Diseño e implementación de método de compresión de grafos basados en clustering de cliques maximales

Defensa de Tesis

Felipe A. Glaría Grego

Lilian Salinas, Ceciclia Hernández
Departamento de Ingeniería Informática y Ciencias de la Computación
Facultad de Ingeniería
Universidad de Concepción

27 de junio de 2019

Motivación

Gran crecimiento en grafos de redes sociales y de la web.

- Estimación sitios indexados: 4,46 mil millones[1].
- Usuarios activos diarios en Facebook: 1,47 mil millones. Crecimiento de 11 % anual[2].

Los grafos de redes sociales son menos comprimibles. Muy útil descubir usuarios reelevantes y cómo se relacionan, para mejorar resultados de búsquedas, recomendaciones, marketing, entre otros.

Alto costo en recursos que demanda su procesamiento.

- Principalmente espacio en memoria.
- Jerarquía de memoria penaliza tiempo de acceso a datos alejados de unidades de procesamiento.

---

[1] `http://www.worldwidewebsize.com/`, consultado el 05 de octubre del 2018.

[2] `https://investor.fb.com`, informe de resultados del segundo trimestre del 2018.

Trabajos Relacionados

WebGraph, *Boldi* y *Vigna* (2003).
k2-tree, *Brisaboa*, *Ladra* y *Navarro* (2009).
Graph Compression by BFS, *Apostolico* y *Drovandi* (2009).
Layered Label Propagation, de *Boldi*, *Vigna*, *Santini* y *Rosa* (2011).
Tight and simple Web graph compression, *Grabowski* y *Bieniecki* (2011).
Compressed representations for web and social graphs, *Hernández* y *Navarro* (2014).
Compressing graphs by grammars, *Maneth* y *Peternek* (2016).

Marco teórico

- Un **grafo** $G = (V, E)$ como el conjunto finito de *vértices* V (nodos) y el conjunto de *aristas* $E \subseteq V \times V$ (arcos).
- Dos vértices v_1 y $v_2 \in V(G)$ son **adyacentes** o **vecinos** si $(v_1, v_2) \in E(G)$ y $v_1 \neq v_2$.
- Un grafo es **no dirigido** cuando la arista conlleva ambos sentidos, es decir $(v_1, v_2) = (v_2, v_1)$.
- El **grado de un vértice** $d(v)$ es la cantidad de vértices en $V(G)$ que son adyacentes con v.
- La **matriz de adyacencia** de un grafo G corresponde a una matriz binaria cuadrada $|V(G)| \times |V(G)|$ donde cada bit representa si un par de vértices v_1 y $v_2 \in V(G)$ son vecinos o no.

Marco teórico (2)

- Un grafo **k-degenerate** es no dirigido, donde cada subgrafo tiene un vértice con grado a lo más **k**.
- El índice de **degeneracy** de un grafo, $D(G)$, es el menor valor **k** para el cual el grafo es **k-degenerate**.
- Un **clique** es un subgrafo donde todos los vértices son adyacentes entre sí. Un **clique maximal** no es subconjunto de otro clique más grande.

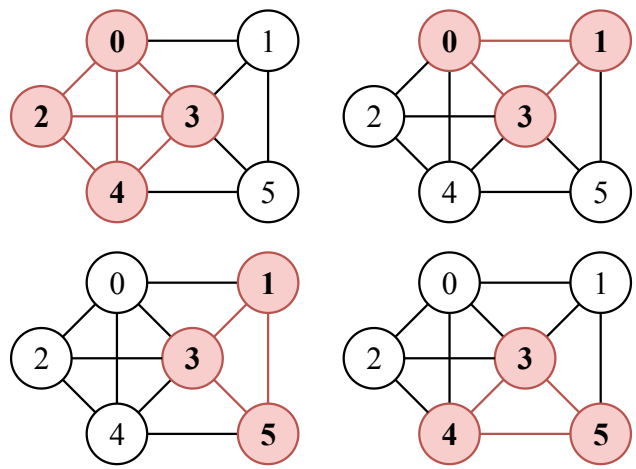

Figura 1: Ejemplo de grafo y sus cliques maximales.

