X[i][j] = 0 then
28 |  |  Eliminar arista A_irr[0];
29 |  |  Agregar arista (V_2[i], V_3[j]);
30 |  |  return grafo modificado;
31 |  end
32 end
   // Caso particular: aprovechar vértices irrelevantes
33 if existen vértices irrelevantes and |A_irr| ≥ 2 then
34 |  x ← primer vértice irrelevante;
35 |  Eliminar aristas A_irr[0] y A_irr[1];
36 |  Agregar aristas (s, x) y (x, V_2[0]);
37 |  return grafo modificado;
38 end
39 return sin cambios;
```

Algoritmo 3: Ajuste de capas externas cuando $n_4 > n_1 + 1$

Entrada: Grafo G con dos terminales s y t

Salida : Dupla (G', cambio) donde G' es una **copia** de G si se modifica y $\mathit{cambio} \in \{\texttt{True}, \texttt{False}\}$

$\mathit{cambio} \leftarrow \texttt{False}$;
$x \leftarrow$ **último** vértice de V_4 (según el orden fijo de la capa);
$m' \leftarrow gr_G(x)$;

`// Paso 1: mover` x `de capa 4 a capa 1 y ajustar aristas`
$V_4 \leftarrow V_4 \setminus \{x\}$;
$V_1 \leftarrow V_1 \cup \{x\}$;
`// (1.a) Reemplazar la arista con` t `por una con` s
$E(G') \leftarrow E(G') \setminus \{(x,\ t)\}$;
$E(G') \leftarrow E(G') \cup \{(x,\ s)\}$;
`// (1.b) Eliminar una arista hacia capa 3 y agregar una hacia el último vértice de capa 2`
`// Sea` $N_{G'}(x)$ `el conjunto de vecinos de` x `en el grafo` G'
$v \leftarrow$ **algún** vértice en $N_{G'}(x) \cap V_3$ (por ejemplo el último);
$E(G') \leftarrow E(G') \setminus \{(x,\ v)\}$;
$v_{\text{last}} \leftarrow$ **último** vértice de V_2;
$E(G') \leftarrow E(G') \cup \{(x,\ v_{\text{last}})\}$;

`// Paso 2: redistribuir el resto de aristas de` x `sin romper la estructura de capas`
RedistribuirAristas(G')

$\mathit{cambio} \leftarrow \texttt{True}$;
return (G', cambio);

Algoritmo 4: Ajustar Entradas-Salidas

```
1  Calcular vectores y, z y matriz X;
   // Buscar un nodo de V_2 que no tenga dupla justa
2  for i ← 1 to |V_2| do
3      tiene_dupla_justa ← falso;
4      for j ← 1 to |V_3| do
5          if X[i][j] = 1 y |y_i − z_j| ≤ 1 then
6              tiene_dupla_justa ← verdadero;
7              break;
8          end
9      end
10     if no tiene_dupla_justa then
           // Encontrar el j más cercano a y_i
11         j* ← arg mín_j |y_i − z_j|;
12         if y_i < z_{j*} then
               // Aumentar y_i y disminuir z_{j*}
13             Buscar u ∈ V_1 tal que (u, V_2[i]) ∉ E(G);
14             Buscar w ∈ V_4 tal que (V_3[j*], w) ∈ E(G);
15             if existen u y w then
16                 Eliminar arista (V_3[j*], w);
17                 Agregar arista (u, V_2[i]);
18                 return grafo modificado;
19             end
20         else
               // Disminuir y_i y aumentar z_{j*}
21             Buscar w ∈ V_4 tal que (w, V_3[j*]) ∉ E(G);
22             Buscar u ∈ V_1 tal que (u, V_2[i]) ∈ E(G);
23             if existen u y w then
24                 Eliminar arista (u, V_2[i]);
25                 Agregar arista (w, V_3[j*]);
26                 return grafo modificado;
27             end
28         end
29     end
30 end
31 return sin cambios;
```

Algoritmo 5: Min-Max

