Tarea 4

JUAN FRANCISCO GORTAREZ RICARDEZ, Tecnológico de Monterrey

Este documento tiene como propósito la demonstración de los pasos seguidos para obtener gráficos en múltiples formatos usando la librería de SNAP y Gephi.

Additional Key Words and Phrases: Gephi, GraphML, SNAP Library, Graph Algorithms

ACM Reference Format:
Juan Francisco Gortarez Ricardez. 2019. Tarea 4. 1, 1 (October 2019), 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 PASO A PASO

En esta sección, se describirá a detalle el procedimiento que se siguió para exportar un grafo a múltiples formatos usando la librería de SNAP y Gephi. En este punto, se asume que uno ya descargó SNAP, Gephi, y todas sus dependencias, y que se encuentra en el directorio de instalación de SNAP.

1.1 Descargar Dataset

Lo primero que se tiene que realizar es la descarga de el Dataset a analizar. Los utilizados para esta tarea se encuentran en https://snap.stanford.edu/data/index.html. Una vez descargado, el archivo se debe ubicar en el directorio de SNAP, en el mismo nivel que los otros archivos.

1.2 Importar Dataset

Usando la librería de SNAP para C++, el proceso para importar el dataset es muy simple; Solamente se tiene que añadir snap.h en los includes del archivo, y utilizar las siguientes funciones:

```
typedef PNGraph DGraph;
DGraph dg = TSnap::LoadEdgeList<DGraph>("<filename>",0,1);
```

una vez realizado esto, se puede proceder a la exportación en múltiples formatos.

1.3 Exportar Grafos

Las implementaciones de la exportación de grafos son diferentes para cada formato, pero todas tienen como salida el archivo "wiki.<TYPE>" donde TYPE es el formato a exportar. Esta implementación exporta en los siguientes formatos:

- GraphML
- Gexf
- GDF
- JSON

Author's address: Juan Francisco Gortarez Ricardez, Tecnológico de Monterrey, a01021926@itesm.mx.

XXXX-XXXX/2019/10-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.4 Cargar Grafos a Gephi

Esta implementación descargó Gephi en Ubuntu, lo cual no genera un shortcut al ejecutable. Para correr Gephi en Ubuntu, se tiene que ubicar en el directorio /bin/ del archivo descargado, y luego correr Gephi con ./gephi.
El dataset utilizado (Wiki-vote) se ve de la siguiente manera:

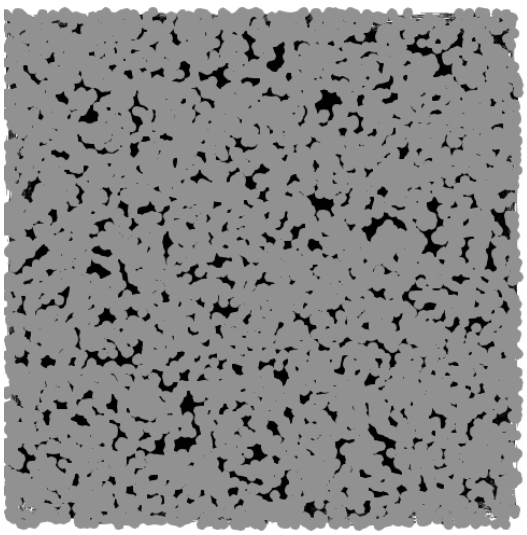

Fig. 1. Gráfico basado en el dataset Wiki-Vote, visto en Gephi

Gephi tiene múltiples opciones para la visualización de grafos, como es el caso de mapas de calor (para medir concentración de aristas en un área), el camino más corto, y coloreado de vértices y aristas en base a su peso. A continuación se muestra la visualización de algunas de esas opciones:

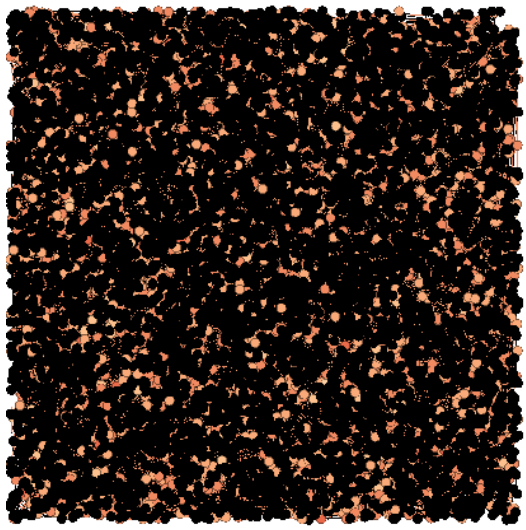

Fig. 2. Gráfico utilizando la funcionalidad de mapa de calor. Puntos más naranjas indican mayor concentración

