Tarea 3 - TSP Euclideano

CC4102/CC53A - Diseño y Análisis de Algoritmos
Profesor: Gonzalo Navarro Auxiliar: Miguel Romero

Fecha de Entrega: 28 de Noviembre del 2014

1 Introducción

El TSP Euclideano (TSP por *traveling salesman problem*) se define como sigue: Dado un conjunto P de n puntos en el plano, queremos encontrar un *circuito hamiltoniano* de costo mínimo. Un circuito hamiltoniano es una forma de recorrer todos los puntos de P de manera de pasar por cada punto exactamente una vez. El costo de un circuito hamiltoniano es la distancia total recorrida, donde la distancia entre un par de puntos es la distancia euclidiana, denotada *dist*. Más formalmente, si el circuito hamiltoniano H está dado por una secuencia de puntos $p_1, \ldots, p_n$, entonces el costo de H es $\sum_{i=1}^{n-1} dist(p_i, p_{i+1}) + dist(p_n, p_1)$.

Este problema es NP-hard, por lo cual se cree que no existe un algoritmo polinomial que lo resuelva. El objetivo de esta tarea es implementar algunas aproximaciones/heurísticas para este problema, y evaluarlas sobre datos geográficos reales. Se pide implementar:

1. Algoritmo de fuerza bruta basado en programación dinámica.
2. 2-aproximación basada en el *minimum spanning tree* (MST).
3. Una heurística basada en *envoltura convexa*.

Se espera que el alumno implemente individualmente los algoritmos y entregue un informe que indique claramente los siguientes puntos:

1. Las *hipótesis* escogidas **antes** de realizar los experimentos.
2. El *diseño experimental*, incluyendo los detalles de la implementación de los algoritmos, la generación de las instancias y las medidas de rendimiento utilizadas.
3. La *presentación de los resultados* en forma de una descripción textual, tablas y/o gráficos.
4. El *análisis e interpretación* de los resultados.

2 Fuerza bruta basada en programación dinámica

El algoritmo básico de fuerza bruta itera por todas las posibles permutaciones de puntos en P y busca el circuito de costo mínimo. Si el tamaño de P es n, la cantidad de permutaciones es $(n-1)!$ (podemos dejar fijo el primer punto p_1 y solo permutar el resto) y computar el costo es $O(n)$. Luego este algoritmo toma tiempo $O(n!)$. A continuación se describe un algoritmo que toma tiempo $O(n^2 2^n)$ basado en programación dinámica (notar que $n^2 2^n = o(n!)$).

Supongamos que $P = \{p_1, \ldots, p_n\}$. Usamos una tabla T donde las filas corresponden a subconjuntos de P que contienen el primer punto p_1, y las columnas corresponden a los puntos $p_2, \ldots, p_n$. Si S es un subconjunto que contiene p_1 y $p \in \{p_2, \ldots, p_n\}$ es un punto que pertenece a S, entonces la casilla $T(S, p)$ contiene el costo mínimo de un camino que parte en p_1, termina en p y

pasa exactamente una vez por todos los puntos en S. Para llenar T podemos usar la siguiente recurrencia

$$T(S,p) = min\{T(S \setminus \{p\}, q) + dist(q,p) : q \in S, q \neq p_1, q \neq p\}$$

El caso base ocurre cuando $|S| = 2$, donde $T(\{p_1, p\}, p) = dist(p_1, p)$. El costo del circuito óptimo es $min\{T(\{p_1, \ldots, p_n\}, p) + dist(p, p_1) : p \in \{p_2, \ldots, p_n\}\}$. Agregando la información necesaria en la tabla, no es dificil modificar el algoritmo para que nos entregue el circuito óptimo.

3 Aproximación basada en MST

El siguiente algoritmo es una 2-aproximación y esta basado en el MST (*minimum spanning tree*). Dado un conjunto de puntos P el algoritmo procede como sigue:

1. Construimos el grafo G *asociado* a P. Los nodos de G son los puntos en P y entre cada par de puntos distintos hay una arista. Además, hay una función de peso w que asocia a cada arista $\{p, p'\}$ un peso $w(\{p, p'\}) = dist(p, p')$.
2. Construimos un MST T para (G, w).
3. Enraizamos T en un nodo arbitrario r, ordenamos los hijos de cada nodo de manera arbitraria y hacemos un recorrido en *pre-orden* de T (padre y luego primer hijo no visitado). Retornamos este recorrido.

