UBA - FCEyN - Departamento de Computación

ALGORITMOS Y ESTRUCTURAS DE DATOS III
Trabajo Práctico Nº3
Heurística Grasp aplicada al Problema de Dibujo de Grafos Incrementales Bipartitos.

Primera entrega: Viernes 27-6-2008, hasta las 19:30 horas
Segunda entrega: Lunes 14-7-2008, entre las 18 y 19:00 horas

Ver información general sobre los Trabajos Prácticos en la página de la materia en Internet.

Sea $G = (V_1 \cup V_2, E)$ un grafo bipartito. Un mismo grafo se puede dibujar de diferentes maneras, permutando el orden (la posición) de los vértices en cada partición. Llamamos *dibujo* de G, $D = (G, \sigma_1, \sigma_2)$ a un dibujo de G donde σ_1 y σ_2 son las permutaciones de los vértices de los conjuntos V_1 y V_2 respectivamente (i.e. $\sigma_i(v)$ es la posición correspondiente al vértice v en el *dibujo* D). Dados $v, w \in V_1$, $\sigma_1(v) < \sigma_1(w)$ (análogamente V_2) se define el valor $K(v, w)$ como la cantidad de "cruces" que hay entre aristas incidentes a v y aristas incidentes a w. Dado un "dibujo" D, se define *el número de cruces* $K(D)$ de un *dibujo* $D = (G, \sigma_1, \sigma_2)$ de la siguiente manera:

$$K(D) = \sum_{\substack{v,w \in V_1 / \\ \sigma_1(v) < \sigma_1(w)}} K(v, w) = \sum_{\substack{v,w \in V_2 / \\ \sigma_2(v) < \sigma_2(w)}} K(v, w)$$

Un grafo bipartito $IG = (IV_1, IV_2, IE)$, es un *grafo incremental* de G si $V_i \subseteq IV_i$ y $E \subseteq IE$. Un *dibujo* de IG es un *dibujo incremental* $ID = (IG, \pi_1, \pi_2)$ si para todo par de vértices $v, w \in V_i$, con $\sigma_i(v) < \sigma_i(w)$, vale que $\pi_i(v) < \pi_i(w)$ (i.e. la permutación π mantiene el orden relativo para los vértices de G).

Dado un grafo incremental IG y un dibujo D de G, el problema del **Dibujo de Grafos Incrementales Bipartitos (DGIB)** [1] consiste en encontrar un dibujo incremental ID que minimice el número de cruces.

1. Describir situaciones de la vida real que puedan modelarse utilizando **DGIB**.

2. Desarrollar e implementar un algortimo exacto para **DGIB**.

3. Desarrollar e implementar una heurística constructiva para **DGIB**.

4. Desarrollar e implementar una heurística de búsqueda local para **DGIB**.

5. Desarrollar e implementar un algortimo que use la metaheurística GRASP [2] para **DGIB**. En cada iteración de GRASP utilizar como primera fase alguna modificación de la heurística dada en 3, y como segunda fase la heurística dada en 4. **FIJAR** todos los parámetros involucrados (tamaño de la lista restringida de candidatos (LRC), criterios de parada, etc) mediante experimentos prácticos. **JUSTIFICAR** las elecciones y justificar también la elección de las instancias de prueba consideradas para decidir los parámetros.

6. Para cada uno de los métodos de los ejercicios 2 a 5:

   - Calcular la complejidad.
   - Tratar de describir instancias de **DGIB** para las cuales el método no proporciona una solución óptima. ¿Qué tan mala puede ser la solución obtenida respecto de la solución óptima?

- Aplicar el método a varias instancias de **DGIB**, respetando los formatos de archivos que se indican más abajo (utilizar como ejemplo los archivos provistos).
- Analizar la calidad de las soluciones obtenidas y el tiempo de ejecución.

7. Presentar los resultados obtenidos en el ejercicio anterior mediante gráficos adecuados, y utilizarlos para comparar los distintos métodos entre sí.

Tp3.in: Leer los datos de entrada en un archivo con este nombre. Cada instancia se representa con $(n_1 + 1) + (n_2 + 1) + (m + 1) + (n'_1 + 1) + (n'_2 + 1) + (m' + 1)$ líneas. En la primera línea del archivo figurará la cantidad n_1 de vértices del conjunto V_1. Luego, en las siguientes n_1 líneas aparecen los vértices de acuerdo con la permutación (el vértice j en la línea k significa que $\sigma_1(j) = k$. Luego, aparece la cantidad n_2 de vértices en V_2 y en las siguientes n_2 líneas los vértices de acuerdo con la permutación σ_2. Luego, una línea que contiene el número m de aristas en G, y en las siguientes m líneas, las aristas correspondientes, de la forma k j. Luego aparece la cantidad de vértices n'_1 que se agregan a V_1 y luego los n'_1 vértices. Luego aparece la cantidad n'_2 de vértices que se agregan a V_2, y luego los n'_2 vértices correspondientes. Por último, una línea que contiene la cantidad m' de aristas que se agregan en IG y luego las m' aristas. El archivo termina con una línea donde n_1 vale -1, la cual no debe interpretarse como una instancia de entrada.

Ejemplo:

4 (cantidad de vértices en V_1)

2

3

1

4

3 (cantidad de vértices en V_2)

6

5

7

2 (cantidad de aristas de V_1 a V_2)

1 7

4 6

2 (cantidad de vértices agregados a V_1 con numeración consecutiva)

8

9

3 (cantidad de vértices agregados a V_2)

10

11

12

4 (cantidad de aristas agregadas de IV_1 a IV_2)

8 5

9 6

11 8

10 2

Tp3X.out: Escribir los resultados en un archivo con este nombre, donde X identifica al método utilizado. Los valores contenidos en cada línea del archivo están separados entre sí por exactamente un espacio en blanco, y no son precedidos ni seguidos por ningún caracter dentro de la línea. El archivo contiene tres líneas por cada instancia de entrada. En la primera línea, el valor de $K(ID)$. En la siguiente línea, la cantidad de vértices de IV_1 y luego los vértices ordenados según su π. Análogamente, en la última línea aparece la cantidad de vértices de IV_2 y después los respectivos vértices ordenados según el valor de π.

Referencias

[1] R. Martí and V. Estruch. Incremental bipartite drawing problem. Computers and Operations Research, 28:1287-1298, 2001.

[2] Thomas A. Feo and Mauricio G. C. Resende. Greedy randomized adaptive search procedures. Journal of Global Optimization, pages 1-27, 1995.