Práctica 3: Algoritmos Voraces (Greedy)

Manuel Vicente Bolaños Quesada
Pablo Gálvez Ortigosa
Carlos García Jiménez

18/5/2022

- Problema 1

  Buque de carga con capacidad K, $\{p_1, \ldots, p_n\}$
  - Maximizar número de contenedores
  - Maximizar carga

- Problema 2 (TSP)

  Ciclo hamiltoniano de mínimo peso de grafo conexo y ponderado
  - Vecino más cercano
  - Inserción
  - Heurística propuesta por el grupo

Características Greedy

- **Candidatos**: contenedores
- **Candidatos ya usados**: candidatos cargados en el barco o candidatos que superan la capacidad.
- **Criterio de parada**: añadir un contenedor de candidatos a nuestra lista supone superar la carga del barco.
- **Criterio solución**: un conjunto es solución si no supera la capacidad del barco
- **Función de selección**
  - Caso 1: Contenedor con menos peso de candidatos
  - Caso 2: Contenedor con más peso de candidatos
- **Función objetivo**: peso de cada conjunto de contenedores

Maximizando el número de contenedores

- Se ordenan los pesos de los contenedores de menor a mayor
- Se cargan los contenedores en orden, hasta que el peso total supere la capacidad
- Eficiencia teórica: $O(n \log n)$
- Algoritmo **óptimo**

Sketch de la demostración

- $p_1 \leq p_2 \leq \cdots \leq p_n$
- $\sum_{i=1}^{k} p_i \leq K, \quad \sum_{i=1}^{k+1} p_i > K$
- $n \geq m > k$. Definimos $\varphi : \{1, \ldots, m\} \to \{1, \ldots, n\}$
- φ inyectiva y creciente $\implies \varphi(i) \geq i$
- $\sum_{i=1}^{m} p_{\varphi(i)} \geq \sum_{i=1}^{m} p_i \geq \sum_{i=1}^{k+1} p_i > K$

Maximizando las toneladas cargadas

- Se ordenan los pesos de los contenedores de mayor a menor
- Se introducen en el barco en orden, siempre que su peso sumado a lo ya cargado no supere la capacidad del barco
- Recorre todo el vector

Eficiencia teórica: $\mathbf{O(n \log n)}$. El algoritmo no es el óptimo.
Contraejemplo: Pesos $= \{7, 5, 4, 3\}$, Capacidad $= 14$, Respuesta algoritmo $= \{7, 5\}$, Solución óptima $\{7, 4, 3\}$

Características Greedy

- **Candidatos**: nodos del grafo de ciudades
- **Candidatos ya usados**: nodos ya recorridos en el camino
- **Criterio de parada**: el camino recorre todos los nodos
- **Criterio solución**: un conjunto es solución si recorre todos los nodos una única vez, y comienza y termina en el mismo nodo
- **Función de selección**: dependerá de la heurística (vecino más cercano, inserción menos costosa, ...)
- **Función objetivo**: distancia total de un ciclo hamiltoniano dado

Vecino más cercano

- Función *min_node* calcula el nodo más cercano de los candidatos
- Introduce ese nodo en la ruta y lo elimina de los candidatos
- Repite el proceso hasta que se acaban los candidatos y vuelve al punto inicial

Eficiencia teórica: $\mathbf{O(n^2)}$

Heurísticas II: Inserción

- Partimos ya de un circuito!
- Comenzamos con tres nodos (más al este, más al oeste y más al norte).
- Nodos candidatos $\Rightarrow$ Insertamos nodo que aumente menos la distancia del recorrido

Eficiencia teórica $\mathbf{O(n^3)}$ $\Rightarrow$ muchas comprobaciones para número de nodos elevado

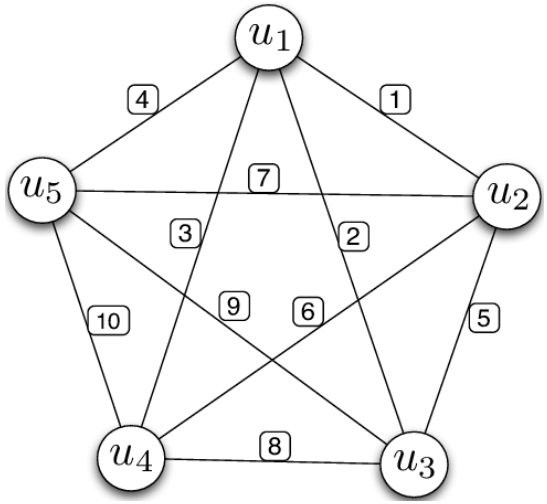
Heurísticas III: Propuesta

Similar a Vecino más cercano

1. Iniciamos recorrido con dos nodos (Arista de menos peso).
2. Calculamos nodos más cercanos para cada nodo de los anteriores.
3. Insertamos el de menor distancia (como primer o como último nodo)
4. Repetir dos pasos anteriores con los nuevos extremos
5. Cuando solo queda un nodo, lo insertamos al principio y al final del recorrido, cerrando el ciclo.

