Trabajo práctico 3 (versión 1.1)

Fecha de entrega: viernes 27 de junio, hasta las 20:00 horas.
Fecha de recuperación: viernes 25 de julio, de 19:00 a 20:00 horas.

Dado un grafo simple $G = (V, E)$, una secuencia de vértices $P = [v_1, v_2, \ldots, v_l]$, con $v_i \in V$ para $i = 1, \ldots, l$, es un *camino* de v_1 a v_l si y sólo si $v_i v_{i+1} \in E$ para todo $i = 1, \ldots, (l-1)$. Dada una función de costo $\omega : E \to \mathbb{R}_+$, el *costo* de un camino $P = [v_1, \ldots, v_l]$ según ω se define como:

$$\omega(P) = \sum_{i=1}^{l-1} \omega(v_i v_{i+1})$$

Siendo $G = (V, E)$ un grafo con dos vértices $u, v \in V$, dadas dos funciones de costo, ω_1 y ω_2, definidas sobre E y un valor $K \in \mathbb{R}_+$, el problema de *Camino Acotado de Costo Mínimo* (CACM) consiste en hallar un camino P de u a v con costo $\omega_1(P) \leq K$ de manera tal que el costo $\omega_2(P)$ sea mínimo.

En el presente trabajo práctico se pide:

1. Describir situaciones de la vida real que puedan modelarse utilizando CACM.
2. Diseñar e implementar un algoritmo exacto para CACM 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 CACM los siguientes:
   - al menos una heurística constructiva golosa (u opcionalmente otro tipo de heurística que cuente con alguna componente golosa e identificarla),
   - al menos una heurística de búsqueda local, y
   - al menos un algoritmo que use la metaheurística GRASP [1].

   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 CACM 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. Este item puede omitirse para la heurística GRASP.
   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. Heurística GRASP (del item 3).
VI. Experimentación general (del item 4).

Para aprobar el trabajo se requiere aprobar todas las partes del mismo. La nota final del trabajo será un promedio ponderado de las notas finales de las partes y el trabajo práctico se aprobará con una nota de 5 (*cinco*) o superior. 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. En caso de reentregar, la nota final del trabajo será el 20 % del puntaje otorgado en la primera corrección más el 80 % del puntaje obtenido al recuperar.

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: La entrada contiene varias instancias del problema. Cada instancia representa un grafo G y comienza con una línea con 5 valores enteros positivos, n, m, u, v y K, separados por espacios. Los valores n y m indican la cantidad de vértices y aristas de G, respectivamente, los valores u y v (ambos entre 1 y n) son dos vértices del grafo y K es la cota para el peso en ω_1 del camino a encontrar. A continuación, le siguen m líneas, representando las aristas del grafo. Cada una de estas líneas tiene el formato:

```
v1 v2 w1 w2
```

donde `v1` y `v2` son los extremos de la arista representada (numerados de 1 a n) y `w1` y `w2` son los pesos de las funciones ω_1 y ω_2, respectivamente, correspondientes a la arista. Se puede suponer que los grafos son simples (i.e., no tienen bucles ni ejes repetidos). La entrada concluye con una línea comenzada por `0`, la cual no debe procesarse.

Formato de salida: La salida debe tener una línea por cada instancia procesada. Para cada instancia, en caso de encontrar una solución, dicha línea deberá tener el siguiente formato:

```
W1 W2 k v1 v2 ... vk
```

donde `W1` y `W2` son los pesos del camino obtenido asociados a las funciones ω_1 y ω_2, respectivamente, `k` es el tamaño del camino y `v1`, ..., `vk` son los vértices que lo conforman (en orden de u a v). En caso de haber más de un camino óptimo, el algoritmo puede devolver cualquiera de ellos. Por el contrario, si el algoritmo no encuentra una solución factible, la línea de salida deberá decir simplemente `no`.

Referencias

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