Análisis y desarrollo de Algoritmos (ADA) Clique Problem

Fabrizio Franco
School of Computer Science
University of Engineering and Technology UTEC
Email: fabrizio.franco@utec.edu.pe

***Resumen*—En el presente trabajo, se realiza un análisis del 'Clique problem' y se abordan algunos algoritmos para este conocido problema de tipo NP-Completo del área de Ciencias de la Computación.**

***Index Terms*—NP-HARD, CLIQUE PROBLEM, NP-HARD PROBLEMS.**

I. Introducción

En el ámbito de Ciencias de la Computación, el Cliqué Problem es un problema computacional que consiste en encontrar subconjuntos de vértices en un grafo en donde todos los vértices de dicho subconjunto sean adyacentes a todos los demás vértices del subconjunto. Este subconjunto encontrado puede obedecer a ciertas restricciones, por ejemplo hallar el cliqué máximo en un grafo con aristas ponderadas o tener la mayor cantidad de vértices posible. Dentro las diferentes variantes existentes del problema, es bien sabido que la enorme mayoría corresponde a problemas NP-HARD debido a la relación con el "decision problem", el cual ya estaba probado como NP. Por lo que, por ejemplo, hallar todos los cliques máximos de un grafo requeriría una solución en tiempo exponencial. Ahora bien, en grafos de tipos específicos con características inherentes a su categoría se pueden encontrar algoritmos que resuelvan el problema mediante un algoritmo que no funciona fuera de dicha clasificación. La primera intuición sería realizar una solución por fuerza bruta, lo cual es posible, pero la complejidad crece de una forma insostenible, por lo que se han desarrollado otros algoritmos no polinomiales pero mejores comparados con un approach por fuerza bruta. Por ejemplo el algortimo Bron–Kerbosch, el cual será detallado posteriormente.

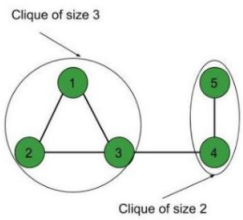

Figura 1: Ejemplo Clique - [2]

II. Definiciones

II-A. Clique

Dado un grafo G, un subconjunto K es un clique de G sí y solo sí, para cada par de vértices en K existe una arista en G que tiene dicho par de vértices como endpoints.

II-B. Clique máximo

Es un clique, en donde no puede agregarse más vértices y cumplir la condición descrita previamente. Formalmente, para cada vértice 'v' que no forma parte del clique, necesariamente debe haber un vértice 'w' $\in K$ y no sea adyacente a 'v'.

II-C. Clique number

Representa el tamaño del clique máximo, es decir, el número de vértices que conforma el clique máximo. Se denota como $\omega(G)$

Para el clique problem se tienen algunas consideraciones:

- El input es un grafo no dirigido y la output es un clique máximo en el grafo. Si hay múltiples cliques, se elige uno arbitrariamente.
- Con grafos ponderados el input es un grafo no dirigido con pesos en sus vértices y el output es una clique con el peso total máximo. El problema clique máximo es un caso particular donde todos los pesos son iguales.
- También, es posible requerir como output una lista con todos y cada uno de los cliques máximos.
- Asimismo, el output requerido también puede ser todos los cliques de tamaño x, donde x es un input adicional.
- Finalmente, en el clique decision problem, se recibe un grafo no dirigido y un número x (tamaño del clique), el output es un valor booleano que indica si el grafo contiene o no almenos un clique de tamaño x.

EL problema de independent set y clique están fuertemente relacionados, de hecho, son complementarios. Es decir, un clique del grafo original G es un independent set del grafo complementario de G. El clique problem desde un enfoque de decision problem sirve para entender el acercamiento a la familia NP-Completo.

III. NP - Completitud

La prueba de completitud NP para este problema es mediante una reducción en tiempo polinomial desde el problema de satisfacibilidad booleana. Es decir, traducir fórmulas booleanas

en forma conjuntiva normal (CNF) en instancias equivalentes del problema de clique máximo. A partir de una fórmula CNF dada, la demostración original toma como notación que v es una variable y su negación $!y$. Entonces, se tiene una cláusula en la fórmula que contiene v. Dos de estos vértices están conectados por un borde si representan asignaciones de variables compatibles para diferentes cláusulas. Si k denota el número de cláusulas en la fórmula CNF, entonces los cliques de k-vértices en este grafo representan formas de asignar valores de verdad. Por lo tanto, la fórmula es satisfactoria si y solo si existe un clique del grafo.

