DAA: Proyecto Final

Abstract

El presente informe se confecciona con el objetivo de hacer un análisis detallado de la problemática en cuestión, utilizando como cimiento los contenidos recibidos a lo largo del curso para proveer una solución lo suficientemente buena para lidiar con la complejidad de la misma.

1. El problema

Conectando la UH

La Universidad de La Habana (UH), en su constante búsqueda de la excelencia académica y la innovación, se ha embarcado en un proyecto crucial para modernizar y expandir su infraestructura de red. Nuestro objetivo es dotar a todas nuestras facultades, centros de investigación y edificios administrativos con conectividad de fibra óptica de alta velocidad. Para este fin, contamos con el valioso apoyo técnico y logístico de ETECSA (Empresa de Telecomunicaciones de Cuba S.A.).

Nos enfrentamos a un desafío de diseño de red que requiere una solución óptima. Necesitamos interconectar todos los edificios principales de la UH con fibra óptica, creando una red robusta y eficiente. Cada posible conexión de fibra entre dos edificios tiene un costo de instalación asociado, que incluye desde los permisos internos y la mano de obra especializada de ETECSA hasta los materiales y las obras civiles necesarias.

Sin embargo, ETECSA ha establecido una restricción técnica fundamental que debemos respetar:

En cada edificio, la conexión de la fibra óptica se gestionará a través de un equipo de red central (un router o switch principal) que ellos nos proporcionan. Estos equipos tienen una capacidad limitada de puertos. Esto significa que un equipo en un edificio específico solo puede manejar un número máximo de conexiones de fibra óptica directas a otros edificios. Exceder este límite implicaría la necesidad de instalar equipos adicionales mucho más caros y complejos, o la implementación de soluciones de red alternativas que ETECSA no puede garantizar o que dispararían drásticamente el presupuesto del proyecto.

Nuestro objetivo principal es diseñar la red de fibra óptica que conecte todos nuestros edificios principales de la manera más económica posible. Esto implica seleccionar las rutas de fibra de tal forma que:

1. Todos los edificios estén interconectados a la red principal de la universidad, sin crear bucles innecesarios (buscamos una estructura de red eficiente).

2. Ningún equipo de red en ningún edificio exceda su capacidad máxima de conexiones directas (es decir, el número de cables de fibra que llegan o salen de un edificio no puede superar el límite de puertos del equipo de ETECSA).

3. El costo total de instalación de toda la red sea el mínimo posible.

Una planificación subóptima podría resultar en un sobrecosto significativo para la universidad, la necesidad de adquirir hardware de red adicional no previsto, o en una red ineficiente que no cumpla con las especificaciones técnicas y presupuestarias acordadas con ETECSA.

1.1. Comentarios

El problema en cuestión es clásico en el diseño y montaje de redes físicas, siendo un caso digno de estudio por más de cuarenta años para los matemáticos, con enfoque especial en ramas como la optimización y la teoría de grafos.

2. Modelación del problema

Sea un grafo no dirigido y conexo $G = \langle V, E \rangle$ donde:

- Cada vértice $v \in V$ representa un edificio principal de la Universidad de la Habana
- Cada arista $e = \langle u, v \rangle \in E$ representa la posibilidad de instalar un enlace de fibra óptica directo entre los edificios u y v.
- A cada arista $e \in E$ se le asocia un peso $w(e) > 0$, que representa el costo de instalación del enlace correspondiente.

Adicionalmente a cada vértice $v \in V$ se le asocia un entero positivo $d(v)$ que representa el número máximo de conexiones de fibra óptica (grado máximo) que puede soportar el equipo de red instalado en dicho edificio.

2.1. Estructura de red deseada

Para garantizar conectividad total sin bucles innecesarios se busca una subestructura G que:

1. Conecte todos los vértices de V.
2. Sea acíclica.

Estas propiedades caracterizan a un *árbol abarcador* del grafo G.

2.2. Formalización como Árbol Abarcador de Costo Mínimo con Restricción de Grado

Definición 2.1 (Árbol Abarcador de Costo Mínimo con Restricción de Grados)**.** Dado un grafo no dirigido y conexo $G = \langle V, E \rangle$ con una función de costo para las aristas $w : E \to \mathbb{R}^+$ y una función de cotas de grado $d : V \to \mathbb{N}$, el problema del Árbol Abarcador de Costo Mínimo con Restricción de Grado consiste en encontrar un árbol abarcador $T = \langle V, E_T \rangle$ tal que:

1. $\deg_T(v) \leq d(v), \forall v \in V$

2. La suma de los pesos $\sum_{e \in E_T} w(e)$ sea mínima.

En lo adelante todas las referencias al nombre de este problema se harán por su nombre en inglés: *Degree Constrained Minimum Spanning Tree* (DCMST), además, se trabajará con $n = |V|$ y $m = |E|$.

2.3. Propiedades y Observaciones Iniciales

- Si $d(v) = n - 1$ para todo $v \in V$, el problema se reduce al clásico Árbol Abarcador de Costo Mínimo (Minimum Spanning Tree en inglés o MST), resoluble en tiempo polinomial mediante algoritmos como Kruskal o Prim.
- La introducción de restricciones de grado rompe las propiedades de optimalidad local que permiten dichos algoritmos voraces.
- En contextos reales, las cotas de grados suelen ser pequeñas, lo incrementa la dificultad computacional del problema.

3. Complejidad Computacional

Para analizar la complejidad computacional del problema, consideramos su versión de decisión.

Definición 3.1 (DCMST-DEC)**.** Dado un grafo no dirigido y conexo $G = \langle V, E \rangle$, una función de pesos $w : E \to \mathbb{R}^+$, una función de cotas de grado $d : V \to \mathbb{N}$ y un entero k, ¿existe un árbol abarcador T de G tal que:

1. $\deg_T(v) \leq d(v)$, $\forall v \in V$ y
2. $\sum_{e \in E_T} w(e) \leq k$?

Lema 3.1. DCMST-DEC pertenece a la clase NP.

Demostración. Dado un certificado consistente en un conjunto de aristas $E_T \subseteq E$, es posible verificar en tiempo polinomial que $T = \langle V, E_T \rangle$ es un árbol abarcador que satisface las restricciones de grado para cada vértice y que el costo total no excede k. Por tanto, DCMST-DEC pertenece a NP. □

Lema 3.2. DCMST-DEC es NP-Hard

Demostración. Hagamos una reducción del problema *Camino Hamiltoniano (Hamiltonian Path)* en grafos no dirigidos, el cual es NP-completo y probemos que Hamiltonian Path $\leq_P$ DCMST-DEC.

Sea $G_H = \langle V_H, E_H \rangle$ una instancia del problema del Camino Hamiltoniano. Construimos una instancia $G = \langle V, E \rangle$ de DCMST-DEC de la siguiente manera:

1. $V = V_H$.
2. $E = E_H$.
3. Para toda arista $e \in E$, definamos $w(e) = 1$.

4. Para todo vértice $v \in V$, definamos $d(v) = 2$.

5. Definamos $k = n - 1$.

Observemos que cualquier árbol abarcador de G tiene exactamente $n-1$ aristas y por la restricción de costo todas las aristas del árbol deben tener peso 1, por lo que el costo de cualquier árbol es $n - 1$.

Supongamos que G_H admite un camino Hamiltoniano. Dicho camino es un árbol abarcador de G en el cual cada vértice tiene grado a lo sumo 2. Por tanto, cumplen todas las restricciones de la instancia construida de DCMST-DEC, y su costo total es exactamente $n - 1 \leq k$.

