Proyecto #3 - Dalgo

(Esteban Castelblanco Gómez 202214942, Carlos Casadiego 202212187, Felipe Lancheros 202211004)

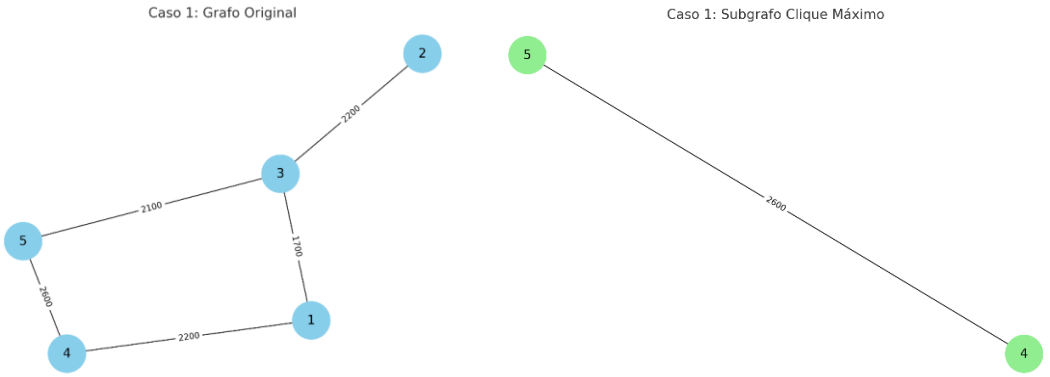
1. Explicación del algoritmo (+ detalles técnicos):

Primero es necesario tener en cuenta que el problema que estamos tratando es una variación del problema del clique el cual es NP-Completo, la única variación es que en vez de maximizar el tamaño del clique, hay que maximizar la cantidad de plata recaudada por la artista. Por lo que nuestra solución no va a ser del todo exacta, lo cual se va a explicar más adelante.

- Modelado del problema como un grafo ponderado
    - Vértices del grafo: Cada persona es representada como un vértice. Esto permite modelar las relaciones entre personas y sus aportes de manera gráfica.
    - Aristas del grafo: Representan las relaciones de conocimiento. Si dos personas (vértices) se conocen, se conectan mediante una arista. Esto es crucial para entender qué grupos de personas se conocen entre sí.
    - Pesos de los vértices: Cada vértice tiene un peso que corresponde al aporte monetario de la persona que representa. Los pesos son fundamentales para calcular el total recaudado por cualquier subconjunto de personas.
- Búsqueda del clique máximo ponderado
    - Definición de clique: Un clique es un subconjunto de vértices donde cada par de vértices está conectado por una arista. En el contexto del problema, un clique representa un grupo de personas que se conocen todas entre sí.
    - Objetivo: Identificar el clique cuya suma de pesos (aportes) sea máxima. Esto implica encontrar el grupo más rentable de personas que cumplan con la condición de conocerse mutuamente.
- Inicialización y preprocesamiento
    - Función `Inicializar`: Reduce la complejidad del problema al eliminar vértices que no pueden formar parte de la solución óptima. Esto se logra mediante la comparación de los pesos de los vértices y sus adyacencias con una cota inferior de peso.
    - Orden de vértices y subgrafo `G_prima`: Se establece un orden inicial para los vértices y se crea un subgrafo `G_prima`, que es una versión reducida del grafo original. Esto facilita las operaciones subsiguientes al trabajar con un conjunto más pequeño y manejable de vértices.
- Búsqueda Recursiva: A través de la función `WLMC` se implementa un enfoque de *Best First Search* para explorar el espacio de soluciones posibles. La función se llama recursivamente para explorar diferentes combinaciones de vértices y calcular el peso total del clique en cada caso.
- Optimización, evaluación y poda: En cada llamada recursiva, se evalúa si el clique actual puede superar el peso del mejor clique encontrado hasta ese momento. Si no es posible, se descarta esa combinación (poda), lo cual optimiza el proceso al evitar el cálculo de combinaciones no prometedoras.
- Resultado Final como un clique máximo ponderado encontrado: El algoritmo termina cuando se han explorado todas las combinaciones posibles o cuando se ha podado el espacio de búsqueda de forma suficiente. El resultado es el clique con el peso máximo encontrado, representando la solución óptima al problema.
- Aproximación: El algoritmo es exacto para casos donde el grafo es disperso, es decir cuando hay muchos más nodos que aristas dentro del grafo, en otros casos, no da la respuesta

exacta. Esto se debe a que, para lograr buenos tiempos, el algoritmo va a borrar varios nodos. Si los nodos hacen parte del clique máximo entonces estos no se van a tener en cuenta a la hora de formar el clique.

