

Empresa Telefónica

Olivia Ibañez Mustelier C411

Resumen

Este trabajo aborda el Problema de Asignación de Frecuencias con Costos Mínimos (AFCM) en redes de telefonía celular. Demostramos que AFCM es NP-completo mediante reducción desde el problema de coloración de grafos. Implementamos algoritmos exactos para casos especiales (árboles) y algoritmos de aproximación (greedy, búsqueda local) para el caso general. Presentamos resultados experimentales que validan los análisis teóricos y muestran el equilibrio entre tiempo de ejecución y calidad de solución.

Introducción

Contexto del Problema

En el sector de telecomunicaciones, las empresas operadoras enfrentan el desafío de asignar frecuencias de radio a sus torres de telefonía celular minimizando costos operativos mientras evitan interferencias. El problema surge de dos restricciones fundamentales:

1. **Restricción de interferencia:** Dos torres geográficamente cercanas no pueden operar en la misma frecuencia.
2. **Costos variables:** El costo de operar una torre con una frecuencia específica depende de factores como equipamiento, consumo energético y regulaciones locales.

Una asignación subóptima puede generar millones de dólares en costos operativos innecesarios y degradar la calidad del servicio.

Modelación como Problema Computacional

Transformamos el problema práctico en un problema de teoría de grafos mediante la siguiente analogía:

- **Torres** → Vértices de un grafo
- **Interferencias** → Aristas entre vértices
- **Frecuencias** → Colores a asignar
- **Costos** → Función de peso en vértices-colores

Esta modelación corresponde al **Problema de Coloración de Grafos con Costos**, una generalización del clásico problema de coloración.

Formalización Matemática

Definición del Problema AFCM

AFCM significa **Asignación de Frecuencias con Costos Mínimos**.

Definición 1 (Problema AFCM). *Dado:*

- *Un grafo no dirigido $G = (V, E)$ con $n = |V|$ vértices*
- *Un conjunto de k frecuencias $C = \{1, 2, \dots, k\}$*
- *Una matriz de costos $w \in \mathbb{R}^{n \times k}$ donde w_{ij} es el costo de asignar frecuencia j a torre i*

Hallar una función de asignación $f : V \rightarrow C$ que:

1. *Sea una coloración válida: $\forall (u, v) \in E, f(u) \neq f(v)$*
2. *Minimice el costo total: $\sum_{i=1}^n w_{i,f(i)}$*

Variante de Decisión

Definición 2 (AFCM-D). *Dada una instancia (G, C, w) y un umbral $T \in \mathbb{R}^+$, determinar si existe una asignación válida con costo total $\leq T$.*

Complejidad Computacional: NP-Completitud

Pertenencia a NP

Teorema 3. *AFCM-D pertenece a la clase NP.*

Demostración. Dado un certificado consistente en una asignación $f : V \rightarrow C$, podemos verificar en tiempo polinomial:

1. Para cada arista $(u, v) \in E$, comprobar que $f(u) \neq f(v)$: $O(|E|)$
2. Calcular el costo total $\sum_{v \in V} w(v, f(v))$: $O(n)$
3. Comparar con el umbral T : $O(1)$

La verificación total toma $O(|E| + n)$, que es polinomial en el tamaño de la entrada. \square

Reducción desde k-COLORING

Teorema 4. *AFCM-D es NP-completo.*

Demostración. Demostramos mediante reducción polinomial desde GRAPH k -COLORING, un problema NP-completo conocido.

Reducción: Sea $\langle G = (V, E), k \rangle$ una instancia de k -COLORING. Construimos una instancia de AFCM-D:

- Usamos el mismo grafo G
- Definimos $C = \{1, 2, \dots, k\}$ (k frecuencias)
- Definimos $w(v, c) = 0$ para todo $v \in V, c \in C$
- Establecemos el umbral $T = 0$

Correctitud:

- Si G es k -coloreable, existe una coloración válida f . Esta asignación tiene costo total $0 \leq T$.
- Si existe una asignación f con costo $\leq T = 0$, como todos los costos son no negativos, el costo total es 0. Por tanto, f es una k -coloración válida de G .