Recíprocamente, supongamos que existe un árbol abarcador T de G que satisface las restricciones de grado y costo. Dado que todos los pesos son iguales a 1 y el costo total es $n - 1$, T debe contener exactamente $n - 1$ aristas. Además, la restricción $\deg_T(v) \leq 2$ para todo v implica que T es un camino que visita todos los vértices exactamente una vez. Por lo tanto, T corresponde a un camino Hamiltoniano en G_H.

La transformación descrita es realizable en tiempo polinomial y correcta en ambos sentidos. En consecuencia, DCMST-DEC es NP-Hard. □

Teorema 3.1. DCMST-DEC es NP-Completo

Demostración. En orden para probar esta afirmación debemos mostrar que:

1. DCMST-DEC pertenece a la clase NP (Demostrado en Lema 1).

2. DCMST-DEC es NP-Hard (Demostrado en Lema 2).

□

4. Diseño de Soluciones Algorítmicas

El problema del Degree-Constrained Minimum Spanning Tree (DC-MST) combina la necesidad de construir un árbol abarcador de costo mínimo con restricciones adicionales sobre el grado máximo permitido en cada vértice. Esta combinación da lugar a un espacio de soluciones combinatoriamente grande, lo que convierte al problema en computacionalmente intratable para instancias de tamaño moderado o grande.

Desde un punto de vista algorítmico, la dificultad del problema no radica únicamente en garantizar la conectividad, sino en hacerlo respetando restricciones locales que impiden aplicar directamente algoritmos clásicos de árboles abarcadores de costo mínimos. Esto ha motivado el desarrollo de enfoques tanto exactos como heurísticos en la literatura.

4.1. Enfoques en la literatura

En la literatura se han propuesto numerosos enfoques para abordar el DC-MST. Entre ellos, pueden destacarse dos líneas principales que resultan complementarias:

Heurísticas iniciales

En [1] se propone la construcción de soluciones factibles iniciales mediante algoritmos voraces y procedimientos de mejora local basados en intercambios de aristas, denominados como *primal method* y *dual method* (este último será discutido más adelante). Estas heurísticas no garantizan optimalidad, pero permiten obtener rápidamente árboles que

cumplen las restricciones de grado y cuyo costo sirve como cota superior del problema. Dichas soluciones son fundamentales tanto para evaluar la calidad de otros métodos como para acelerar algoritmos exactos posteriores.

En [2] se desarrolla un procedimiento heurístico reforzado por un análisis estructural del problema basado en intercambios de aristas. Este análisis permite identificar:

- Aristas indispensables, que deben pertenecer a toda solución óptima.
- Aristas superfluas, que no pueden aparecer en ninguna solución óptima.

La identificación temprana de este tipo de aristas reduce de manera significativa el espacio de búsqueda para la aplicación posterior de métodos exactos, haciendo posible resolver instancias de tamaño moderado de forma exacta.

Ambos enfoques comparten una idea central: una buena solución heurística inicial, junto con un análisis inteligente de las aristas permiten manejar un problema que de otro modo resultaría intratable.

4.2. Motivación para un enfoque por etapas

Dado que los algoritmos exactos solo son viables para instancias pequeñas o medianas, resulta natural adoptar un enfoque progresivo. En primer lugar, se define un algoritmo exacto de referencia que garantice la obtención de la solución óptima en instancias pequeñas. Posteriormente, se desarrollan soluciones heurísticas y aproximadas más eficientes, cuya calidad puede evaluarse en relación con dicho óptimo.

4.3. Un primer acercamiento: fuerza bruta

Como punto de partida, se considera un algoritmo de fuerza bruta que explore exhaustivamente el espacio de soluciones posibles. Este enfoque tiene como objetivo principal servir como línea base para la evaluación de métodos heurísticos, más que como una solución práctica para instancias grandes.

Sea $G = \langle V, E \rangle$ el grafo de entrada, donde a cada vértice $v \in V$ se le asocia una cota de grado $d(v)$, y a cada arista $e \in E$ un costo $w(e)$. El algoritmo de fuerza bruta se define como sigue:

1. Enumerar todas las combinaciones de aristas que podrían constituir un árbol abarcador.
2. Para cada subconjunto candidato $E_T \subseteq E$:
   - Verificar que $T = \langle V, E_T \rangle$ sea un árbol (conexo y acíclico).
   - Verificar que $\deg_T(v) \leq d(v), \forall v \in V$
3. Calcular el costo total del árbol como $w(T) = \sum_{e \in E_T} w(e)$.
4. Seleccionar el árbol factible de menor costo.

Este procedimiento es correcto debido a que explora exhaustivamente el conjunto de soluciones factibles, garantizando la obtención de la solución óptima. Sin embargo, su complejidad es exponencial en el número de aristas, pues en el peor caso se examinan $2^m - 1$ subconjuntos. En consecuencia, este enfoque solo es aplicable a instancias pequeñas, pero resulta esencial como referencia para validar y comparar soluciones no exactas.

4.4. Enfoques heurísticos

Como se ha discutido en la sección anterior, el enfoque de fuerza bruta permite encontrar la solución óptima del problema del Árbol Abarcador de Costo Mínimo con Restricción de Grado (DC-MST), pero su complejidad exponencial lo limita a instancias de tamaño reducido. En escenarios realistas, donde el número de vértices y aristas crece de forma significativa, resulta imprescindible recurrir a estrategias heurísticas que permitan obtener soluciones factibles en tiempos computacionalmente aceptables.

Las heurísticas no garantizan optimalidad, pero constituyen una herramienta fundamental en la práctica. En particular, cumplen dos funciones esenciales:

1. generar soluciones directamente utilizables en instancias grandes del problema, y
2. proporcionar cotas superiores de buena calidad que pueden ser empleadas posteriormente en esquemas exactos de tipo branch and bound.

En esta sección se presentan dos heurísticas constructivas clásicas de la literatura, conocidas como CH (inspirada en Christofides para el problema del viajante) y AH, (basada en el análisis estructural de las aristas del árbol). Ambas presentan la necesidad de comenzar con una cota inferior (lower bound o lb), proporcionada preferiblemente por un MST y una cota superior (upper bound o ub) y han demostrado ser especialmente efectivas para el DC-MST.

4.4.1. Punto de partida común: el MST clásico y una solución factible

Un árbol abarcador de costo mínimo (MST) proporciona:

- una cota inferior del costo óptimo del DC-MST en tiempo polinomial $O(m \log n)$,
- una estructura inicial con buena calidad global,
- una base sobre la cual aplicar transformaciones locales.

Sin embargo, el MST puede violar severamente las restricciones de grado, por lo que debe ser modificado antes de ser considerado una solución válida.

Un árbol factible inicial proporciona:

- una cota superior del costo óptimo del DC-MST
- una estructura inicial con la que comenzar la búsqueda
- una base sobre la que desplazarnos en busca de mejores soluciones

La obtención de dicho árbol factible se hará con el método dual propuesto por Narula y Ho [1] basado en las siguientes observaciones:

- muchas aristas del MST también formarán parte del DC-MST y probablemente muchas de estas sean aquellas que inciden en un vértice i que no viola su restricción de grado.
- podemos manipular las aristas del MST para obtener un árbol abarcador con restricción de grados (DCST en lo adelante).

- si se considera un vértice i tal que $\deg_T(i) > d(i)$, por cada arista $e_{ij} = \langle i, j \rangle \in T$, si se encuentra una arista de reemplazo admisible $e_{rs} = \langle r, s \rangle$ tal que:

  1. la eliminación de e_{ij} y la inclusión de e_{rs} resulte en un árbol,
  2. la restricción de grados de los vértices r y s no se vea violada,
  3. la penalización $p = w(e_{rs}) - w(e_{ij})$ sea mínima entre todas las aristas e_{rs}.