2. Gráfica



a. Cálculo del peso total del clique: Para los subconjuntos que son cliques, calculamos el peso total, que es la suma de los aportes (pesos de los nodos) de todas las personas en ese clique.
b. Identificación del clique máximo: El clique máximo es aquel con el mayor peso total. Entre todos los cliques posibles, seleccionamos el que tiene la suma más alta de aportes. Esto puede requerir comparar el peso total de varios cliques diferentes.
c. Resultados finales: Una vez identificado el clique máximo, presentamos este subconjunto como la solución al problema. Este subconjunto representa el grupo de personas que se conocen entre sí y que, en conjunto, pueden aportar la mayor cantidad de dinero.

3. Complejidad

En el peor de los casos, nuestro algoritmo predomina su complejidad en la función BuscarMaxCliquePonderado con $O(2^V)$.

- Exploración de todos los subconjuntos de vértices:
  - La función `BuscarMaxCliquePonderado` busca el clique máximo ponderado, lo que implica encontrar el subconjunto de vértices cuya suma total de pesos es máxima y todos los vértices en el subconjunto están interconectados. La búsqueda de cliques se realiza de manera recursiva, explorando diferentes combinaciones de vértices.
  - En el peor de los casos, para encontrar este subconjunto óptimo, el algoritmo necesita explorar todas las combinaciones posibles de vértices. Esto es similar a generar todos los posibles subconjuntos de un conjunto de *V* elementos. Este enfoque recursivo puede llevar a explorar todas las combinaciones posibles de los vértices, especialmente si no hay una poda efectiva o heurística que reduzca el espacio de búsqueda.
- Número de subconjuntos en un conjunto: Matemáticamente, el número de subconjuntos de un conjunto con *V* elementos es 2^V. Esto se debe a que cada elemento tiene dos opciones en cada subconjunto: estar presente o estar ausente.

Por ejemplo, para un conjunto con 3 elementos (V = 3), los subconjuntos serían: {{}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3} }, lo que suma $2^3 = 8$ subconjuntos.

Debido a esta necesidad de explorar todas las combinaciones posibles en el peor de los casos, la complejidad se convierte en exponencial, específicamente $O(2^V)$. También se refleja el hecho de

que, para cada vértice, hay una decisión binaria (incluir o excluir), y estas decisiones se multiplican a través de todos los vértices.

4. Escenarios

- 1: Aceptar no conocer máximo una persona
    - Retos
        - Flexibilidad en las conexiones: Modificar la definición de un clique válido. En el problema original, una clique es válida si todos los miembros se conocen entre sí. En este escenario, se permite que una persona no conozca a una persona dentro de la clique, lo que requiere una nueva lógica para determinar las cliques válidas.
        - Complejidad computacional: Permitir una desconexión podría aumentar significativamente la cantidad de combinaciones posibles a considerar, afectando potencialmente el rendimiento del algoritmo.
    - Cambios Necesarios
        - Modificar la función `BuscarMaxCliquePonderado` y `MaxCliquePonderadoExacto` para permitir cliques donde cada miembro conoce al menos a todos los demás miembros excepto uno.
        - Revisar la lógica de generación de subgrafos y la función `Inicializar` para considerar esta nueva definición de clique.
- 2: Conocer al menos a una persona
    - Retos
        - Definición más amplia de clique: Este escenario relaja aún más la restricción, permitiendo cliques donde cada miembro solo necesita conocer a una persona de la clique. Esto podría llevar a la formación de grupos mucho más grandes y heterogéneos.
        - Aumento de combinaciones: Al igual que en el escenario anterior, la cantidad de combinaciones posibles aumenta, lo que puede hacer que el algoritmo sea menos eficiente.
    - Cambios Necesarios
        - Modificar la lógica en `BuscarMaxCliquePonderado` y `MaxCliquePonderadoExacto` para identificar cliques válidas bajo esta nueva regla.
        - Adaptar la generación de subgrafos y la inicialización para aceptar estas nuevas cliques. Esto puede implicar un cambio significativo en la manera de procesar y evaluar las conexiones entre los vértices.

Referencias: El algoritmo fue fuertemente basado en la siguiente documentación: Jiang, H., Li, C.-M., & Manya, F. (2017). An Exact Algorithm for the Maximum Weight Clique Problem in Large Graphs. Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17).