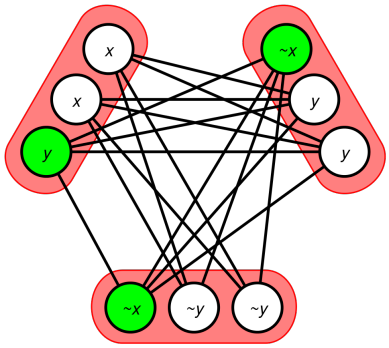

Figura 2: NP - SAT reducción - [7]

IV. Aplicaciones

Los cliques de un grafo tienen diversas aplicaciones prácticas en muchos campos que utilizan Ciencias de Computación como herramienta, así como también campos propios de Ciencias la computación. Entre los ejemplos más notables destacan:

- Química: Para encontrar productos químicos que coincidan con una estructura química específica; modelar el acoplamiento molecular y los sitios de unión de reacciones químicas. También se pueden utilizar para encontrar estructuras similares dentro de diferentes moléculas. Se forma un grafo donde cada vértice representa un par de átomos emparejados, uno de cada una de las dos moléculas. Dos vértices están conectados por un borde si las coincidencias que representan son compatibles entre si.
- Bio-informática: Los algoritmos de clique pueden ser utilizados para inferir árboles evolutivos, predecir estructuras de proteínas, y encontrar grupos de proteínas que interactúan entre ellas. Enumerar los cliques en un grafo de dependencia es un paso importante en el análisis de ciertos procesos aleatorios en el campo de la bio-informática.
- Redes: Dada una red social, donde los vértices del grafo representan personas y las aristas del grafo son relaciones de amistad. Un clique representaría subconjunto de personas que se conocen entre sí, y de esta forma se puede encontrar grupos de amigos en común dado un grupo de personas más grande.

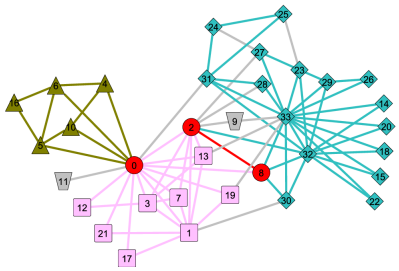

Figura 3: Ejemplol Red Social - Clique - [1]

V. Algoritmos

V-A. *Hallar Clique Máximo - Greedy Approach*

- Se parte de un clique vacio
- Para cada vértice v que examina este bucle, agregue v al clique si está adyacente a cada vértice que ya está en el clique, y descarte v en caso contrario.

Este algoritmo se ejecuta en tiempo lineal

V-B. *Hallar Clique de X vértices*

Para determinar si el grafo G contiene un clique de k-vértices es posible hacerlo mediante un algoritmo de fuerza bruta o hacerle cierta mejora. Este algoritmo examina cada subgrafo con k vértices y comprueba si forma un clique. Se necesita tiempo $O(n^k \cdot k^2)$. Esto se debe a que hay n^k subgrafos para verificar, cada uno de los cuales tiene un total de k^2) edges. Por tanto, el problema puede resolverse en tiempo polinomial siempre que k sea una constante. Sin embargo, cuando k no tiene un valor fijo, sino que puede variar como parte de la entrada al problema, el tiempo es exponencial

V-C. *Listar todos los cliques máximos*

Dada la investigación de [**?**], un grafo de n vértices tiene como máximo $3^{n/3}$ máximos cliques. Todos y cado uno de dichos cliques pueden ser retornados por el algoritmo de Bron-Kerbosch, el cual a través de backtracking recursivo obtiene su resultado. La principal subrutina recursiva de este procedimiento tiene tres argumentos: una cliqué local, un conjunto de vértices candidatos que podrían agregarse al clique y otro conjunto de vértices que no deberían agregarse. El algoritmo intenta agregar los vértices candidatos uno por uno al clique parcial, haciendo una llamada recursiva para cada uno.

```
algorithm BronKerbosch1(R, P, X) is
    if P and X are both empty then
        report R as a maximal clique
    for each vertex v in P do
        BronKerbosch1(R UNION {v}, P INTERSECT N(v),
     X INTESECT N(v))
        P := P \ {v}
        X := X UNION {v}
```

Listing 1: BronKerbosch sin pivote [6]

Se tiene tres conjuntos disjuntos de vértices R , P , y X, se encuentra el cliqué máximo que incluyen todos los vértices en R, algunos de los vértices en P , y ningún vértice en X . Siempre, P y X son conjuntos disjuntos cuya unión consiste en aquellos vértices que forman cliques cuando se añaden al conjunto R . En otras palabras, P U X es el conjunto de vértices que están unidas a cada elemento de R . Cuando P y X están vacíos, no hay más elementos que se puedan agregar a R , por lo que R es un cliqué máximo y el algoritmo genera R.

```
BronKerbosch2(R,P,X):
     if P and X are both empty:
         report R as a maximal clique
     choose a pivot vertex u in P UNION X
     for each vertex v in P \ N(u):
         BronKerbosch2(R UNION {v}, P INTERSECT N(v)
    , X INTERSECT N(v))
         P := P \ {v}
         X := X UNION {v}
```