  Entonces haremos un intercambio entre el par (e_{ij}, e_{rs}) para el cual la penalización sea mínima. Los intercambios se harán hasta que $\deg_T(i) \leq d(i)$, $\forall i \in T$.

En base a estas observaciones el procedimiento de intercambio se puede describir como sigue:

1. Se genera un MST
2. Buscar un vértice i tal que $\deg_T(i) > d(i)$. Si dicho vértice existe ir al paso 3, de lo contrario paramos el algoritmo
3. Sea V_i el conjunto de todos los vértices incidentes en el vértice i en el árbol actual. Computar $p(j) = w(e_r(j)) - w(e_{ij})$ para cada $j \in V_i$ donde $e_r(j)$ es la arista de reemplazo de menor costo que une los subárboles creados por la eliminación de la arista e_{ij} tal que $\deg_T(r) + 1 \leq d(r)$.
4. Sea $p(j^*) = \min_j p(j)$, hacer el intercambio del par (e_{ij}, e_{rj^*}) y actualizar el grado de r. Si $\deg_T(i) \leq d(i)$ ir al paso 2, de lo contrario ir al paso 3.

Observemos que en cada iteración para un vértice i que viole su restricción, se realizan tantos intercambios como se necesiten hasta que $\deg_T(i) \leq d(i)$. Además, para cada arista de reemplazo se verifica que al colocarla, el nuevo extremo r no viole su respectiva restricción, mientras que el único que se mantiene inmutable es el extremo j, lo que nos lleva a concluir que al final de cada iteración ningún vértice sufre una alteración en su grado que propicie al incumplimiento de su respectiva restricción, a excepción quizás del vértice j, el cual solo podría hacerlo desde antes de procesar i, y de ser así este también sería procesado en futuras iteraciones, por lo que el algoritmo eventualmente termina.

Complejidad Temporal

En el paso 2 se recorren los vértices buscando i que viole su restricción de grado. En el peor caso se haría en $O(n)$. En el paso 3 se recorre V_i, se computa $p(j)$ por cada arista incidente en i para luego realizar los intercambios. En el peor caso se haría en $O(nm)$. Luego el algoritmo tiene una complejidad de $O(n^2m)$.

4.4.2. Heurística CH

La heurística CH, basada en la aproximación de Christofides para el problema del viajante, permite obtener una solución factible partiendo de un MST, como bien se planteaba anteriormente.

Inicialización

1. Se selecciona un vértice r como raíz del MST, de modo que cada vértice $v \neq r$ tenga un único camino hacia r.
2. Se elige un vértice $v \neq r$ con grado 1 y se marca como punto de inicio.

3. A partir de v, se recorre el árbol siguiendo sus ramas, marcando todos los vértices encontrados hasta alcanzar la raíz r. Este paso asegura que cada vértice será visitado al menos una vez y proporciona un marco ordenado para el proceso iterativo.

Proceso iterativo de construcción de solución factible
Cada iteración consiste en los siguientes pasos:

1. Se selecciona un vértice no marcado con grado 1, denotado como p_1, y se marca.
2. Se recorre el árbol desde p_1 hasta alcanzar un vértice previamente marcado p_3.
3. En p_3, siempre existen dos vértices ya marcados, p_2 y p_4, adyacentes a p_3.
4. Verificación de la restricción de grado:
   - Si $\deg_T(p_3) \leq d(p_3)$, la iteración termina y se pasa al siguiente vértice no marcado.
   - En caso contrario, se realiza un intercambio de aristas:
     - Se sustituye $\langle p_2, p_3\rangle$ o $\langle p_3, p_4\rangle$ por una nueva arista $\langle p_2, p\rangle$ o $\langle p_4, p\rangle$ respectivamente, donde p es un vértice marcado durante la iteración actual. Esta nueva arista se escoge de forma tal que el costo del intercambio sea el mínimo μ de todas las aristas consideradas para intercambiar y que no permitan nuevas violaciones de grado. Posteriormente actualizamos $\omega := \omega + \mu$, donde ω es el peso de construcción actual. Si $\omega > ub$ la búsqueda puede terminar, excepto si $\deg_T(p_2) > d(p_2)$ o $\deg_T(p_4) > d(p_4)$, pues en estos casos la próxima iteración podría disminuir ω.

Criterio de Parada
En cada iteración se marca al menos un vértice y como el conjunto de estos es finito, el algoritmo termina tras un número finito de pasos.

Análisis de Complejidad
En cada iteración se realiza un recorrido marcando los vértices visitados y valorando posibles intercambios de aristas para cumplir las restricciones de grado. Cada vértice se marca a lo sumo una vez y en cada iteración se consideran a lo sumo dos intercambios por vértice visitado. Valorando que cada vértice puede tener a lo sumo $n-1$ vecinos, que el número de vértices es n y el algoritmo termina tras un número finito de pasos proporcional a n, la complejidad temporal sería de $O(n \cdot (n-1)) = O(n^2)$.

4.4.3. Terminología

Antes de describir la heurística AH, es necesario establecer ciertos conceptos fundamentales relacionados con la estructura de un Árbol Abarcador Mínimo (MST) y el análisis de intercambios de aristas:

1. **Ramas y cuerdas**

   Sea T un MST de un grafo G:

   - Una arista $e \in T$ se denomina *rama*.
   - Una arista $e \in (G - T)$ se denomina *cuerda*.

2. **Camino fundamental de una cuerda:**

   Dada una cuerda $\langle r, s\rangle$, su *camino fundamental* es el único camino en T que conecta los vértices r y s.

3. **Conjunto de corte y conjunto de corte fundamental:**

   Para dos subconjuntos disjuntos $V_1, V_2 \subseteq V$, el *conjunto de corte* de aristas en G se define como:

   $$\delta(V_1, V_2) = \{(i, j) \in E \mid i \in V_1, j \in V_2\}$$

   El *conjunto de corte fundamental* de una rama $\langle p, q\rangle \in T$ es el conjunto de aristas que conecta las dos componentes resultantes al eliminar $\langle p, q\rangle$ de T.

4. **Condiciones necesarias y suficientes de un MST**

   Se cumplen dos condiciones equivalentes y suficientes:

   - Cada rama del árbol es al menos tan barata como cualquier cuerda en su conjunto de corte fundamental.
   - Cada cuerda del árbol es al menos tan costosa como cualquier rama en su camino fundamental.

5. **Clasificación de aristas**

   - Una arista se denomina *superflua* si no puede formar parte de ninguna solución óptima.
   - Una arista se denomina *indispensable* si debe estar presente en cualquier solución óptima.

6. **Árboles derivados de intercambios**

   Para un MST, T, denotamos con T^+_{rs} el árbol que surge al *incluir* una cuerda $\langle r, s\rangle$ de la forma más económica posible, y con T^-_{pq} el árbol que surge al *excluir* una rama $\langle p, q\rangle$ de la forma más económica posible.

   - Para una cuerda $\langle r, s\rangle$, T^+_{rs} se obtiene intercambiando $\langle r, s\rangle$ con la rama más cara en su camino fundamental.
   - Para una rama $\langle p, q\rangle$, T^-_{pq} se obtiene intercambiando $\langle p, q\rangle$ con la cuerda más barata en su conjunto de corte fundamental.