Marco teórico (3)

- Un **triángulo** es un subgrafo de tres vértices y tres aristas. Se define $\lambda(v)$ como la cantidad de triángulos donde participa un nodo v.
- Se define $\lambda(G)$ como la cantidad de triángulos de un grafo. Se calcula sumando $\lambda(v)$ para cada vértice v, y dividiendo el total en tres.

$$\lambda(G) = \frac{1}{3} \sum_{v \in V} \lambda(v) \tag{1}$$

- Un **triplete** es un subgrafo de tres vértices y dos aristas, donde las aristas comparten un vértice común. Se define $\tau(v)$ como la cantidad de tripletes donde v es el vértice común.
- Se define $\tau(G)$ como la cantidad de tripletes de un grafo.

$$\tau(G) = \sum_{v \in V} \tau(v) \tag{2}$$

Marco teórico (4)

- El **coeficiente de clusterización** de un vértice indica cuánto está conectado con sus vecinos, y se define como $c(v) = \lambda(v)/\tau(v)$.
- El coeficiente de clusterización de un grafo ($C(G)$) es el promedio del coeficiente de todos los nodos del grafo.

$$C(G) = \frac{1}{|V'|} \sum_{v \in V'} c(v) \tag{3}$$

con $V' = \{v \in V | d(v) \geq 2\}$.

- La **transitividad** de un grafo ($T(G)$) es la probabilidad que un par de nodos adyacentes estén interconectados.

$$T(G) = \frac{3\lambda(G)}{\tau(G)} \tag{4}$$

Método de Compresión Propuesto

Consta de tres etapas:

1. Listar todos los cliques maximales del grafo.
2. Particionar listado de cliques, utilizando heurística eficiente que explote su superposición.
3. Comprimir particiones en estructura compacta.

Etapa 1: Cliques Maximales

Es un problema complejo desde un punto de vista teórico y práctico.
Eppstein, Strash et. al. proponen un algoritmo[3] rápido para listar cliques maximales de grafos dispersos.
Se asume que algoritmo puede entregar el listado de cliques maximales de un grafo.
Problema a resolver: Encontrar un método eficiente para dividir en particiones el grafo de cliques.

[3] https://github.com/darrenstrash/quick-cliques

Etapa 1: Cliques Maximales (2)

Definición

Gafo de cliques

Dado un grafo $G = (V, E)$ *y* $\mathcal{C} = \{c_1, c_2, ..., c_N\}$ *el conjunto de tamaño* N *de cliques maximales que cubren* G*, se tiene* $CG_{\mathcal{C}} = (V_{\mathcal{C}}, E_{\mathcal{C}})$ *un grafo de cliques donde:*

1. $V_{\mathcal{C}} = \mathcal{C}$
2. $\forall c, c' \in \mathcal{C}, (c, c') \in E_{\mathcal{C}} \iff c \cap c' \neq \varnothing$

Etapa 1: Cliques Maximales (3)

