Optimización de Red de Fibra Óptica (Degree-Constrained Minimum Spanning Tree)

Diseño y Análisis de Algoritmos - Universidad de La Habana

Ernesto Abreu Peraza, Eduardo Brito Labrada

6 de enero de 2026

Resumen

Este informe detalla el análisis y la solución al problema del Árbol de Expansión Mínimo con Grado Restringido (DCMST) para la infraestructura de red de la Universidad de La Habana. Se presenta la formalización matemática, la demostración de su complejidad NP-Hard y una comparativa experimental entre diferentes algoritmos implementados en C++.

Índice

1. Formalización del Problema **3**
1.1. Modelo Matemático 3

2. Análisis de Complejidad Computacional **3**
2.1. Demostración de NP-Hardness 3
2.2. Degree-Constrained Spanning Tree 3

3. Diseño de Soluciones Algorítmicas **4**
3.1. Enumeración con Máscara de Bits para el DCMST 4
3.1.1. Descripción del Algoritmo 4
3.1.2. Análisis de Correctitud 4
3.1.3. Análisis de Complejidad 5
3.2. Enumeración Lexicográfica para el DCMST 5
3.2.1. Descripción del Algoritmo 5
3.2.2. Análisis de Correctitud 6
3.2.3. Análisis de Complejidad 6

4. Implementación y Análisis Experimental **6**

5. Conclusiones **6**

1. Formalización del Problema

Partiendo de la necesidad de interconectar los edificios de la universidad minimizando costos y respetando la capacidad de puertos de ETECSA, definimos el modelo matemático.

1.1. Modelo Matemático

Sea $G = (V, E)$ un grafo conexo y no dirigido, donde:

- V es el conjunto de edificios.
- E es el conjunto de posibles conexiones de fibra.
- $w : E \to \mathbb{R}^+$ es una función de costo.
- $k : V \to \mathbb{N}$ es la capacidad de puertos por edificio.

El problema consiste en encontrar un subgrafo $T = (V, E')$ tal que:

$$T^* = \arg\min_{T \in \mathcal{T}} \sum_{e \in E'} w(e) \tag{1}$$

Sujeto a:

1. T es un árbol de expansión de G.
2. $\forall v \in V, \deg_T(v) \leq k(v)$.

Este problema es conocido como *Degree-Constrained Minimum Spanning Tree* (DCMST).

2. Análisis de Complejidad Computacional

2.1. Demostración de NP-Hardness

Para demostrar que el problema es NP-Hard, realizamos una reducción desde el problema *Hamiltonian Path*, un conocido problema NP-Completo.

Teorema 1. *El problema DCMST es NP-Hard.*

Demostración. Dada una instancia del problema *Hamiltonian Path*, construimos una instancia del problema DCMST donde cada vértice tiene grado máximo 2. Si existe un árbol de expansión que cumpla las restricciones de grado, entonces tenemos un camino de hamilton del grafo original.

Si supieramos resolver DCMST en tiempo polinomial, podríamos resolver *Hamiltonian Path* en tiempo polinomial.

Por lo tanto, DCMST es NP-Hard. □

2.2. Degree-Constrained Spanning Tree

El problema de decisión Degree-Constrained Spanning Tree (DCST), en el cual queremos saber si existe un árbol de expansión que cumpla con las restricciones de grado, es NP-Completo. Ya que es fácil verificar que un árbol dado es de expansión de G y cumple con las restricciones de grado en tiempo polinomial.

3. Diseño de Soluciones Algorítmicas

3.1. Enumeración con Máscara de Bits para el DCMST

Esta es una solución exacta para el problema DCMST basada en **enumeración exhaustiva mediante máscara de bits**. Esta solución está diseñada para instancias de tamaño pequeño, sirviendo como algoritmo de referencia y verificación de metaheurísticas.

El algoritmo explora todos los subconjuntos posibles de aristas que contienen exactamente $|V|-1$ aristas y verifica cuáles inducen un árbol generador que cumple con la restricción de grado. Entre todas las soluciones factibles, selecciona la de costo mínimo.

3.1.1. Descripción del Algoritmo

Sea $G = (V, E)$ un grafo completo, no dirigido y ponderado, con $|V| = n$ vértices y $|E| = m = \dfrac{n(n-1)}{2}$ aristas. El algoritmo procede de la siguiente manera:

1. Se enumeran todas las aristas E y se indexan de 0 a $m-1$.
2. Se recorre cada máscara de bits $mask \in \{0,1\}^m$.
3. Solo se consideran las máscaras cuyo número de bits activos es exactamente $n-1$.
4. Cada máscara define un subgrafo G_{mask} compuesto por las aristas seleccionadas.
5. Se verifica si:
   - El subgrafo es conexo (usando Depth-First Search).
   - El grado de cada vértice es a lo sumo k.
6. Si ambas condiciones se cumplen, el sugrafo es un árbol generador factible y se evalúa su costo.
7. Se devuelve el mínimo costo entre todas las soluciones factibles.

El uso de una máscara de bits permite representar subconjuntos de aristas de forma compacta y eficiente a nivel de implementación.

3.1.2. Análisis de Correctitud

Demostraremos que el algoritmo es correcto, es decir, que devuelve exactamente el costo del árbol generador mínimo con restricción de grado.