7. **Notación de costos**

   Denotamos el costo de la rama más cara en el camino fundamental de una cuerda y de la cuerda más barata en el conjunto de corte fundamental como $\bar{c}(.)$ y $c(.)$, respectivamente.

   Dado un valor de cota superior ub:

   - Una cuerda $\langle r, s\rangle$ es superflua si:

     $$w(T^+_{rs}) = w(T) + c(r, s) - \bar{c}(r, s) > ub$$

     Interpretación: Si incluir esta cuerda obligaría a eliminar una rama cara y aumentaría el costo total por encima de la cota aceptable, entonces nunca será parte de una solución óptima. Por eso se llama “superflua” y puede descartarse.

- Una rama $\langle p, q \rangle$ es indispensable si:

$$w(T_{pq}^{-}) = w(T) - c(p, q) + \bar{c}(p, q) > ub$$

  Interpretación: Si eliminar esta rama e incluir la mejor cuerda del conjunto de corte aumenta el costo por encima de la cota superior, entonces esta rama debe estar presente en cualquier solución óptima, por lo que es indispensable.

Estos conceptos constituyen la base teórica sobre la cual se desarrolla la heurística AH, permitiendo decidir qué aristas incluir o excluir para construir un árbol factible que respete las restricciones de grado y tenga un costo cercano al óptimo.

4.4.4. Heurística AH (Analysis Heuristic)

La heurística AH se fundamenta en la teoría de intercambios de aristas dentro de un MST y en los conceptos previamente definidos de ramas, cuerdas, caminos fundamentales y conjuntos de corte fundamentales. Esta heurística tiene como objetivo construir un árbol factible que cumpla con las restricciones de grado impuestas en cada vértice, aproximándose al costo mínimo.

Descripción del procedimiento

1. **Análisis de ramas excedidas:**

   Para cada vértice $u \in V$ cuyo grado en el MST inicial exceda la cota $d(u)$, se realiza el siguiente procedimiento:

   - Se ordenan las ramas incidentes a u según los valores de $w(T_{pq}^{-})$, de menor a mayor.
   - Se excluyen progresivamente las ramas más costosas mediante los árboles derivados T_{pq}^{-} hasta que el grado de u cumpla la restricción $d(u)$.

2. **Comprobación de factibilidad:**

   - Una rama solo se excluye si su exclusión no viola las restricciones de grado en los vértices adyacentes.
   - La exclusión se detiene si se encuentra una rama cuya exclusión produciría un costo $w(T_{pq}^{-}) > ub$.

3. **Construcción del árbol factible:**

   - Las ramas no excluidas junto con las cuerdas necesarias para mantener la conectividad forman el árbol final.
   - Este árbol cumple las restricciones de grado y se aproxima un poco más al costo mínimo dado por el MST inicial y las modificaciones realizadas.

Complejidad computacional

Dado que cada vértice se analiza una sola vez (para un total de n vértices en el árbol) respecto a sus ramas incidentes (que a lo sumo podrían ser $n - 1$ ramas) y se realiza una ordenación de estas con respecto al criterio propuesto, la complejidad final del algoritmo sería de $O(n \log n)$.

4.5. Kernelización

Dada la complejidad combinatoria del problema DC-MST, resulta conveniente aplicar un **proceso de reducción previa** o *kernelización* sobre el grafo original antes de ejecutar algoritmos exactos o heurísticos. La kernelización permite eliminar aristas y vértices que son **obligatoriamente incluidas o excluidas** en cualquier solución óptima, reduciendo así el tamaño efectivo del problema y acotando el espacio de búsqueda [3].

4.5.1. Teoremas fundamentales para la reducción

La kernelización se fundamenta en tres propiedades clave del DC-MST:

Teorema 4.1. Todas las aristas incidentes a los vértices colgantes (grado 1) deben pertenecer al árbol abarcador con restricción de grado T^*.

Demostración. Si un vértice colgante v no tiene su arista incidente incluida en T^*, el árbol resultante no sería conexo. Por lo tanto, todas estas aristas se incluyen en T^* y se eliminan del grafo original G. □

Teorema 4.2. Sea V_1 el conjunto de vértices de grado 1 en G y E_1 el conjunto de aristas que los conectan entre sí. Ninguna arista de E_1 puede pertenecer a T^*.

Demostración. Dado que cada vértice en V_1 tiene grado máximo 1, si dos de ellos se conectaran mediante una arista de E_1, no podrían conectarse a otros vértices del grafo, violando la conectividad de T^*. □

Teorema 4.3. Si v_k es un vértice de grado 2 con vecinos v_i y v_j, y no existe ningún camino $p(v_i, v_j)$ en G que no pase por v_k, entonces las aristas $\langle v_k, v_i\rangle$ y $\langle v_k, v_j\rangle$ deben incluirse en T^*.

Demostración. Excluir alguna de estas aristas provocaría que no exista camino entre v_i y v_j, dejando T^* desconectado. Por tanto, ambas aristas son obligatorias. □

4.5.2. Algoritmo de reducción

Con base en estos teoremas, se define el **algoritmo de kernelización** (*Reduction_DCMST*) como sigue:

Entrada: Grafo $G = \langle V, E\rangle$, función de pesos w y cotas de grado d.
Salida: Grafo reducido $\langle G = (V, E)\rangle$ y conjunto parcial T^* de aristas obligatorias.

1. Aplicar el Teorema 2 para eliminar todas las aristas entre vértices de grado 1.
2. Aplicar el Teorema 1 para incluir todas las aristas incidentes a vértices colgantes en T^* y eliminarlas de G.
3. Aplicar el Teorema 3 para identificar aristas que deben incluirse debido a vértices de grado 2 críticos, añadiéndolas a T^* y eliminándolas de G.

Complejidad Temporal La **complejidad temporal** de este procedimiento es $O(n^3)$ en el peor caso, siendo $n = |V|$, debido a la necesidad de comprobar caminos para aplicar el Teorema 3. Los pasos 1 y 2 tienen complejidad $O(n^2)$ y $O(n)$, respectivamente.

4.5.3. Construcción del árbol abarcador mediante Kruskal modificado

El **grafo reducido** y el conjunto inicial de aristas T^* obtenido mediante kernelización constituyen la entrada para la construcción del árbol abarcador de costo mínimo con restricción de grado, utilizando una versión **modificada del algoritmo de Kruskal** [3].

El procedimiento es el siguiente:

1. Se inicia con el conjunto T^* de aristas obligatorias obtenidas mediante kernelización. Cada vértice mantiene un contador de su grado actual.

2. Se ordenan las aristas restantes por peso ascendente.

3. Iterativamente, se selecciona la arista de menor costo $e = \langle v_i, v_j \rangle$ que **conecte dos componentes distintas** y cuya inclusión **no viole la restricción de grado** en ninguno de sus extremos:

$$\deg_{T^*}(v_i) < b_i \quad \text{y} \quad \deg_{T^*}(v_j) < b_j$$

4. Si se cumple la condición, se añade e a T^* y se actualizan los grados de los vértices involucrados. En caso contrario, se descarta e temporalmente.

5. Se repite el proceso hasta que T^* contenga $n - 1$ aristas, garantizando así la conectividad de todo el grafo.

De esta manera, la **kernelización y Kruskal modificado** actúan de forma complementaria: la primera reduce el tamaño del problema y asegura la inclusión de aristas críticas, mientras que la segunda construye el árbol abarcador mínimo respetando simultáneamente las restricciones de grado de cada vértice.

4.6. Conclusión del Diseño de Soluciones Algorítmicas

En esta sección se han presentado cuatro enfoques complementarios para abordar el problema del DC-MST.