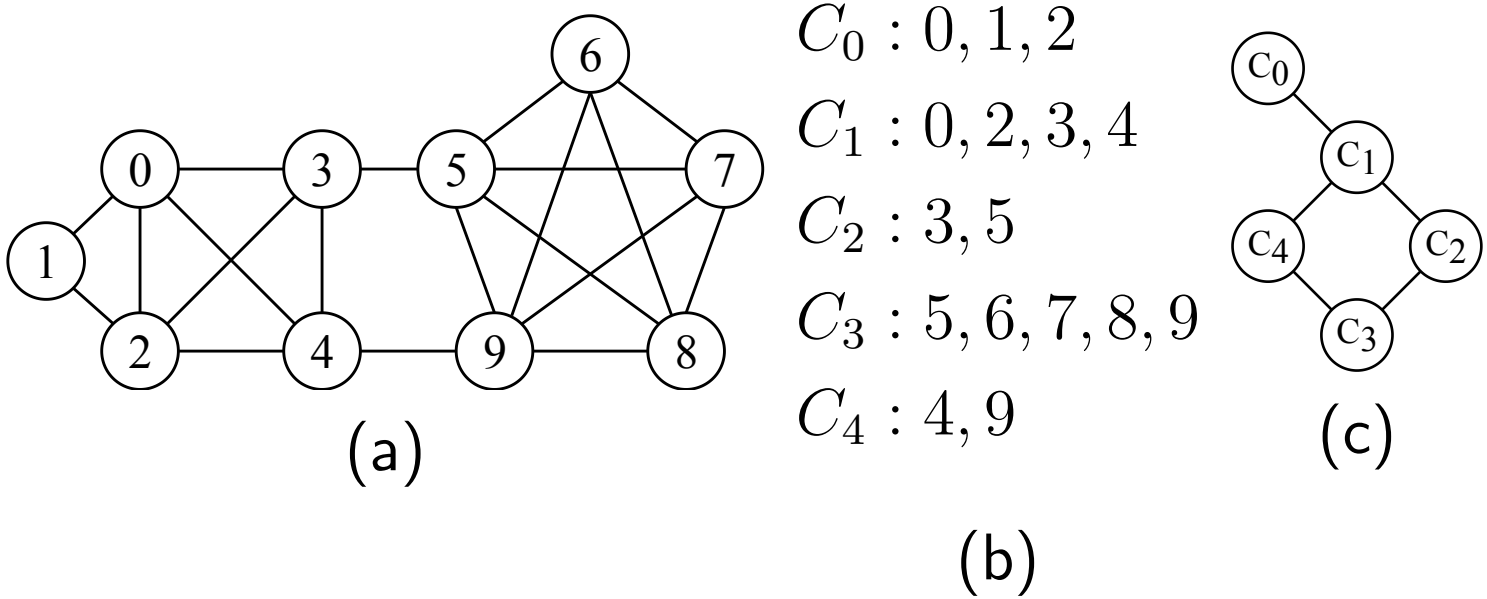

Figura 2: (a) Grafo no dirigido. (b) Lista de cliques maximales. (c) Grafo de cliques.

Etapa 2: Particionar listado de cliques

Problema

Encontrar particiones de cliques para el grafo de cliques $CG_{\mathcal{C}}$. *Dado un grafo de cliques* $CG_{\mathcal{C}} = (V_{\mathcal{C}}, E_{\mathcal{C}})$, *encontrar un set de particiones de cliques* $\mathcal{CP} = \{cp_1, cp_2, ..., cp_M\}$ *de* $CG_{\mathcal{C}}(V_{\mathcal{C}}, E_{\mathcal{C}})$ *con* $M \geq 1$, *tal que*

1. $\bigcup_{i=1}^{M} cp_i = CG_i$
2. $cp_i \cap cp_j = \varnothing$ *para* $i \neq j$
3. *cualquier* $cp_i \in \mathcal{CP}$ *es un subgrafo de* $CG_{\mathcal{C}}(V_{\mathcal{C}}, E_{\mathcal{C}})$ *inducido por el subset de vértices en* cp_i

Etapa 2: Particionar listado de cliques (2)

Definir una heurística que permita agruparlos en particiones eficientes, que exploten dicha redundancia de vértices en los cliques maximales.

Definición

Función de ranking
Dado un grafo $G = (V, E)$ *y* $\mathcal{C} = \{c_1, c_2, ..., c_N\}$ *el conjunto de tamaño* N *de cliques maximales que cubren* G*, una función de ranking es una función* $r : V \to \mathbb{R}^+$ *que retorna un valor de puntuación para cada vértice* $v \in V$*.*

Muchas gracias