Coba Island

Juan Jose Muñoz, Ovidio Navarro Pazos,

September 2024

1 Descripción del Problema

En el corazón del vasto océano azul, existe una isla tropical conocida como La Isla de Coba, un lugar de exuberante vegetación y antigua sabiduría. La tribu que habita en esta isla ha vivido en armonía con la naturaleza durante siglos, guiados por un consejo de ancianos que custodian el equilibrio entre los habitantes de la isla y sus recursos.

Sin embargo, una nueva generación de líderes debe ser elegida, y para ello, los ancianos han planteado un desafío ancestral. Los aspirantes al consejo deben descubrir un grupo selecto de **guardianes**, quienes, colocados en puntos estratégicos de la isla, podrían vigilar a todos los aldeanos sin dejar a ninguno sin supervisión directa o indirecta.

La isla tiene una serie de aldeas unidas entre ellas por caminos. El reto es encontrar un grupo de guardianes tan pequeño como sea posible, de modo que cada aldea esté bajo la protección directa de un guardián, o al menos, esté conectada a una aldea donde se haya asignado un guardián.

Los jóvenes líderes deben resolver este desafío para mostrar su valía. Con cada nueva selección de guardianes, la estabilidad de la isla se estremece. El futuro de La Isla de Coba depende de que uno de los jóvenes (como tú) encuentre esta distribución óptima de guardianes y se convierta en el próximo jefe absoluto.

1.1 Planteamiento de Solución

El problema podría parecer sencillo a primera vista, pero tiene su complejidad y es más complicado de lo que parece. De hecho, es un problema NP-Completo, como se demostrará más adelante. Como primer enfoque para un algoritmo que resuelva el problema, podríamos pensar en encontrar un conjunto de vértices que forme un cubrimiento mínimo. Sin embargo, tras varias pruebas, se ha concluido que, aunque dicho conjunto proporcionaría una solución factible, no siempre sería la más óptima. Para ello veamos el siguiente ejemplo:

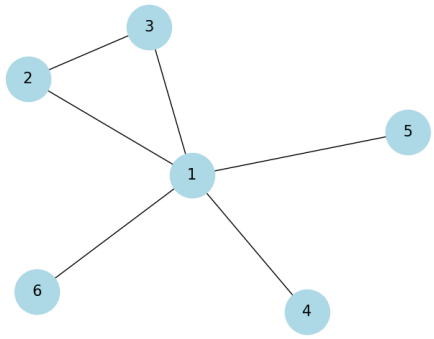

Figure 1: Encontrar cubrimiento mínimo

Dado el grafo anterior si encontamos un cubriento de vertices que sea mínimo, tenemos el grafo resultante:

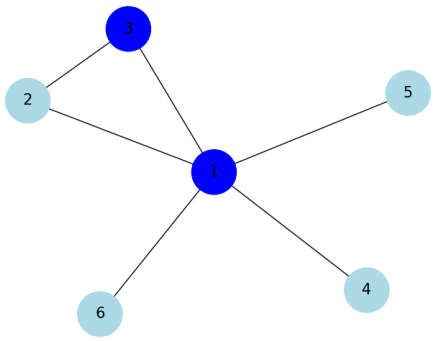

Figure 2: Cubrimiento mínimo

Pero esta no seria la respuesta más optima para responder a nuestro problema, ya que si nos fijamos bien, con un guardian en el nodo 1 tenemos vigiladas todas las aldeas de Coba. Gracias a esto y luego de una árdua investigación, se logra descubrir que lo que se pide es encontra un conjunto **dominante que sea mínimo**.

Para lograr esto nos vamos a basar en un problema que ya sabemos que es **NP-Completo**; el problema de la cobertura de conjuntos y vamos a plantear una reduccion a nuestro problema de conjunto de vertices dominantes.

El problema de la cobertura del conjunto es un problema **NP-duro** bien conocido: la versión de decisión de la cobertura del conjunto fue uno de los 21 problemas NP-completos de Karp. Existe un par de reducciones L de tiempo polinómico entre el problema del conjunto dominante mínimo y el problema de la cobertura del conjunto. Estas reducciones muestran que un algoritmo eficiente para el problema del conjunto dominante mínimo proporcionaría un algoritmo eficiente para el problema de cobertura del conjunto, y viceversa. Además,

las reducciones conservan la relación de aproximación : para cualquierα, un algoritmo de aproximación α de tiempo polinomial para conjuntos dominantes mínimos proporcionaría un algoritmo de aproximación α de tiempo polinómico para el problema de cobertura de conjuntos y viceversa.