```
1  Calcular vectores y, z y matriz X;
2  if |V_3| = 1 or mín(z) < máx(z) then
3  |  return sin cambios;
4  end
   // Intentar romper los empates en z
5  for j ← 1 to |V_3| do
6  |  I_j ← {i : X[i][j] = 1 ∧ |y_i − z_j| ≤ 1} ;          // Índices justos
7  |  if I_j = ∅ then
   |  |  // Buscar k que pueda acoplarse con j
8  |  |  Buscar k ∈ |V_2| tal que (y_k − máx(z)) ∈ {0, −1} y X[k][j] = 0;
9  |  |  if existe tal k then
10 |  |  |  Buscar w ∈ V_4 tal que (V_3[j], w) ∈ E(G);
11 |  |  |  Buscar u ∈ V_1 tal que (u, V_2[k]) ∉ E(G);
12 |  |  |  if existen w y u then
13 |  |  |  |  Eliminar arista (V_3[j], w);
14 |  |  |  |  Agregar arista (u, V_2[k]);
15 |  |  |  |  return grafo modificado;
16 |  |  |  end
17 |  |  else
   |  |  |  // Mover arista de j a otro l ≠ j
18 |  |  |  Tomar l ∈ |V_3| con l ≠ j;
19 |  |  |  Buscar w, w' ∈ V_4 con (V_3[j], w) ∈ E(G) y (V_3[l], w') ∉ E(G);
20 |  |  |  if existen w y w' then
21 |  |  |  |  Eliminar arista (V_3[j], w);
22 |  |  |  |  Agregar arista (V_3[l], w');
23 |  |  |  |  return grafo modificado;
24 |  |  |  end
25 |  |  end
26 |  end
27 |  else
   |  |  // Verificar si algún i ∈ I_j tiene a j como único justo
28 |  |  for i ∈ I_j do
29 |  |  |  es_único ← verdadero;
30 |  |  |  for j' ∈ |V_3| \ {j} do
31 |  |  |  |  if X[i][j'] = 1 y |y_i − z_j'| ≤ 1 then
32 |  |  |  |  |  es_único ← falso;
33 |  |  |  |  |  break;
34 |  |  |  |  end
35 |  |  |  end
36 |  |  |  if es_único then
   |  |  |  |  // Disminuir y_i y z_j, aumentar otro par (k, l)
37 |  |  |  |  Buscar par (k, l) con k ≠ i, l ≠ j, X[k][l] = 1, |y_k − z_l| ≤ 1;
38 |  |  |  |  if existe (k, l) then
39 |  |  |  |  |  Buscar u, u' ∈ V_1 y w, w' ∈ V_4 apropiados;
40 |  |  |  |  |  Eliminar aristas (u, V_2[i]) y (V_3[j], w);
41 |  |  |  |  |  Agregar aristas (u', V_2[k]) y (w', V_3[l]);
42 |  |  |  |  |  return grafo modificado;
43 |  |  |  |  end
44 |  |  |  end
45 |  |  end
46 |  end
47 end
```

Algoritmo 6: Aumentar Capa Uno

Calcular capas V_1, V_2, V_3, V_4;
$k_5(n,m) \leftarrow \text{mín}\{n/4, m/5\}$;
if $|V_1| \geq k_5(n,m) + 1$ ***y*** $|A_{irr}| \leq k_5(n,m)$ **then**
    return sin cambios;
end
`// Intentar usar nodos de capa 4`
if $|V_4| > 1$ **then**
    $x \leftarrow \arg\text{mín}_{v \in V_4}\, gr(v)$; `// Nodo menos acoplado`
    $m' \leftarrow gr(x)$;
    Eliminar todas las aristas incidentes a x;
    Recalcular capas temporalmente;
    if $1 + |V_1'| + |V_2'| + |V_{irr}'| \geq m'$ **then**
        `// Hay espacio para reubicar las aristas de` x
        Agregar arista (s, x);
        Conectar x con nodos de V_2' hasta agotar aristas;
        Si sobran aristas, conectar con V_1';
        Si sobran aristas, conectar con vértices irrelevantes;
        return grafo modificado;
    end
end
`// Intentar usar nodos de capa 3`
if $|V_3| > 1$ **then**
    $x \leftarrow \arg\text{mín}_{v \in V_3}\, gr_{der}(v)$; `// Mínimo grado por derecha`
    Realizar proceso similar al de capa 4;
    if *éxito* **then**
        return grafo modificado;
    end
end
`// Intentar usar nodos de capa 2`
if $|V_2| > 1$ **then**
    $x \leftarrow \arg\text{mín}_{v \in V_2}\, gr_{der}(v)$; `// Menor grado por derecha`
    Realizar proceso similar al de capa 3;
    if *éxito* **then**
        return grafo modificado;
    end
end
`// Intentar usar nodos irrelevantes`
if *Es posible usar nodo irrelevante* **then**
    return grafo modificado;
end
return sin cambios;

Algoritmo 7: RedistribuirAristas

```
// Primero saturar conexiones de x con capa 2 en orden
for v ∈ V_2 en orden desde el último al primero do
    if (x, v) ∉ E(G') and existe arista (x,u) ∈ E(G') con u ∈ V_3 then
        E(G') ← E(G') \ {(x,u)};
        E(G') ← E(G') ∪ {(x, v)};
    end
end
if aún quedan aristas de x por reubicar and existe u ∈ V_1 tal que u no
  está conectado a todo V_2 then
    while aún quedan aristas de x por reubicar and existe u ∈ V_1 tal que
      u no está conectado a todo V_2 do
        u ← último vértice de V_1 tal que |N_{G'}(u) ∩ V_2| < |V_2|;
        v ← último vértice de V_2 tal que uv ∉ E(G');
        E(G') ← E(G') ∪ {(u,v)};
    end
else
    return;
end
if aún quedan aristas de x por reubicar and existe par {u,u'} ⊆ V_1 con
  (u,u') ∉ E(G') then
    while aún quedan aristas de x por reubicar and existe par
      {u,u'} ⊆ V_1 con (u,u') ∉ E(G') do
        Tomar {u,u'} ⊆ V_1 (por ejemplo, los dos últimos distintos) con
          (u,u') ∉ E(G');
        E(G') ← E(G') ∪ {(u,u')};
    end
else
    return;
end
if aún quedan aristas de x por reubicar and existe par {w,w'} ⊆ V_4 con
  (w,w') ∉ E(G') then
    while aún quedan aristas de x por reubicar and existe par
      {w,w'} ⊆ V_4 con (w,w') ∉ E(G') do
        Tomar {w,w'} ⊆ V_4 (por ejemplo, los dos últimos distintos) con
          ww' ∉ E(G');
        E(G') ← E(G') ∪ {(w,w')};
    end
else
    return;
end
// Si todavía sobran aristas, se sale de las hipótesis
if aún quedan aristas de x por reubicar then
    error("El grafo se sale de las hipótesis: el vértice movido tenía m'
      aristas y  m' > n_2 + 1 + C(n_1,2) + C(n_4,2).");
end
```

Anexo B

B. Anexo 1

Repositorio de GitLab: https://gitlab.fing.edu.uy/santiago.marquez.laguna/modulo-de-taller-2025-2s#