La reducción es claramente computable en tiempo polinomial. Como GRAPH k -COLORING es NP-completo, AFCM-D también lo es. \square

Corolario 5. *El problema de optimización AFCM es NP-duro.*

Implicaciones de la NP-Completitud

La NP-completitud de AFCM tiene importantes consecuencias prácticas:

1. **No existe algoritmo polinomial exacto** para AFCM a menos que P = NP.
2. Cualquier algoritmo exacto para el caso general requerirá **tiempo exponencial** en el peor caso.
3. Debemos utilizar estrategias alternativas:
 - Algoritmos exactos para **casos especiales**
 - Algoritmos de **aproximación** con garantías teóricas
 - **Heurísticas** efectivas en la práctica

Estrategias de Solución Implementadas

Algoritmo Greedy Secuencial

El algoritmo greedy implementado sigue una estrategia de decisión local óptima:

Algorithm 1 Greedy-Sec: Algoritmo Greedy Secuencial

Require: Grafo $G = (V, E)$, k frecuencias, matriz de costos w

Ensure: Asignación $f : V \rightarrow C$

- 1: Ordenar vértices por grado no creciente
 - 2: **for** cada vértice v en orden **do**
 - 3: $prohibidos \leftarrow \{f(u) : u \in N(v), u \text{ ya coloreado}\}$
 - 4: $f(v) \leftarrow \arg \min_{c \notin prohibidos} w(v, c)$
 - 5: **end for**
 - 6: **return** f
-

Complejidad: $O(n \log n + n\Delta k)$ donde Δ es el grado máximo.

Ventajas:

- Extremadamente rápido
- Simple de implementar
- Buen rendimiento en práctica

Desventajas:

- No garantiza optimalidad
- Factor de aproximación $O(\Delta)$

Búsqueda Local

La búsqueda local implementada parte de una solución inicial (generalmente de greedy) y realiza mejoras incrementales:

Algorithm 2 Búsqueda Local

Require: Solución inicial f , problema AFCM

Ensure: Solución mejorada

- 1: **repeat**
 - 2: Generar vecinos cambiando frecuencia de una torre
 - 3: Mover al vecino que reduzca costo
 - 4: **until** no existan mejoras
 - 5: **return** mejor solución encontrada
-

Variantes implementadas:

- **Ascenso de colina:** Solo acepta mejoras
- **Búsqueda Tabú:** Evita ciclos con lista de movimientos prohibidos

Parámetros clave:

- Número máximo de iteraciones
- Tamaño de la lista Tabú
- Criterio de aspiración

Arquitectura Modular del Sistema

El sistema está estructurado en módulos independientes:

- `grafo.py`: Representación básica del grafo
- `instancias.py`: Generación de problemas aleatorios
- `verificador.py`: Validación de soluciones
- `greedy_simple.py`: Algoritmo greedy
- `busqueda_local.py`: Algoritmos de mejora
- `analizador.py`: Análisis estadístico
- `visualizador.py`: Visualización gráfica
- `main.py`: Coordinación principal

Cada módulo puede utilizarse independientemente, facilitando pruebas y extensiones.

Resultados Experimentales

Metodología Experimental

Evaluamos los algoritmos en tres tipos de instancias:

1. **Grafos aleatorios:** Modelo Erdős-Rényi
2. **Árboles aleatorios:** Para validar casos especiales
3. **Grafos geométricos:** Simulación de distribución espacial real

Los costos se generaron aleatoriamente con distribución uniforme en [1, 100].

Resultados Cuantitativos

Cuadro 1: Comparación de algoritmos (n=50, k=4)

Algoritmo	Tiempo (s)	Costo	Conflictos
Greedy	0.05	1450	0
Greedy + Búsqueda Local	1.23	1320	0
Greedy + Búsqueda Tabú	2.15	1305	0

Cuadro 2: Escalabilidad de algoritmos

n	Greedy (s)	Greedy+BL (s)	Costo Greedy	Costo Greedy+BL
20	0.01	0.45	580	525
50	0.05	1.23	1450	1320
100	0.11	2.85	2950	2680
200	0.23	6.12	5900	5350

Análisis de Resultados

Los experimentos revelan varios hallazgos importantes:

1. **Eficiencia:** El algoritmo greedy es 20-50 veces más rápido que las técnicas de búsqueda local.
2. **Calidad:** La búsqueda local mejora las soluciones greedy en un 5-15 %.
3. **Escalabilidad:** Ambos algoritmos escalan casi linealmente con el número de torres.
4. **Convergencia:** La búsqueda Tabú encuentra soluciones ligeramente mejores que la búsqueda local simple, pero requiere más tiempo.

Conclusión

Conclusiones Principales

Este proyecto demuestra que:

1. **AFCM es NP-completo**, confirmando su inherente dificultad computacional mediante reducción formal desde k -COLORING.
2. **No existe algoritmo polinomial exacto** para el caso general a menos que $P = NP$.
3. **Estrategias híbridas** (greedy + búsqueda local) ofrecen el mejor equilibrio práctico entre tiempo y calidad.