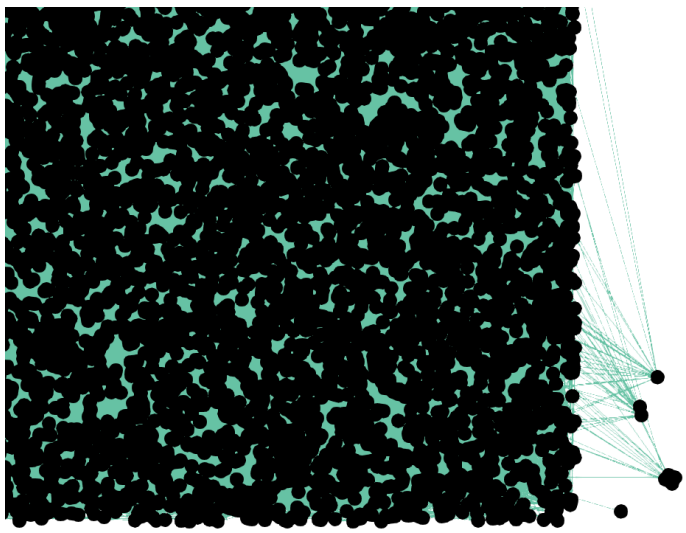

Fig. 3. Gráfico utilizando la funcionalidad de coloración por peso, con algunos nodos expandidos.Se puede observar que el peso no cambia mucho entre aristas

2 VENTAJAS/DESVENTAJAS

- **Gephi:** Gephi permite una representación intuitiva y rápida de un grafo, y permite ver agrupaciones y tendencias a gran escala. Además, soporta múltiples formatos de grafos, y hace rápido al análisis de tendencias. Sin embargo, no está libre de desventajas: es muy ineficiente en recursos, y en sistemas con poca memoria de video, la representación alenta al programa a tal escala que este es inutilizable. Además, no contiene herramientas de análisis numérico exacto, lo cual obstaculiza su uso en aplicaciones de ingeniería. [2]

- **GraphML:** Utiliza esquemas XML, permite la coloración de nodos, y permite al usuario determinar atributos personalizados de los elementos. Sin embargo, no permite el uso de una matriz de adyacencia. [3]
- **GEXF:** Es el formato recomendado por Gephi, y contiene todas las funcionalidades de otros formatos (con excepción de soporte para una lista de adyacencia). Contiene datos para el reconocimiento de forma, tamaño y posición de nodos. [4]
- **GDF:** No utiliza esquemas XML ni una jerarquía, por lo que no puede ser visualizado de la mejor manera, sin embargo, es legible para un humano al ser ordenada en formato de texto como .csv.
- **JSON:** JSON no es soportado por Gephi, por lo cual se tendrían que desargar aplicaciones adicionales o modificaciones, pero utiliza un esquema JSON como análogo del esquema XML de otros métodos para validación, y permite la visualización de posición, peso, color, entre otros. [1] [3]

3 COMPLEJIDAD Y TIEMPOS

Los métodos de exportación son muy similares, teniendo complejidad espacial constante, dado que no se utilizan estructuras adicionales a una lista de nodos, la cual es iterada con un apuntador para representar o exportar los datos. De igual manera, la complejidad temporal es igual para todos los métodos, ya que estos utilizan dos ciclos no anidados, uno dependiente de vértices y uno de aristas.Por tanto, la complejidad sería

```
O(V+E)
```

Los tiempos de ejecución de cada algoritmo se muestran a continuación:

- GraphML: 175 ms.
- JSON: 125 ms.
- GEXF: 159 ms.
- GDF: 117 ms.

Se puede apreciar que existe una diferencia considerable entre el método más rápido (GDF) y el más lento (GraphML). Se realizaron múltiples iteraciones, y GDF siempre era el formato que más rápido se exportaba, seguido de JSON.

4 CÓDIGOS

El código que se muestra a continuación puede ser consultado en la siguiente liga: https://github.com/tec-csf/tc2017-t4-otono-2019-Starfleet-Command , pero adicionalmente se incluyen en el directorio de este archivo.

REFERENCES

[1] Anthony Bargnesi. 2019. JSON Graph Format (JGF). http://jsongraphformat.info/. Accessed:26/10/2019.
[2] Gephi. 2018. Supported Graph Formats. https://gephi.org/users/supported-graph-formats/. Accessed:26/10/2019.
[3] GraphML Working Group. 2007. GraphML Specification. http://graphml.graphdrawing.org/specification.html. Accessed:26/10/2019.
[4] Gexf Working Group. 2018. GEXF File Format. https://gephi.org/gexf/format/. Accessed:26/10/2019.