Lema 2. *El algoritmo examina **todos** los subconjuntos de aristas de tamaño $n-1$.*

Demostración. Cada máscara de bits m representa un subconjunto único de aristas. Al iterar sobre todas las máscaras con popcount(mask) $= n-1$, se enumeran exactamente todos los subconjuntos de E con $n-1$ aristas. □

Lema 3. *Un subconjunto de aristas $E' \subseteq E$ con $|E'| = n-1$ es un árbol generador factible si y solo si:*

1. *El grafo inducido es conexo.*
2. *Para todo vértice v, $deg(v) \leq k$.*

Demostración. Un grafo conexo con $n-1$ aristas es un árbol. La segunda condición garantiza la restricción de grado. Por lo tanto, ambas condiciones son necesarias y suficientes. □

Lema 4. *La función `check` devuelve verdadero si y solo si el subgrafo inducido por la máscara es un árbol generador factible.*

Demostración. Inicialmente, verifica que ningún vértice tenga grado mayor que k. Luego, ejecuta un DFS desde el vértice 0 y comprueba que todos los vértices son alcanzables, lo que implica conectividad. Por el Lema 3, esto es equivalente a ser un DCST factible. □

Teorema 5. *El algoritmo devuelve el costo mínimo entre todos los árboles generadores que cumplen la restricción de grado.*

Demostración. Por el Lema 2, el algoritmo considera todas las soluciones candidatas. Por el Lema **??**, acepta exactamente las soluciones factibles. Finalmente, toma el mínimo costo entre ellas. Por lo tanto, el resultado es óptimo. □

3.1.3. Análisis de Complejidad

El número total de máscaras es 2^m, pero las máscaras consideradas efectivamente son $\binom{m}{n-1}$ y por cada una de ellas se hace lo siguiente:

- Construcción del subgrafo: $O(n)$.
- Verificación de grados: $O(n)$.
- DFS para conectividad: $O(n)$.

Por tanto, el costo por máscara es $O(n)$ y la complejidad total es: $O(2^m + \binom{m}{n-1} \cdot n)$.

3.2. Enumeración Lexicográfica para el DCMST

El algoritmo se basa en la enumeración exhaustiva de subconjuntos de aristas mediante una representación binaria y generación lexicográfica de combinaciones. Su aplicabilidad es sobre todo para instancias de tamaño reducido y es aplicable como algoritmo de referencia para la validación de metaheurísticas.

3.2.1. Descripción del Algoritmo

Sea $G = (V, E)$ un grafo completo con $|V| = n$ vértices y $|E| = m = \frac{n(n-1)}{2}$ aristas. El algoritmo procede como sigue:

1. Se indexan todas las aristas de E.
2. Se construye una cadena binaria `state` de longitud m con exactamente $n - 1$ bits activos.
3. Cada permutación lexicográfica de `state` representa un subconjunto distinto de $n - 1$ aristas.
4. Para cada subconjunto:
   - Se construye el subgrafo inducido.
   - Se verifica la restricción de grado.
   - Se comprueba conectividad mediante DFS.
   - Si es factible, se evalúa su costo.
5. Se devuelve el mínimo costo encontrado.

La generación de subconjuntos se realiza mediante la función `next_permutation`, lo que garantiza que cada combinación se visita exactamente una vez.

3.2.2. Análisis de Correctitud

Lema 6. *El algoritmo enumera todos los subconjuntos de aristas de tamaño $n-1$.*

Demostración. La cadena binaria inicial contiene exactamente $n-1$ unos y $m-(n-1)$ ceros. El uso de `next_permutation` genera todas las permutaciones distintas de dicha cadena, que corresponden biyectivamente a los subconjuntos de E con $n-1$ elementos. □

Lema 7. *Un subconjunto $E' \subseteq E$ con $|E'| = n-1$ es un árbol generador factible si y solo si:*

1. *El subgrafo inducido es conexo.*
2. *Para todo $v \in V$, $\deg(v) \leq k$.*

Demostración. Un grafo conexo con $n-1$ aristas es un árbol. La segunda condición garantiza la restricción de grado. Ambas son necesarias y suficientes. □

Lema 8. *La función `check` devuelve verdadero si y solo si el subgrafo inducido por la máscara es un árbol generador factible.*

Demostración. Inicialmente, verifica que ningún vértice tenga grado mayor que k. Luego, ejecuta un DFS desde el vértice 0 y comprueba que todos los vértices son alcanzables, lo que implica conectividad. Por el Lema 7, esto es equivalente a ser un DCST factible. □

Teorema 9. *El algoritmo devuelve el costo mínimo entre todos los árboles generadores que satisfacen la restricción de grado.*

Demostración. Por el Lema 6, todas las soluciones candidatas son evaluadas. Por el Lema 7, se aceptan exactamente las factibles. El algoritmo selecciona el mínimo costo entre ellas, lo que implica optimalidad. □

3.2.3. Análisis de Complejidad

El número de combinaciones evaluadas es: $\binom{m}{n-1}$ y para cada combinación:

- Construcción del subgrafo: $O(n)$.
- Verificación de grados: $O(n)$.
- DFS de conectividad: $O(n)$.

Por lo tanto, la complejidad es $O(\binom{m}{n-1} \cdot n)$.

4. Implementación y Análisis Experimental

5. Conclusiones

Referencias

[1] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed., Chapter 34). MIT Press.