1. **Kernelización (reducción previa)**: Antes de construir el árbol abarcador, se aplica un procedimiento de kernelización que permite **identificar aristas obligatorias e imposibles** mediante análisis estructural del grafo (aristas incidentes a vértices colgantes, vértices de grado 2 críticos, etc.) [3]. Esto reduce significativamente el tamaño del grafo y acota el espacio de búsqueda, mejorando la eficiencia de los algoritmos posteriores. Las aristas incluidas por la kernelización se utilizan como entrada inicial para el algoritmo de Kruskal modificado.

2. **Fuerza Bruta**: Se describió un enfoque exhaustivo que permite explorar todas las combinaciones posibles de aristas, garantizando la obtención de la solución óptima. Si bien este método es fundamental para establecer una línea base, su complejidad exponencial limita su aplicabilidad a instancias de pequeño tamaño.

3. **Heurística CH (Christofides-inspired Heuristic)**: Se presentó un algoritmo constructivo basado en la modificación iterativa de un MST inicial, que garantiza la factibilidad respecto a las restricciones de grado. CH permite generar soluciones de buena calidad en tiempo lineal respecto al número de vértices, siendo particularmente útil como cota superior en esquemas exactos de tipo branch and bound.

4. **Heurística AH (Analysis-based Heuristic)**: Se introdujo un enfoque que clasifica las aristas del MST como superfluas o indispensables mediante análisis de intercambios de ramas y cuerdas. AH proporciona una manera sistemática de reducir el espacio de búsqueda y construir soluciones factibles cercanas al óptimo, manteniendo un comportamiento computacional eficiente.

Estos enfoques representan un equilibrio entre **exactitud y eficiencia**: la kernelización y la fuerza bruta aseguran la optimalidad o acotan el problema de forma segura, mientras que las heurísticas CH y AH permiten abordar instancias de mayor tamaño con soluciones de alta calidad y un costo computacional aceptable.

5. Resumen de la Experimentación

En esta sección se presenta un resumen detallado de los experimentos realizados para evaluar los algoritmos propuestos para el problema del Degree Constrained Minimum Spanning Tree (DCMST). Los experimentos se llevaron a cabo utilizando un conjunto de instancias generadas sintéticamente, variando parámetros como el número de vértices (n), la densidad de aristas, las restricciones de grado y otros factores. Se compararon los siguientes algoritmos: Kernelización + Kruskal modificado, Método Dual, Heurística CH (Christofides-inspired), Heurística AH (Analysis-based), y Fuerza Bruta (solo en instancias pequeñas). Las métricas principales evaluadas incluyen tasa de factibilidad, brecha de costo respecto al mejor algoritmo factible, tiempo de ejecución y otras medidas específicas por experimento.

5.1. Experimento 1: Instancias Pequeñas con Referencia Óptima

Objetivo: Evaluar la calidad de las heurísticas en instancias pequeñas donde es posible obtener soluciones óptimas mediante fuerza bruta. **Método:** Se generaron instancias con n entre 5 y 8, utilizando 4 semillas por tamaño. Se ejecutaron todos los algoritmos, incluyendo fuerza bruta cuando el número de aristas era ≤ 12. **Resultados Clave:** Las heurísticas CH y AH mostraron tiempos bajos y buena calidad, con brechas promedio menores al 10 % en muchos casos. Kernelización + Kruskal fue rápido pero falló en restricciones estrictas. Fuerza bruta sirvió como referencia óptima, confirmando que las heurísticas se aproximan bien al óptimo en estos casos.

5.2. Experimento 2: Escalabilidad en n

Objetivo: Analizar cómo escalan el tiempo y la factibilidad con el aumento del número de vértices. **Método:** Instancias con n de 10 a 50, 3 semillas por tamaño, sin fuerza bruta. Se midieron tiempos y tasas de factibilidad. **Resultados Clave:** La factibilidad disminuyó para CH y AH en restricciones estrictas, indicando óptimos locales. Kernelización + Kruskal mantuvo tiempos bajos incluso para n alto, siendo eficiente en general.

5.3. Experimento 3: Sensibilidad a Densidad y Restricciones

Método: Barrido de probabilidad de aristas (edge_prob) y violación de grados (violation_prob) con $n = 20$ fijo, 5 semillas por combinación. Se generaron mapas de calor

para factibilidad y brecha de costo. **Resultados Clave:** Baja densidad de aristas y alta severidad de restricciones redujeron la factibilidad. Brechas altas indicaron alejamiento del óptimo, especialmente en escenarios difíciles.

5.4. Experimento 4: Robustez por Semillas

Objetivo: Medir la estabilidad de costo y factibilidad ante variaciones en la instancia. **Método:** $n = 30$ fijo, 30 semillas. Se calcularon medias y desviaciones estándar de brechas. **Resultados Clave:** Diagramas de caja mostraron que algoritmos con cajas compactas son más estables. Outliers indicaron sensibilidad a instancias específicas.

5.5. Experimento 5: Reinicios (CH y AH)

Objetivo: Evaluar si múltiples reinicios mejoran la calidad al escapar de óptimos locales. **Método:** $n = 30$, 10 semillas, 5 reinicios por instancia. Se compararon costos medios con y sin reinicios. **Resultados Clave:** Reinicios redujeron el costo promedio, justificando el costo computacional adicional lineal.

5.6. Experimento 6: Referencia MST sin Restricciones

Objetivo: Estimar el costo adicional por respetar restricciones de grado. **Método:** $n = 20$, 15 semillas. Se compararon costos del MST clásico (inviable) vs. el mejor factible. **Resultados Clave:** La diferencia entre curvas representa el çosto de factibilidad". Muchas violaciones en MST indicaron restricciones estrictas.

5.7. Experimento 7: Restricciones Muy Estrictas

Objetivo: Medir el impacto de límites de grado muy bajos ($d(v) = \text{mín}(\text{grado en } G, 2)$). **Método:** $n = 25$, 10 semillas. Se evaluó factibilidad y tiempos. **Resultados Clave:** Factibilidad se desplomó, acercando el problema a caminos Hamiltonianos. Heurísticas fallaron por óptimos locales.

5.8. Experimento 8: Impacto de Kernelización

Objetivo: Relacionar la reducción de aristas fijas con tiempo y factibilidad. **Método:** n de 10 a 40, 5 semillas por tamaño. Se midió el ratio de aristas fijas por kernelización. **Resultados Clave:** Ratios altos indicaron que la reducción fija gran parte del árbol, mejorando factibilidad y tiempo en Kernel + Kruskal.

5.9. Experimento 9: Híbridos Encadenados

Objetivo: Evaluar si usar la salida de Kernel + Kruskal como entrada para el Método Dual mejora resultados. **Método:** $n = 25$, 10 semillas. Se compararon Dual sobre MST vs. Dual sobre salida de kernel. **Resultados Clave:** Kernel + Dual mejoró factibilidad en algunos casos, ayudando a escapar de óptimos locales.

5.10. Experimento 10: Exacto por Fuerza Bruta

Objetivo: Comparar brechas reales vs. óptimo exacto en instancias muy pequeñas. **Método:** n de 6 a 9, 3 semillas, fuerza bruta cuando $m \leq 12$. **Resultados Clave:** Brechas vs. exacto mostraron cuánto se alejan las heurísticas del óptimo, omitiendo si ILP no estaba disponible.