Eficiencia teórica: misma que vecino más cercano $\Rightarrow \mathbf{O(n^2)}$

Heurísticas III: Propuesta (Ejemplo)



1. Mínima arista $\rightarrow \{1, 2\}$
   - Más cercano a 1: 3 a distancia 2
   - Más cercano a 2: 3 a distancia 5
2. Insertamos 3 $\rightarrow \{3, 1, 2\}$
3. Mismo procedimiento $\rightarrow \{3, 1, 2, 4\}$
4. Finalmente $\{5, 3, 1, 2, 4, 5\}$

Comparación Heurísticas (I)

- Utilizaremos 3 ficheros: 16, 29 y 76 nodos resp.
- Suponemos grafo completo, parte entera de distancia euclídea $\Rightarrow$ Calculamos matriz de adyacencia
- Comparamos tanto tiempos como resultados obtenidos
- A continuación, veremos el proceso con el grafo de 76 ciudades (en memoria, realizado con los tres archivos).

Comparación Heurísticas (II)

Ejemplo matriz adyacencia (76 ciudades):

$$\begin{pmatrix}
\infty & 14 & 23 & 26 & 33 & 16 & 39 & 40 & 37 & 47 & \dots & 19 & 33 & 46 & 40 & 40 & 5 & 18 & 23 & 25 \\
14 & \infty & 24 & 12 & 19 & 8 & 27 & 26 & 34 & 40 & \dots & 7 & 26 & 34 & 31 & 34 & 9 & 7 & 11 & 14 \\
23 & 24 & \infty & 26 & 42 & 16 & 29 & 34 & 14 & 28 & \dots & 20 & 50 & 58 & 55 & 20 & 21 & 31 & 20 & 19 \\
26 & 12 & 26 & \infty & 18 & 12 & 15 & 14 & 30 & 31 & \dots & 7 & 31 & 34 & 33 & 26 & 21 & 15 & 5 & 7 \\
33 & 19 & 42 & 18 & \infty & 26 & 30 & 25 & 48 & 48 & \dots & 21 & 16 & 16 & 15 & 44 & 28 & 15 & 22 & 25 \\
16 & 8 & 16 & 12 & 26 & \infty & 23 & 24 & 25 & 32 & \dots & 5 & 34 & 42 & 38 & 26 & 11 & 15 & 7 & 9 \\
39 & 27 & 29 & 15 & 30 & 23 & \infty & 7 & 25 & 18 & \dots & 20 & 46 & 44 & 45 & 18 & 34 & 31 & 16 & 14 \\
40 & 26 & 34 & 14 & 25 & 24 & 7 & \infty & 32 & 25 & \dots & 20 & 41 & 38 & 40 & 25 & 35 & 29 & 17 & 15 \\
37 & 34 & 14 & 30 & 48 & 25 & 25 & 32 & \infty & 15 & \dots & 28 & 60 & 64 & 63 & 9 & 35 & 41 & 26 & 23 \\
47 & 40 & 28 & 31 & 48 & 32 & 18 & 25 & 15 & \infty & \dots & 33 & 62 & 63 & 63 & 7 & 43 & 46 & 29 & 26 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
19 & 7 & 20 & 7 & 21 & 5 & 20 & 20 & 28 & 33 & \dots & \infty & 31 & 37 & 35 & 27 & 14 & 13 & 4 & 7 \\
33 & 26 & 50 & 31 & 16 & 34 & 46 & 41 & 60 & 62 & \dots & 31 & \infty & 16 & 9 & 57 & 30 & 18 & 34 & 37 \\
46 & 34 & 58 & 34 & 16 & 42 & 44 & 38 & 64 & 63 & \dots & 37 & 16 & \infty & 7 & 60 & 42 & 28 & 38 & 41 \\
40 & 31 & 55 & 33 & 15 & 38 & 45 & 40 & 63 & 63 & \dots & 35 & 9 & 7 & \infty & 60 & 37 & 24 & 37 & 39 \\
40 & 34 & 20 & 26 & 44 & 26 & 18 & 25 & 9 & 7 & \dots & 27 & 57 & 60 & 60 & \infty & 36 & 40 & 23 & 20 \\
5 & 9 & 21 & 21 & 28 & 11 & 34 & 35 & 35 & 43 & \dots & 14 & 30 & 42 & 37 & 36 & \infty & 13 & 18 & 20 \\
18 & 7 & 31 & 15 & 15 & 15 & 31 & 29 & 41 & 46 & \dots & 13 & 18 & 28 & 24 & 40 & 13 & \infty & 17 & 20 \\
23 & 11 & 20 & 5 & 22 & 7 & 16 & 17 & 26 & 29 & \dots & 4 & 34 & 38 & 37 & 23 & 18 & 17 & \infty & 3 \\
25 & 14 & 19 & 7 & 25 & 9 & 14 & 15 & 23 & 26 & \dots & 7 & 37 & 41 & 39 & 20 & 20 & 20 & 3 & \infty
\end{pmatrix}$$

