Práctica 3
Algoritmos Greedy

ÓSCAR BERMÚDEZ,
FRANCISCO DAVID CHARTE,
IGNACIO CORDÓN,
JOSÉ CARLOS ENTRENA,
MARIO ROMÁN
Universidad de Granada
22 de abril de 2014

Resumen

Índice

1. Terminales de venta

1.1. Implementación

Implementamos en Ruby la heurítica pedida.

```ruby
#!/usr/bin/env ruby
# encoding: utf-8

def cambio (monedas, precio)
  vuelta = []

  monedas.sort.reverse.each { |moneda|
    numero_monedas = precio / moneda
    precio = precio - numero_monedas*moneda

    vuelta.push [moneda, numero_monedas]
  }

  return vuelta
end
```

2. Red de comunicaciones

2.1. Algoritmo

Dado un conjunto de ciudades, buscamos interconectarlas con una red que minimice la longitud de red. Modelizaremos el problema como un grafo completo G del conjunto de ciudades E. Con aristas $V = \{[a, b] |\ a, b \in C\}$, donde la arista conectando los nodos a y b tiene peso igual a la distancia que los separa:

$$\forall [a, b] \in V : \quad w([a, b]) = dist(a, b) \tag{1}$$

Pretendemos interconectar ciudades de manera que la suma total de las distancias de los caminos hechos sea mínima. Es decir, buscamos el subgrafo recubridor de menor peso. Como un grafo recubridor con ciclos tiene siempre un subgrafo recubridor estrictamente contenido en él, buscamos sólo entre los grafos acíclicos. El subgrafo acíclico recubridor de menor costo es el árbol recubridor minimal. Es decir, buscamos el árbol recubridor minimal euclídeo de un conjunto de puntos.

Son conocidos los algoritmos de Prim y Kruskal para calcular el árbol recubridor minimal. Ambos alcanzan una complejidad temporal de $\mathcal{O}(|E|log|V|)$ usando árboles binarios y listas de adyacencia para representar el grafo.

2.2. Triangulación de Delaunay

Para reducir la carga del algoritmo, podemos reducir el grafo sobre que que buscamos el árbol generador minimal. El grafo inicial es completo, ya que toda ciudad es suceptible de ser comunicada con cualquiera otra. Pero al ser un grafo completo sobre puntos en un plano euclídeo, podemos aprovechar las propiedades de la distancia para reducir el grafo sobre el que buscamos.

Sin embargo, podemos demostrar que el árbol generador minimal está contenido la triangulación de Delaunay; y que, por tanto, sólo es necesario aplicar el algoritmo de Kruskal o Prim al subgrafo resultante de la triangulación.

Demostración. Demostraremos que cada arista del árbol generador está contenida en la triangulación de Delaunay. Sea (p, q) una arista arbitraria del árbol generador. Consideramos el círculo que tiene como diámetro $\overline{pq}$, si hubiera otro punto r en este círculo, tendríamos:

$$\overrightarrow{pr} \leq \overrightarrow{pq} \qquad \overrightarrow{rq} \leq \overrightarrow{pq} \tag{2}$$

Y entonces podríamos formar el ciclo p, r, q, p. Sabemos que la arista de mayor peso en un ciclo no forma parte del árbol generador minimal y por tanto $\overline{pq}$ no pertenecería a él, llegando a contradicción.

Así, no puede haber ningún punto en el círculo que tiene como diámetro a $\overline{pq}$. Una arista que cumple esto está forzosamente en el diagrama de Delaunay, por aplicación de la Condición de Delaunay. □

Para encontrar la triangulación de Delaunay usar

3. **Segmentación de clientes**

4. **Asignación de trabajos**

5. **Asignación de aulas**

6. **Memorias caché**

7. **El problema de asignación cuadrática**