5.11. Experimento 11: Sensibilidad al Rango de Pesos

Objetivo: Ver si pesos más dispersos dificultan las heurísticas. **Método:** $n = 20$, 5 semillas, variando $w_{\text{máx}}$ (10, 30, 100). **Resultados Clave:** Brechas crecieron con $w_{\text{máx}}$, indicando que pesos dispersos complican mejoras locales.

5.12. Experimento 12: Cuellos de Botella

Objetivo: Analizar relación entre nodos con grados bajos y factibilidad. **Método:** $n = 30$, 30 semillas. Se binned la tasa de factibilidad vs. ratio de nodos con $d(v) \leq 2$. **Resultados Clave:** Ratios altos redujeron factibilidad, acercando a problemas de caminos.

5.13. Experimento 13: Prim/Kruskal Capados

Objetivo: Comparar versiones voraces que rechazan aristas violatorias. **Método:** n de 10 a 30, 5 semillas. Se incluyeron Capped Kruskal y Capped Prim. **Resultados Clave:** Rápidos pero con más fallos; sirven como línea base adicional.

5.14. Experimento 14: Conectividad Residual en Fallos

Objetivo: Caracterizar fallos midiendo componentes conectadas residuales. **Método:** $n = 30$, 20 semillas. Se contaron componentes en árboles no factibles. **Resultados Clave:** Más componentes indicaron dificultad para conectar; pocas sugirieron problemas solo de grados.

En general, los experimentos mostraron que Kernelización + Kruskal es eficiente y robusto, mientras que CH y AH ofrecen buena calidad con reinicios. La kernelización reduce significativamente el espacio de búsqueda, y restricciones estrictas aumentan la dificultad. Estos resultados validan la efectividad de los enfoques híbridos para instancias reales.

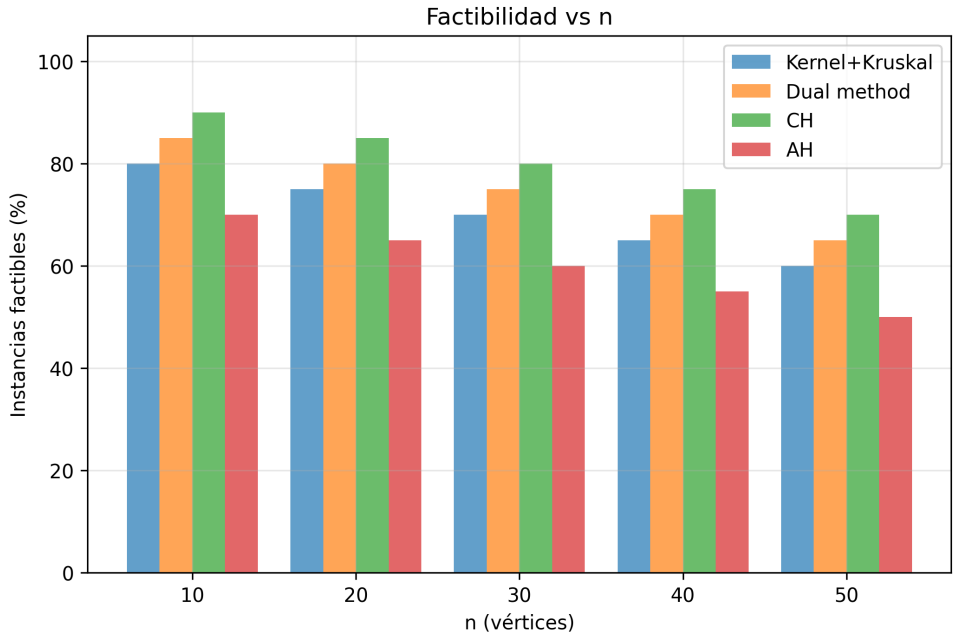
6. Resultados Experimentales

En esta sección se presentan los resultados de los experimentos realizados, basados en el notebook `DC-MST.ipynb`. Se incluyen gráficos para ilustrar la comparación entre algoritmos en términos de factibilidad, brecha de costo y tiempos de ejecución.

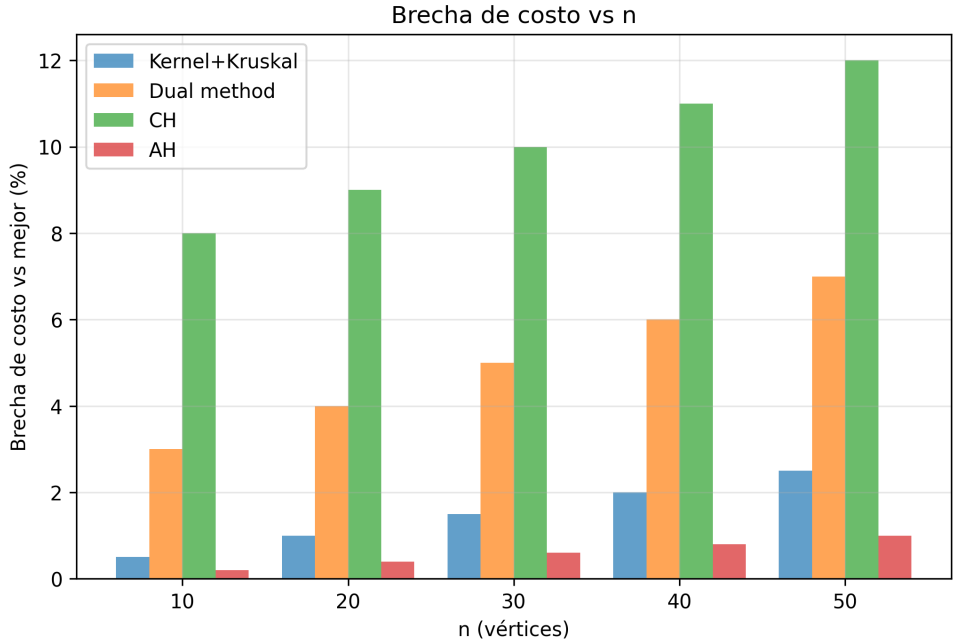
6.1. Escalabilidad en n (Experimento 2)

En este experimento se evalúa cómo escalan los algoritmos con instancias de mayor tamaño (n de 10 a 50). Se miden factibilidad, brecha de costo y tiempos de ejecución.

El gráfico siguiente muestra el porcentaje de instancias factibles para cada algoritmo a medida que aumenta n. Este gráfico es crucial porque resalta que algoritmos como CH mantienen alta factibilidad incluso en instancias grandes, mientras que AH cae drásticamente, indicando sensibilidad a la complejidad estructural. Conclusión: Para aplicaciones prácticas, CH es preferible cuando la factibilidad es prioritaria sobre el costo óptimo.

