Trabajo práctico 3

Fecha de entrega: viernes 28 de Junio, hasta las 20:00 horas.
Fecha de recuperación: una semana después de la devolución del trabajo corregido.

Dado un grafo $G = (V, E)$, un coloreo de G es una función $\mathcal{C} : V \to \mathbb{N}$ que asigna un "color" a cada vértice del grafo de manera tal que $\mathcal{C}(v) \neq \mathcal{C}(w)$ para cada arista $vw \in E$. El problema clásico de coloreo de grafos consiste en hallar la cantidad mínima de colores necesaria para colorear G, la cual se denomina el *número cromático* de G y se denota por $\chi(G)$. Sin embargo, existe una gran cantidad de variantes del problema de coloreo en donde el objetivo es optimizar algún otro aspecto del coloreo hallado. A continuación definimos una de estas variantes.

Sean G y H dos grafos sobre el mismo conjunto de vértices V, es decir $G = (V, E_G)$ y $H = (V, E_H)$. Dado un coloreo $\mathcal{C}$ de G, se define el *impacto* de $\mathcal{C}$ en H, como la cantidad de aristas $vw \in E_H$ tal que $\mathcal{C}(v) = \mathcal{C}(w)$. El problema de *coloreo de máximo impacto* (CMI) sobre G y H consiste en hallar un coloreo de G que maximice el impacto en H. Vale aclarar que el coloreo hallado no necesariamente minimiza la cantidad de colores utilizados.

En el presente trabajo práctico se pide:

1. Describir situaciones de la vida real que puedan modelarse utilizando CMI.
2. Diseñar e implementar un algoritmo exacto para CMI 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 CMI los siguientes:
   - al menos una heurística constructiva golosa,
   - 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 utilizando el modelo uniforme.
   c) Describir (si es posible) instancias de CMI 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. 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. De ser necesario (o si el grupo lo desea) cada parte podrá ser recuperada individualmente. En caso de recuperar, la nota final de cada parte 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 correr 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 descriptos anteriormente. 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 *"TP3: 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 la entrada y salida estándar del sistema respetando los siguientes formatos:

Formato de entrada: La entrada contiene varias instancias del problema. Cada instancia comienza con una línea con tres valores enteros n, m_g y m_h separados por espacios. El valor n indica la cantidad de vértices de los grafos G y H y los valores m_g y m_h indican las cantidades de aristas de G y de H, respectivamente. A continuación, le siguen m_g líneas, cada una determinando una arista de G y luego otras m_h líneas determinando las aristas de H. 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). La entrada concluye con una línea comenzada por `#`, la cual no debe procesarse.

Formato de salida: La salida debe contener una línea por cada instancia de entrada, con el siguiente formato:

```
I c1 c2 ... cn
```

donde `I` es el impacto en H del coloreo obtenido y `c1`, ..., `cn` son los colores asignados a cada uno de los vértices de los grafos. Es decir, `ci` es el color asignado al vértice i.

Referencias

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