La Figura 1 muestra un árbol y su recorrido pre-orden. Cada vez que la línea punteada pasa por un punto negro, sea anota ese nodo en la salida. En este caso, el recorrido es $F, B, A, D, C, E, G, I, H$. Observe que el recorrido entregado por el algoritmo es siempre un circuito hamiltoniano.

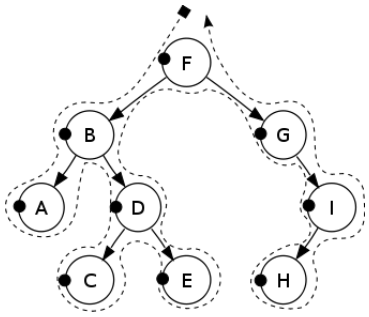

Figure 1: Recorrido pre-orden

Para construir el MST debe implementar el algoritmo de Kruskal o el algoritmo de Prim (Cormen, capítulo 23).

4 Heurística basada en la envoltura convexa

Esta heurística no posee ninguna garantía formal. La *envoltura convexa* (*convex hull* en inglés) de un conjunto de puntos P es el conjunto convexo más pequeño que contiene a todos los puntos. La envoltura convexa queda determinada por un subconjunto C de P, donde C son los puntos extremos de esta. En la Figura 2 se puede ver la envoltura convexa de un conjunto de puntos. El borde de la envoltura es el polígono de color azul.
Dado un conjunto de puntos P, la heurística es como sigue:

1. Comenzamos con un circuito C dado por la envoltura convexa de P.

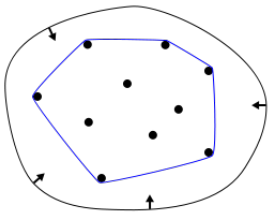

Figure 2: Envoltura convexa

2. En cada paso, para cada punto p fuera de C, buscamos el arco (p_1, p_2) en C, que minimiza $dist(p, p_1) + dist(p, p_2) - dist(p_1, p_2)$. A continuación, escogemos el punto p^* fuera de C que minimiza
$$\frac{dist(p^*, p_1^*) + dist(p^*, p_2^*)}{dist(p_1^*, p_2^*)}$$
y extendemos C, incluyendo p^* entre los puntos p_1^* y p_2^*.
3. Si ya no quedan puntos fuera de C, retornamos C.

No es necesario implementar un algoritmo que construya la envoltura convexa. Puede utilizar algunas de las librerías disponibles en la Web.

5 Experimentos

Para los experimentos, utilizaremos el set de datos geográficos `National TSPs`, disponible en el siguiente link: `http://www.math.uwaterloo.ca/tsp/world/countries.html`. Sólo consideraremos países que no tengan duplicados.

Primero, debemos generar un conjunto de instancias de tamaño n, para todo $n \in \{5, \ldots, 20\}$. Para esto, escoja un país entre Vietnam, Japan o Finland y escoja n ciudades al azar en ese país. Una vez generado el set de instancias deberá comparar los tres algoritmos. En este caso, haga los experimentos hasta $n = 20$ o hasta que el algoritmo de fuerza bruta se lo permita.

Luego consideraremos países cuyo óptimo es conocido. Estos son: Djibouti, Finland, Greece, Japan, Oman, Qatar, Sweden, Uruguay, Vietnam, Western Sahara, Zimbabwe, Canada, Italy. En este caso, sólo mediremos el desempeño del algoritmo aproximado y de la heurística.

En cualquier caso, deberá reportar:

- Tiempo de ejecución.
- Ratio ALG/OPT.

6 Entrega de la Tarea

- La tarea puede realizarse en grupos de a lo más 2 personas.
- Se descontará medio punto por día de atraso (hábil o no).
- Para la implementación puede utilizar `C`, `C++`, `Java` o `Python`. Para el informe se recomienda utilizar LaTeX.
- Escriba un informe claro y conciso. Las ponderaciones del informe y la implementación en su nota final son las mismas.
- La entrega será a través de U-Cursos y deberá incluir el informe junto con el código fuente de la implementación (y todas las indicaciones necesarias para su ejecución). El día hábil siguiente a la entrega, deberá dejar un informe impreso donde Sandra.