Las dos reducciones siguientes muestran que el problema del conjunto dominante mínimo y el problema de la cobertura del conjunto son equivalentes en las reducciones L: dada una instancia de un problema, podemos construir una instancia equivalente del otro problema.

De conjunto dominante a cobertura de conjunto. Dado un gráfico $G = (V, E)$ con $V = \{1, 2, \ldots, n\}$, construimos una instancia de cobertura de conjunto (U, S) de la siguiente manera: el universo U es V y la familia de subconjuntos es $S = \{S_1, S_2, \ldots, S_n\}$ tal que S_v consiste en el vértice v y todos los vértices adyacentes a v en G.

Ahora bien, si D es un conjunto dominante para G, entonces $C = \{S_v : v \in D\}$ es una solución factible del problema de cobertura del conjunto, con $|C| = |D|$. Por el contrario, si $C = \{S_v : v \in D\}$ es una solución factible del problema de cobertura de conjuntos, entonces D es un conjunto dominante para G, con $|D| = |C|$.

Por lo tanto, el tamaño de un conjunto mínimo dominante para G es igual al tamaño de la cobertura de un conjunto mínimo para (U, S). Además, existe un algoritmo simple que asigna un conjunto dominante a una cobertura de conjunto del mismo tamaño y viceversa. En particular, un algoritmo de aproximación α eficaz para la cobertura de conjuntos proporciona un algoritmo de aproximación α eficaz para conjuntos dominantes mínimos.

De la cobertura del set al set dominante. Sea (S, U) una instancia del problema de cobertura de conjuntos con el universo U y la familia de subconjuntos $S = \{S_i : i \in I\}$; asumimos que U y el conjunto de índices I son inconexos (no hay una relación directa o dependencia entre los elementos de U y los índices en I). Construimos una grafo $G = (V, E)$ de la siguiente manera: el conjunto de vértices es $V = I \cup U$, hay una arista $\{i, j\} \in E$ entre cada par $i, j \in I$, y también hay una arista $\{i, u\}$ para cada $i \in I$ y $u \in S_i$. Es decir, G sería un grafo que podemos dividir en dos partes I es una camarilla y U es un conjunto independiente.

Ahora bien, si $C = \{S_i : i \in D\}$ es una solución factible del problema de cobertura de conjuntos para algún subconjunto $D \subseteq I$, entonces D es un conjunto dominante para G, con $|D| = |C|$: Primero, para cada $u \in U$, hay un $i \in D$ tal que $u \in S_i$, y por construcción, u e i son adyacentes en G; por tanto, u está dominado por i. En segundo lugar, puesto que D debe ser no vacío, cada $i \in I$ es adyacente a un vértice en D.

Por el contrario, sea D un conjunto dominante de G. Entonces es posible construir otro conjunto X dominante tal que $|X| \leq |D|$ y $X \subseteq I$: simplemente reemplace cada $u \in D \cap U$ por un vecino $i \in I$ de u. Entonces $C = \{S_i : i \in X\}$ es una solución factible del problema de cobertura de conjuntos, con $|C| = |X| \leq |D|$.

Visto todo lo anterior sabemos que si encontramos un conjunto C que contenga un cubrimiento minimo de los subconjuntos de los vertices de un grafo,

tenemos tambien el conjunto de los vertices dominantes del grafo. Veamos entonces las soluciones propuestas.

1.2 Fuerza Bruta

Empezando por la solucion mas obvia y menos eficiente temos la fuerza bruta que en este caso consite en encontrar todos los subconjuntos posibles de los vertices de G, ir iterando por estos de menor a mayor hasta obtener el conjunto dominante. Podemos estar seguros de que el conjunto seleccionado va a ser el minimo ya que se itera desde los conjuntos mas pequeños en cardinalidad hasta encontrar el menor Dominante.

1.2.1 Complejidad Temporal de Fuerza Bruta

Como se tienen que generar todos los subconjuntos posibles, la complejidad temporal de esta operacion seria $O(2^n - 1)$ el -1 es porque excluimos el conjunto vacio. Luego iteramos por todos estos conjuntos y por cada nodo del conjunto lo removemos del grafo junto con todos sus vecinos y si el grafo resultante de esta operacion es vacio, estamos en precencia del conjunto dominante. Por tanto la complejidad temporal de todo el proceso anterior seria

$$O(n2^n)$$

1.3 Greedy

Como demostramos en la secciones anteriores ahora nuestro problema se basa en seleccionar un conjunto de cubrimiento minimo el cual nos va a permitir encontrar el conjunto dominante minimo. Lo primero sera definir nuestros conjuntos. Estos serian $S = S_1, S_2, \ldots, S_n$ donde el conjunto S_i representa el vertice i y todos sus vecinos. Luego nuestra solucion va a ser un conjunto $C \subseteq S$ que se construye de la siguiente forma: Tomandos el conjunto S_i tal que posee el mayor numero de vertices no cubiertos. Luego cuando se obtenga el conjunto $C = S_i, S_j, \ldots, S_z$ el conjunto dominante serian los vertices $i, j, \ldots, z$