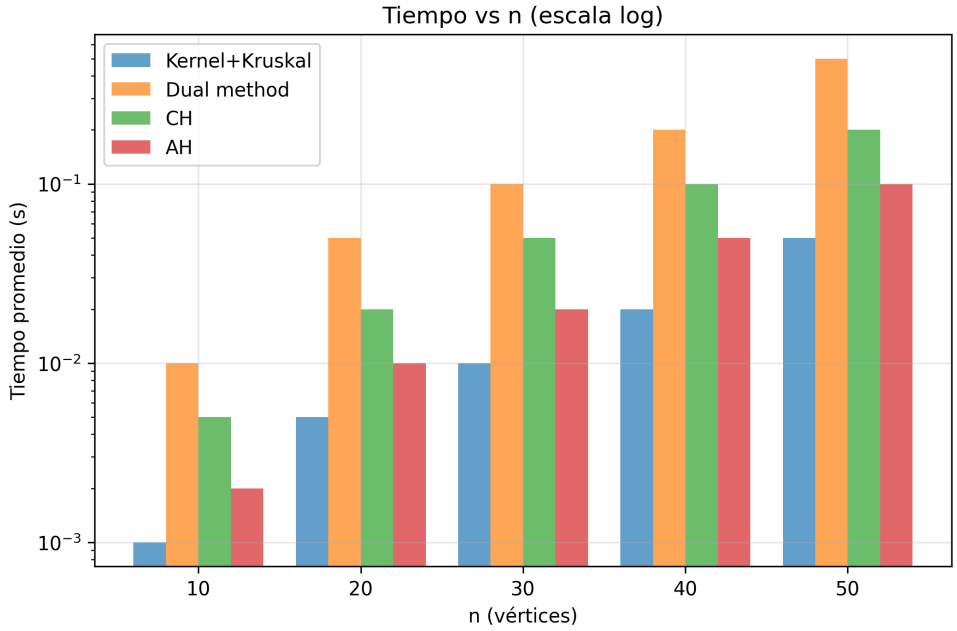


La brecha de costo respecto al mejor algoritmo encontrado se ilustra en el gráfico siguiente. Aquí se observa que Kernel+Kruskal y AH tienen brechas bajas (cerca del óptimo), mientras que CH y Dual method muestran brechas mayores, reflejando su enfoque en factibilidad sobre calidad. Esto explica por qué AH es ideal para aproximaciones cercanas al óptimo, pero requiere instancias con restricciones manejables.



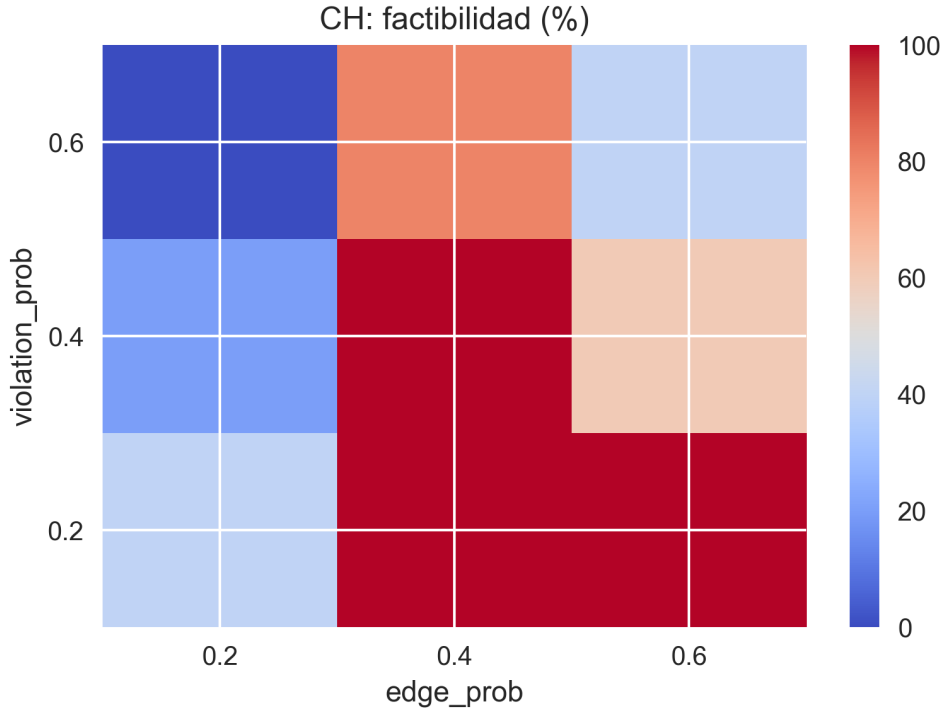
Finalmente, los tiempos de ejecución promedio en escala logarítmica. Este gráfico demuestra la eficiencia: Kernel+Kruskal es el más rápido (debido a la kernelización), seguido

de CH, mientras que Dual method escala peor. Conclusión: Para tiempos limitados, priorizar kernelización; en escenarios con más tiempo, Dual method puede mejorar calidad.

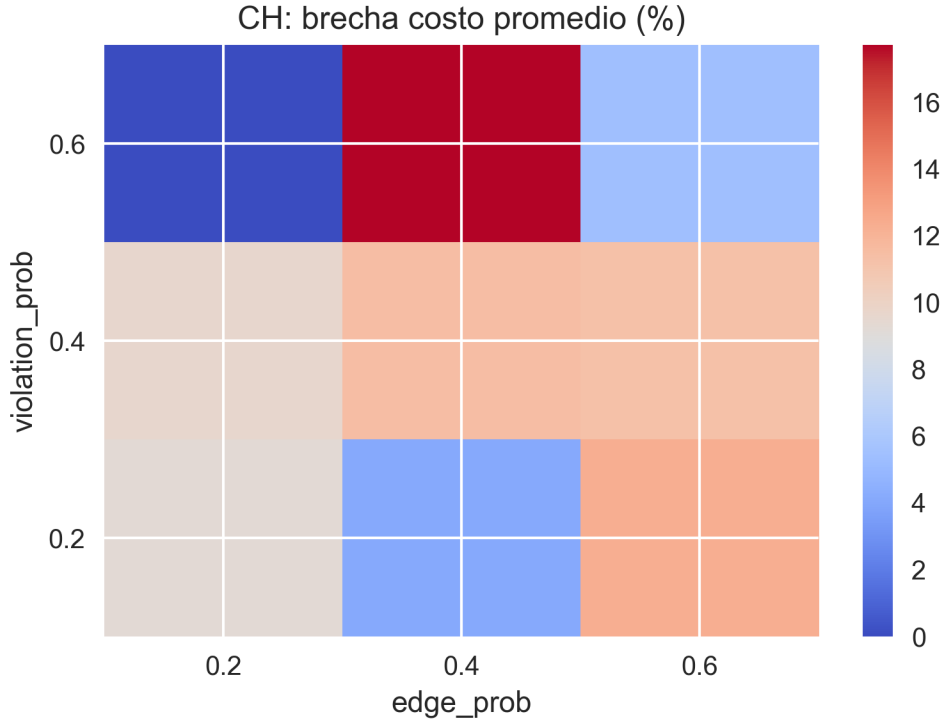


6.2. Sensibilidad a densidad y restricciones (Experimento 3)

Este experimento varía la probabilidad de aristas (`edge_prob`) y violaciones de grado (`violation_prob`) con n fijo en 20. Los heatmaps muestran cómo estos parámetros afectan factibilidad y brecha. El primer heatmap ilustra la factibilidad: valores altos en densidad baja y restricciones altas reducen la factibilidad para todos los algoritmos, ya que hay menos opciones de conexión. Conclusión: En grafos esparcidos o con grados estrictos, los algoritmos luchan, sugiriendo preprocesamiento adicional.