Listing 2: BronKerbosch con pivote [6]

Es ineficaz en el caso de grafos con muchos cliques no máximas, realiza una llamada recursiva para cada camarilla, máxima o no. Para ahorrar tiempo y permitir que el algoritmo retroceda más rápidamente en las ramas de la búsqueda que no contienen camarillas máximas, Bron y Kerbosch introdujeron una variante del algoritmo que involucra un ”vértice pivote”. Cualquier clique máximo debe incluir a u o uno de sus no vecinos, ya que de lo contrario el clique podría aumentarse añadiéndole “u”. Por lo tanto, solo es necesario probar “u” y sus no vecinos como opciones para el vértice “v” que se agrega a R en cada llamada recursiva al algoritmo.

```
BronKerbosch3(G):
    P = V(G)
    R = X = empty
    for each vertex v in a degeneracy ordering of G:
        BronKerbosch2({v}, P INTERSECT N(v), X
    INTERSECT N(v))
        P := P \ {v}
        X := X UNION {v}
```

Listing 3: BronKerbosch vertex ordering pivote [6]

Un método alternativo para mejorar la forma básica del algoritmo de Bron-Kerbosch implica renunciar a pivotar en el nivel más externo de recursividad y, en su lugar, elegir el orden de las llamadas recursivas con cuidado para minimizar los tamaños de los conjuntos P de vértices candidatos dentro de cada llamada recursiva. Para mejorar aún más el tiempo de ejecución del algoritmo, introducimos el orden de vértices en el nivel más externo de la recursividad, lo que optimiza el tamaño general del conjunto P, por lo que el uso de la estrategia de pivote y el orden de vértices da el peor tiempo de ejecución de $O(3^{n/3})$. Esto es óptimo en función de n, ya que existen hasta $3^{n/3}$ camarillas máximas en un gráfico de n vértices.

```
def find_cliques(potential_clique=[],
    remaining_nodes=[], skip_nodes=[], depth=0):

    if len(remaining_nodes) == 0 and len(skip_nodes)
     == 0:
        print('This is a clique:', potential_clique)
        return 1

    found_cliques = 0
    for node in remaining_nodes:

        new_potential_clique = potential_clique + [
    node]
        new_remaining_nodes = [n for n in
    remaining_nodes if n in node.neighbors]
        new_skip_list = [n for n in skip_nodes if n
    in node.neighbors]
        found_cliques += find_cliques(
    new_potential_clique, new_remaining_nodes,
    new_skip_list, depth + 1)

        remaining_nodes.remove(node)
        skip_nodes.append(node)
    return found_cliques
```

Listing 4: Low Level Implementation [6]

Dado el siguiente grafo:

```
A.neighbors = [B, C, E]
B.neighbors = [A, C, D, F]
C.neighbors = [A, B, D, F]
D.neighbors = [C, B, E, F]
E.neighbors = [A, D]
F.neighbors = [B, C, D]

all_nodes = [A, B, C, D, E, F]
```

Figura 4: Ejemplo Clique - [5]

Se obtiene el siguiente resultado:

```
This is a clique: [A, B, C]
This is a clique: [A, E]
This is a clique: [C, B, D, F]
This is a clique: [E, D]
Total cliques found: 4
```

Figura 5: Ejemplo Clique - [5]

REFERENCIAS

[1] Sacristán, V. (n.d.). The enumeration of maximal cliques of large graphss.

[2] Sacristán, V. (n.d.). Subestructura y búsqueda de subestructura común máxima.

[3] Quach, T. (

[4] 2012). ”Heuristics for maximum clique and independent set. Barr, V., Siegelmann, H., Sarkozy, G., Horn, M., & Weber, J. (2004). Computational Geometry Lecture

[5] Bowyer, A. (1981) Listing all maximal cliques in large sparse real-world graphs. https://doi.org/0.1093/comjnl/24.2.162

[6] Watson, D. F. (1981). Coloring Perfect Graphs

[7] de Berg, M., Cheong, O., van Kreveld, M., & Overmars, M. (2008). Computational Geometry: Algorithms and Applications. Springer-Verlag Berlín Heidelberg.

[8] Fortune, S. (1986). .An introduction to chordal graphs and clique trees”, 313–322.