Trabajo práctico 3

Fecha de entrega: domingo 25 de junio, hasta las 23:59 horas.
Fecha de recuperación: domingo 16 de julio, hasta las 23:59 horas.

Dado un grafo simple $G = (V, E)$, un subconjunto de vértices de G es una *clique* si y sólo si éste induce un subgrafo completo de G. Es decir, $K \subseteq V$, tal que $K \neq \emptyset$, es una *clique* de G si y sólo si para todo par de vértices $u, v \in K, u \neq v$, existe la arista uv en E. Definimos la *frontera* de una clique K como el conjunto de aristas de G que tienen un extremo en K y otro en $V \setminus K$. Formalmente, la *frontera* de una clique K queda definida por

$$\delta(K) = \{vw \in E \ / \ v \in K \wedge w \in V \setminus K\}.$$

Dado un grafo G, el problema de *clique de máxima frontera* (CMF) en G consiste en hallar una clique K de G cuya frontera $\delta(K)$ tenga cardinalidad máxima.

En el presente trabajo práctico se pide:

1. Describir situaciones de la vida real que puedan modelarse utilizando CMF.
2. Diseñar e implementar un algoritmo exacto para CMF y desarrollar los siguientes puntos:
   a) Explicar detalladamente el algoritmo implementado.
   b) Calcular el orden de complejidad temporal de peor caso del algoritmo utilizando el modelo uniforme.
   c) Realizar una experimentación que permita observar la performance del algoritmo en términos de tiempo de ejecución en función del tamaño de entrada. Presentar los resultados obtenidos mediante gráficos adecuados.
3. Diseñar e implementar para CMF los siguientes:
   - al menos una heurística constructiva golosa,
   - al menos una heurística de búsqueda local, y
   - al menos un algoritmo que use una metaheurística (puede ser Búsqueda Tabú [1, 2] o GRASP [3]).

   Para los métodos implementados, desarrollar los siguientes puntos:
   a) Explicar detalladamente el algoritmo implementado.
   b) Calcular el orden de complejidad temporal de peor caso del algoritmo.
   c) Describir (si es posible) instancias de CMF para las cuales el método no proporciona una solución óptima. Indicar (si es posible) qué tan mala puede ser la solución obtenida respecto de la solución óptima.
   d) Realizar una experimentación que permita observar la performance del algoritmo comparando la calidad de las soluciones obtenidas y los tiempos de ejecución en función de la entrada (y de otros parámetros de ser apropiado). Dentro de los casos de prueba se deben incluir también, como casos patológicos, aquellos descriptos en el item 3*c*. En caso de que el algoritmo tenga algún parámetro configurable que determine su comportamiento (la metaheurística por ejemplo, aunque queda abierto a los demás también), se debe experimentar variando los valores de los parámetros y elegir, si es posible, la configuración que mejores resultados provea para el grupo de instancias utilizado. Presentar los resultados obtenidos mediante gráficos adecuados.
4. Una vez elegidos los mejores valores de configuración para cada heurística implementada en 3, realizar una experimentación **sobre un conjunto nuevo de instancias** para observar la performance de los métodos comparando nuevamente la calidad de las soluciones obtenidas y los tiempos de ejecución en función del tamaño de entrada. Para los casos que sea posible, comparar también los resultados del algoritmo exacto implementado en 2. Presentar todos los resultados obtenidos mediante gráficos adecuados y discutir al respecto de los mismos.

Condiciones de entrega y términos de aprobación

Este trabajo práctico consta de varias partes las cuales pueden separarse de la siguiente manera:

I. Descripción de situciones reales (del item 1).

II. Algoritmo exacto (del item 2).

III. Heurística constructiva golosa (del item 3).

IV. Heurística de búsqueda local (del item 3).

V. Metaheurística (del item 3).

VI. Experimentación general (del item 4).

Para aprobar el trabajo se requiere aprobar todas las partes del mismo. De ser necesario (o si el grupo lo desea) el trabajo podrá reentregarse una vez corregido por los docentes y en ese caso la reentrega deberá estar acompañada por un *informe de modificaciones*. Este informe deberá detallar brevemente las diferencias entre las dos entregas, especificando los cambios, agregados y/o partes eliminadas del trabajo. Cualquier cambio que no se encuentre en dicho informe podrá no ser tenido en cuenta en la corrección de la reentrega.

Respecto de las implementaciones, se acepta cualquier lenguaje que permita el cálculo de complejidades según la forma vista en la materia. Además, debe poder compilarse y ejecutarse correctamente en las máquinas de los laboratorios del Departamento de Computación. La cátedra recomienda el uso de C++ o Java, y se sugiere consultar con los docentes la elección de otros lenguajes para la implementación.

Deberá entregarse un informe impreso que desarrolle los puntos mencionados. Por otro lado, deberá entregarse el mismo informe en formato digital acompañado de los códigos fuentes desarrollados e instrucciones de compilación, de ser necesarias. Estos archivos deberán enviarse a la dirección `algo3.dc@gmail.com` con el asunto *"TP 3: Apellido_1, ..., Apellido_n"*, donde n es la cantidad de integrantes del grupo y *Apellido_i* es el apellido del i-ésimo integrante.

La entrada y salida de los programas **deberá hacerse por medio de la entrada y salida estándar del sistema**. No se deben considerar los tiempos de lectura/escritura al medir los tiempos de ejecución de los programas implementados.

Formato de entrada: Cada instancia representa un grafo G y comienza con una línea con dos valores enteros n y m separados por espacios. Estos valores indican la cantidad de vértices y aristas de G, respectivamente. A continuación, le siguen m líneas, cada una determinando una arista del grafo. Cada una de estas líneas tiene el formato:

```
v1 v2
```

donde `v1` y `v2` son los extremos de la arista representada (numerados de 1 a n). Se puede suponer que los grafos son simples (i.e., no tienen bucles ni ejes repetidos).

Formato de salida: La salida debe contener una línea con el siguiente formato:

```
F k v1 v2 ... vk
```

donde `F` es el cardinal de la frontera de la clique dada como solución, `k` es el tamaño de la misma y `v1`, ..., `vk` son los vértices que conforman la clique (en cualquier orden). Para el algoritmo exacto, en caso de haber más de una clique de frontera máxima, el algoritmo puede devolver cualquiera de ellas.

Referencias

[1] Fred Glover. *Tabu Search - Part 1*. ORSA Journal on Computing **1** (3), pp: 190–206, 1989.

[2] Fred Glover. *Tabu Search - Part 2*. ORSA Journal on Computing **2** (1), pp: 4–32, 1990.

[3] Thomas A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization* 6, pp 109–134, 1995.