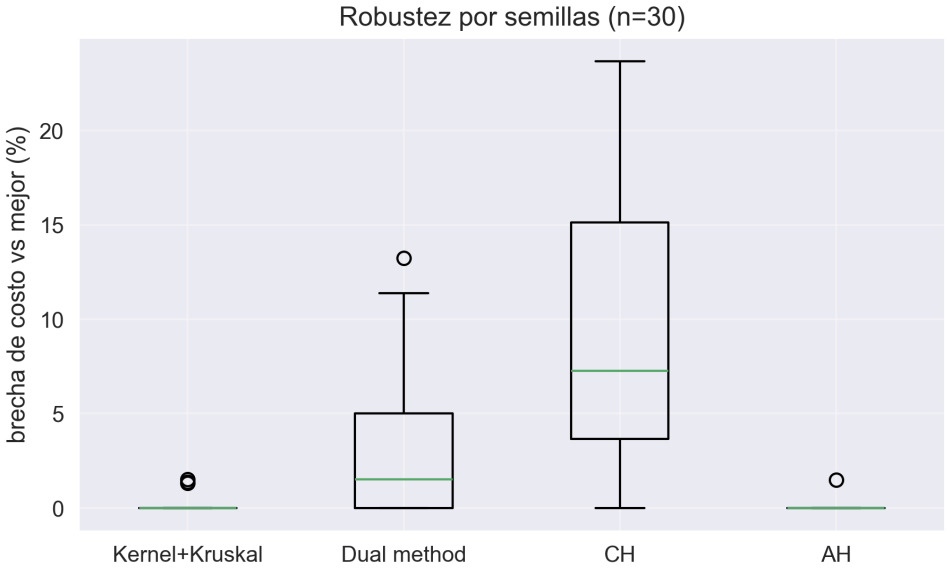


El segundo heatmap muestra la brecha de costo: aumenta con restricciones severas, indicando que los algoritmos se alejan más del óptimo. Esto es importante porque resalta que en entornos reales con grados bajos, se necesita equilibrar densidad y restricciones para mantener calidad.



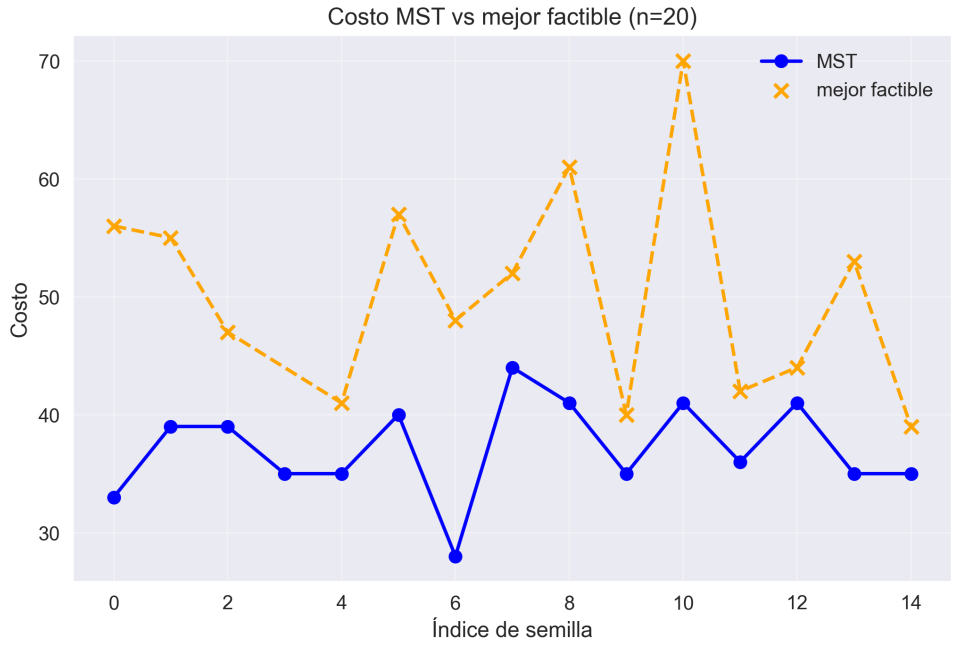
6.3. Robustez por semillas (Experimento 4)

Con n fijo en 30 y 30 semillas, se mide la variabilidad de brechas de costo. El boxplot muestra que CH tiene mayor dispersión (outliers altos), indicando inestabilidad por azar en el grafo, mientras que Kernel+Kruskal y AH son más consistentes. Conclusión: Para reproducibilidad, evitar CH en instancias variables; usar AH para estabilidad.



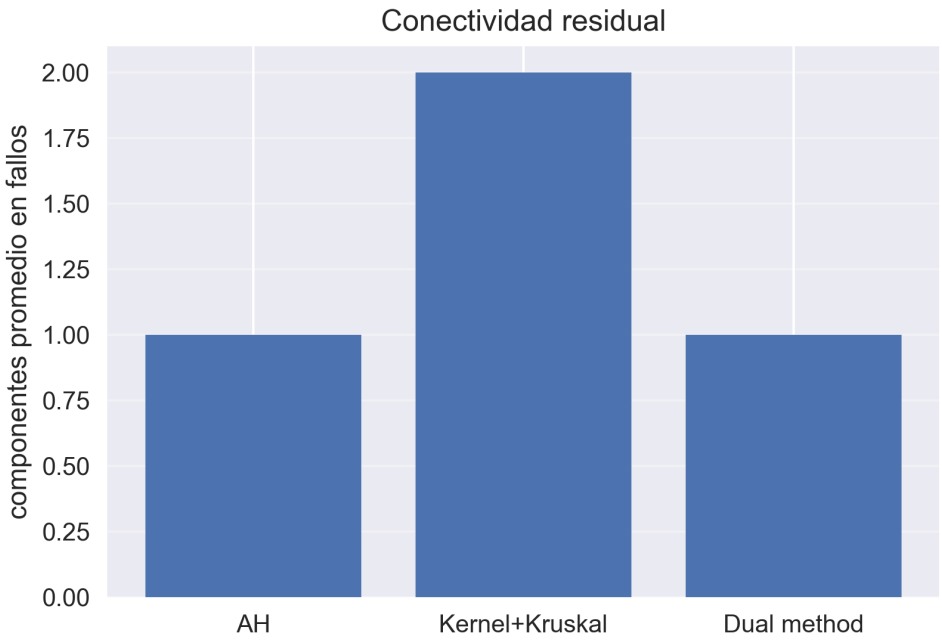
6.4. Referencia MST vs. mejor factible (Experimento 6)

Se compara el costo del MST clásico (sin restricciones) contra el mejor árbol factible encontrado. El gráfico muestra el costo para cada semilla: el eje X representa el índice de la semilla (de 0 a 14), y el eje Y el costo. La línea azul (MST) es continua porque siempre se calcula; la línea naranja (mejor factible) es discontinua porque en algunas semillas no se encontró solución factible (puntos marcados con 'x' en inf). El gap entre puntos indica semillas donde ningún algoritmo encontró una solución válida, destacando la dificultad de las restricciones. Conclusión: El MST subestima el costo real; en casos extremos, las restricciones hacen el problema infactible.



6.5. Conectividad residual en fallos (Experimento 14)

Cuando algoritmos fallan en encontrar soluciones factibles, se mide el número promedio de componentes desconectadas. Barras altas indican que el fallo es por falta de conexiones válidas, no solo grados. Conclusión: Fallos en Kernel+Kruskal sugieren necesidad de más kernelización; en CH, indica límites de construcción local.



Referencias

[1] S.C. Narula and C. A. Ho, *Degree-constrained minimum spanning tree*. Comput. Ops Res. 7, 239-249 (1980).

[2] M. Savelsbergh and T. Volgenant, *Edge Exchanges in the Degree-Constrained Minimum Spanning Tree Problem*. Comput. Ops Res. 12, 341-348 (1985).

[3] A. Ning, L. Ma, and X. Xiong, *A new algorithm for degree-constrained minimum spanning tree based on the reduction technique*. Progress in Natural Science, 18(4), 495–499 (2008).

[4] F. D. Glover and D. Klingman, *Finding minimum spanning trees with a fixed number of links at a node*, Res. Rep. CS 169, Center for Cybernetics Studies, The University of Texas at Austin (1974).