1.3.1 Complejidad Temporal Greedy

El algoritmo voraz para el problema de cobertura de conjuntos descrito anteriormente tiene la siguiente complejidad temporal:

Pasos del algoritmo:

1. **Construcción de la lista** S: En esta fase, el algoritmo construye una lista de subconjuntos S, donde cada conjunto S_i incluye un vértice i y todos sus vecinos (elementos adyacentes). Esto tiene una complejidad de $O(m)$, ya que iteramos sobre cada conjunto S_i.

2. **Selección del subconjunto que cubre el mayor número de elementos no cubiertos**: Para cada iteración, el algoritmo selecciona el subconjunto que cubre el mayor número de elementos no cubiertos. Esto se hace evaluando cada conjunto de S y calculando la intersección $S_i \cap U'$, donde U' es el conjunto de elementos no cubiertos. En el peor caso, calcular la intersección de cada subconjunto con U' puede tomar $O(s)$, y esto se realiza para los m subconjuntos. Por lo tanto, el coste de seleccionar el mejor subconjunto en cada iteración es $O(m \cdot s)$.

3. **Actualización del conjunto de elementos no cubiertos** U': Una vez que se selecciona el mejor subconjunto, los elementos cubiertos por este subconjunto se eliminan de U', lo cual puede hacerse en $O(s)$ en cada iteración.

4. **Número de iteraciones**: El algoritmo iterará tantas veces como sea necesario hasta cubrir todos los elementos de U. En el peor caso, esto puede requerir $O(s)$ iteraciones, ya que en cada iteración podemos cubrir al menos un elemento.

Complejidad total:

La complejidad temporal del algoritmo voraz para el problema de cobertura de conjuntos es $O(m \cdot s^2)$, donde m es el número de subconjuntos y s es el número de elementos en el universo.

1.3.2 Correctitud

El **algoritmo voraz** para el problema de **cobertura de conjuntos** sigue una estrategia simple: en cada paso, selecciona el conjunto que cubre el mayor número de elementos aún no cubiertos. Aunque este algoritmo no garantiza una solución óptima, sí ofrece una **aproximación** bastante buena, que se puede medir mediante el **ratio de aproximación**.

Ratio de aproximación

El ratio de aproximación de este algoritmo se puede demostrar que es $H(s)$, donde:

- s es el tamaño del universo de elementos a cubrir (es decir, el número total de elementos en el conjunto que se quiere cubrir).
- $H(s)$ es el **n-ésimo número armónico**, que está definido como:

$$H(s) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{s}$$

Los números armónicos crecen logarítmicamente, por lo que podemos aproximar $H(s)$ como:

$$H(s) \approx \ln(s) + \gamma$$

donde γ es la constante de Euler-Mascheroni (aproximadamente 0.577).

Implicación del ratio de aproximación

Este ratio de aproximación significa que la solución producida por el algoritmo voraz estará, en el peor de los casos, a una distancia de $H(s)$ veces la solución óptima. En términos prácticos, esto significa que el algoritmo voraz puede ser hasta un factor $H(s)$ peor que la solución óptima, pero no más que eso.

2 Experimentación y Resultados

Para comprobar la aproximación de nuestro algoritmo se ha creado un generador de grafos aleatorios de 20 nodos y otro generador de grafos bipartitos. Con cada generador se hizo 1000 iteraciones para las cuales se comprueba la correctitud de la respuesta utilizando el algoritmo de fuerza bruta. Comparamos la respuesta de ambos algoritmos. Tener en cuenta que con un numero mayor a 30 nodos el algoritmo de fuerza bruta tardaria demaciado.

