Trabajo Práctico 3: Comunidades NP-Completas

El presente trabajo busca evaluar el desarrollo y análisis de un algoritmo de Backtracking para resolver un Problema NP-Completo, así como el análisis de posibles aproximaciones. La fecha de entrega del mismo es el 16/06.

Introducción

Luego de los grandes resultados obtenidos en los trabajos anteriores, fuimos nuevamente ascendidos. Ahora somos la mano derecha del Gringo (que dejó de decirnos cada 3 palabras cosas de desayunar con peces).

En función de poder evitar que situaciones así vuelvan a suceder, obtuvo la información de qué integrante de la organización es cercano a otro. Con esta información construyó un grafo no dirigido y no pesado (vértices: personas, aristas: si las personas son cercanas). Nos pidió que separemos el grafo en comunidades. Es decir, separar los vértices en K clusters (grupos).

Todo venía bien, nuestra primera idea era la de maximizar la distancia mínima entre comunidades, lo cual puede resolverse con un algoritmo de árboles de tendido mínimo (y siendo este grafo no pesado, aún más fácil), pero Amarilla Pérez nos indicó que esa métrica da muy malos resultados. Que en vez de esto, mejor minimizar la distancia máxima entre dos vértices dentro de la misma comunidad.

Pasamos de un un problema que hasta se podía resolver de forma lineal, a un problema que, creemos, es NP-Completo. Debemos convencer a Amarilla Pérez de que esta no es una buena idea, que demorará mucho. Para esto, igualmente, vamos a tener que venir con demostraciones, mediciones, y al menos una alternativa...

Consigna

Para los primeros dos puntos considerar la versión de decisión del problema de *Clustering por bajo diámetro*: Dado un grafo no dirigido y no pesado, un número entero k y un valor C, ¿es posible separar los vértices en a lo sumo k grupos/clusters disjuntos, de tal forma que todo vértice pertenezca a un cluster, y que la distancia máxima dentro de cada cluster sea a lo sumo C? (Si un cluster queda vacío o con un único elemento, considerar la distancia máxima como 0).

Al calcular las distancias se tienen en cuenta tanto las aristas entre vértices dentro del cluster, como cualquier otra arista dentro del grafo.

1. Demostrar que el Problema de Clustering por bajo diámetro se encuentra en NP.
2. Demostrar que el Problema de Clustering por bajo diámetro es, en efecto, un problema NP-Completo. Si se hace una reducción involucrando un problema no visto en clase, agregar una (al menos resumida) demostración que dicho problema es NP-Completo.
3. Escribir un algoritmo que, por backtracking, obtenga la solución óptima al problema (valga la redundancia) en la versión de optimización: Dado un grafo no dirigido y no pesado, y un valor k, determinar los k clusters para que la distancia máxima de cada cluster sea mínima. Para esto, considerar minimizar el máximo de las distancias máximas (es decir, de las distancias máximas de cada cluster, nos quedamos con la mayor, y ese valor es el que queremos minimizar).

   Generar sets de datos para corroborar su correctitud, así como tomar mediciones de tiempos.
4. Escribir un modelo de programación lineal que resuelva el problema de forma óptima. Recordar que el cálculo de las distancias mínimas se puede hacer previamente, y no ser parte del modelo de programación lineal (problema muy sencillo de resolver con un algoritmo BFS, pero no así con programación lineal).

   Implementar dicho modelo, y ejecutarlo para los mismos sets de datos para corroborar su correctitud. Tomar mediciones de tiempos y compararlas con las del algoritmo que implementa Backtracking.
5. Ya pudiendo mostrar que no es una buena idea trabajar con estas métricas para obtener las comunidades de una organización tan grande, la agente T nos recomendó analizar la posibilidad de no clusterizar por bajo diámetro, sino maximizando la Modularidad.

   > ¿La qué?

   La modularización es una métrica para definir la densidad de aristas dentro de una comunidad. En una comunidad real, hay un alto coeficiente de clustering. Básicamente, hay muchos triángulos; es decir, amigos en común. Esto se manifiesta en tener que una comunidad tiene necesariamente muchas aristas dentro de la misma, y al mismo tiempo pocas *hacia afuera*. Podemos definir la modularidad como:

   $$Q = \frac{1}{2m} \sum_i \sum_j \left(peso(v_i, v_j) - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

   Donde:

   - $peso(v_i, v_j)$: el peso de la arista entre i y j (0 si no están unidos, y en nuestro caso 1 sí si lo están).

- k_i: es la suma de las aristas del vértice i (en nuestro caso, su grado).
- m: es la suma de todos los pesos de las aristas.
- c_i: es la comunidad del vértice i.
- δ: función delta de Kronecker, que es básicamente 1 si ambas comunidades son iguales, 0 si son diferentes.

¿El problema? Maximizar la modularización es también un problema NP-Completo. ¿Lo bueno? Es conocido un algoritmo greedy que funciona muy bien para esto: El Algoritmo de Louvain. Obviamos transcribir la descripción del algoritmo, que pueden leer allí, pero la agente T logró obtener una grabación de una clase de la facultad de ingeniería donde explicaban este algoritmo. Por supuesto, si les interesa el tema pueden revisar el video entero. Dejamos también un link a las diapositivas de dicha clase.

Implementar el algoritmo de Louvain para obtener una separación en K clusters. Realizar mediciones para determinar una cota empírica de aproximación al utilizar dicho algoritmo para aproximar al problema de Clustering por bajo diámetro. Realizar esto con datos generados por ustedes, incluyendo potencialmente set de datos de volúmenes inmanejable para los algoritmos antes implementados. Recomendamos leer la explicación de la complejidad del algoritmo, la cual no es sencilla (y, por lo tanto, no la pedimos aquí tampoco).

6. **Opcional**: Implementar alguna otra aproximación (o algoritmo greedy) que les parezca de interés. Comparar sus resultados con los dados por la aproximación del punto anterior. Indicar y justificar su complejidad, en lo posible. No es obligatorio hacer este punto para aprobar el trabajo práctico (pero si resta un punto no hacerlo).

7. Agregar cualquier conclusión que parezca relevante.

Entrega

Debe enviarse al corrector asignado, por mail o slack, el link al repositorio donde se encuentre el código fuente, y donde debe encontrarse el informe en formato PDF, que debe seguir los lineamientos establecidos en el TP1 y TP2. En este caso, la forma de ejecutar el programa debe ser:

```
./tp3 ruta/a/grafo.csv <K>
```

O bien:

```
python3 tp3.py ruta/a/grafo.txt <K>
```

Por ejemplo:

```
python3 tp3.py 15-3.csv 3
```

La nota del trabajo práctico tendrá en cuenta tanto la presentación y calidad de lo presentado, como también el desarrollo del trabajo. No será lo mismo un trabajo realizado con lo mínimo indispensable, que uno bien presentado, analizado, y probado con diferentes volúmenes, set de datos, o estrategias de generación de sets, en el caso que corresponda.