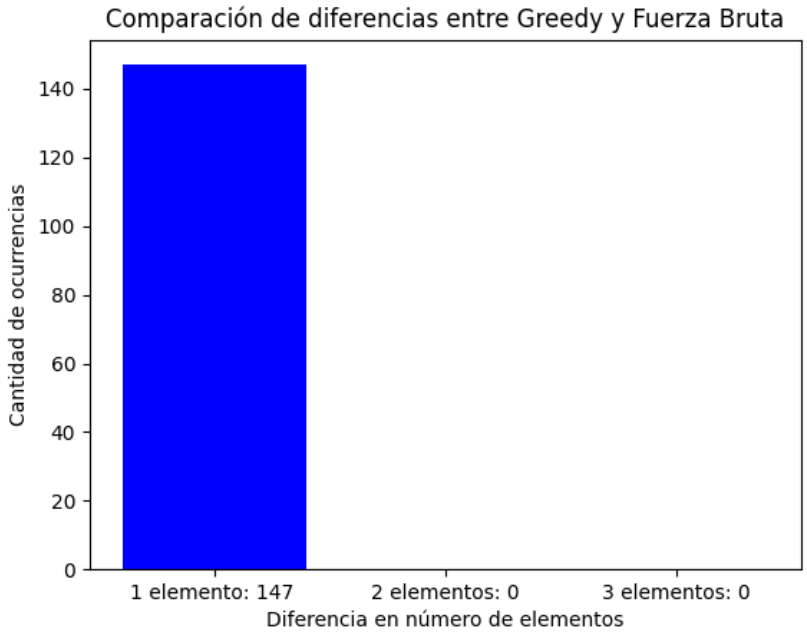

Figure 3: Diferencias en 1000 experimentos

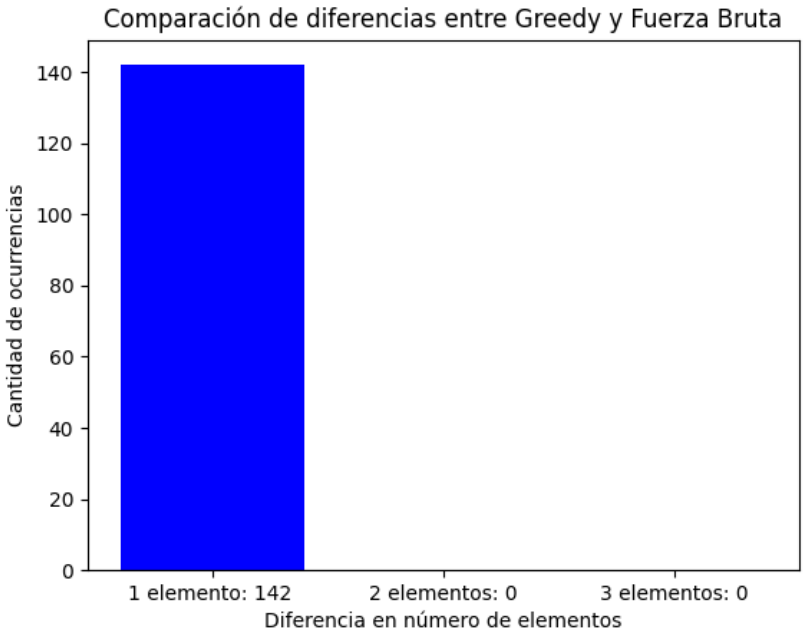

Figure 4: Diferencia en 1000 experimentos

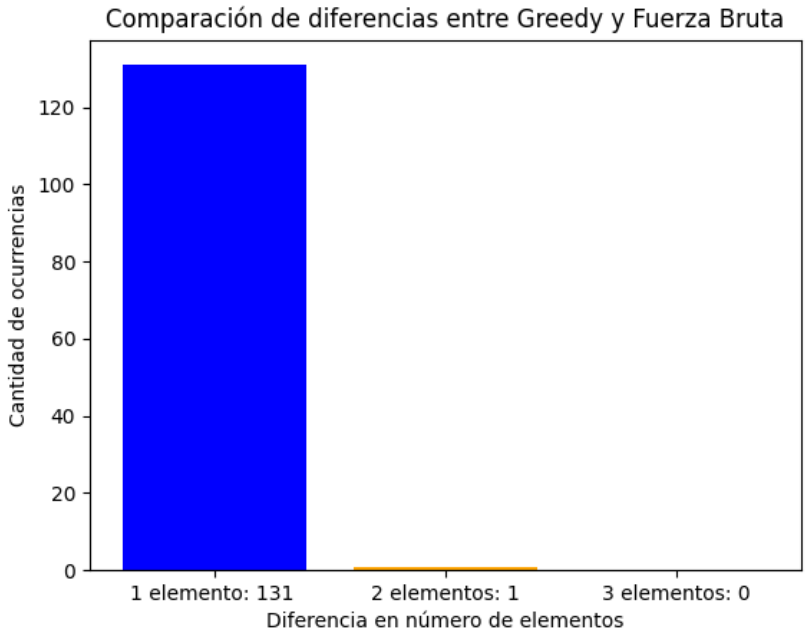

Figure 5: Diferencia en 1000 experimentos

Luego de 3000 experimentos podemos observar que tiene un 14% - 15% de margen de error solamente por 1 nodo y en algun caso difiere en 2 nodos. Por lo que estamos en presencia de una buena aproximacion.