Comparación Heurísticas (III): Nube de Puntos de 76 ciudades

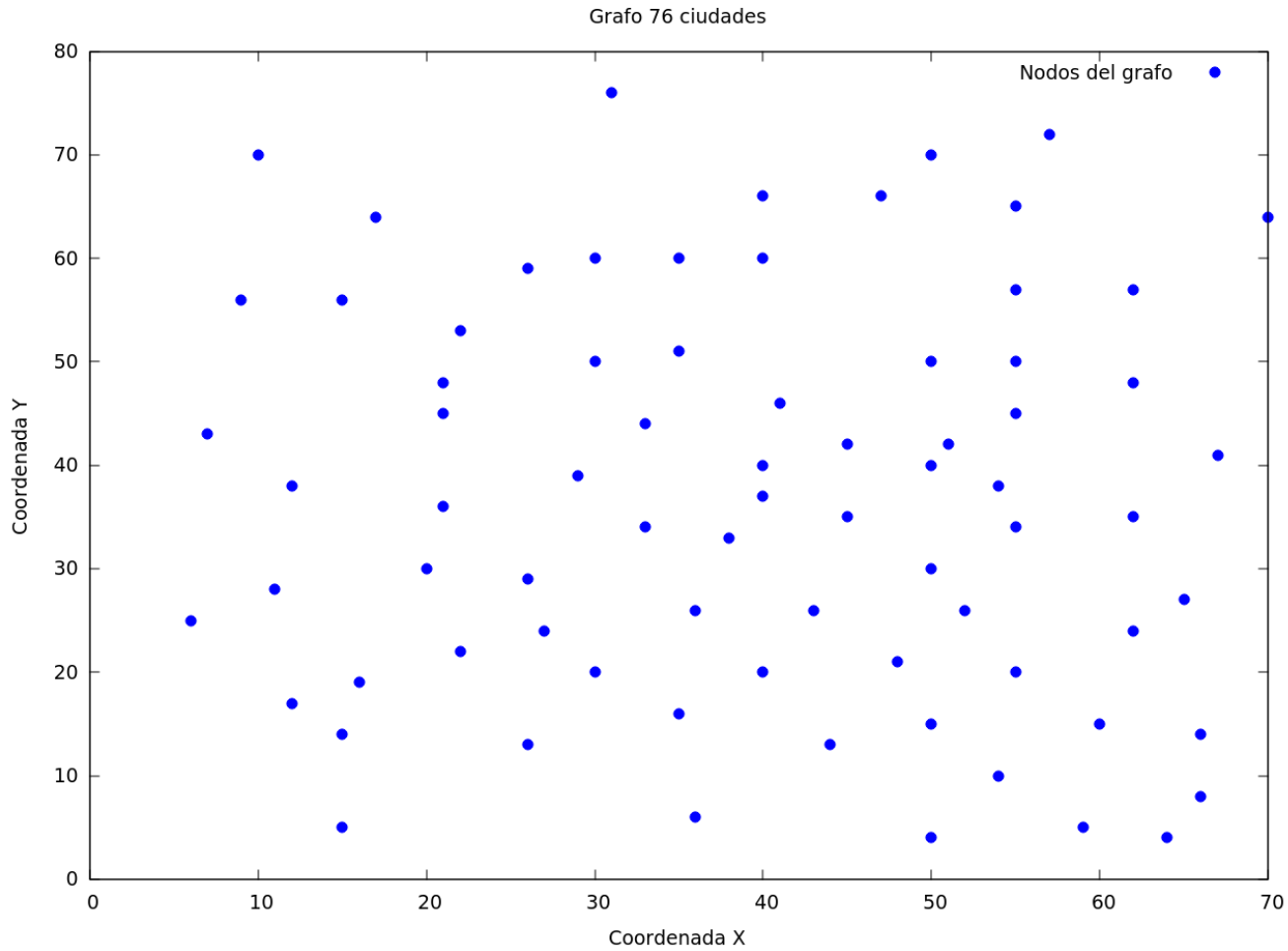


Figure 1: 76 ciudades

Comparación Heurísticas (IV): Resultados obtenidos (76 ciudades)

HEURÍSTICA	CAMINO	COSTE	TIEMPO (μs)
VECINO MÁS CERCANO	0 , 72 , 32 , 62 , 15 , 50 , 5 , 67 , 74 , 75 , 66 , 25 , 11 , 39 , 16 , 2 , 43 , 31 , 8 , 38 , 71 , 57 , 9 , 37 , 64 , 10 , 65 , 13 , 52 , 34 , 6 , 7 , 45 , 33 , 51 , 26 , 44 , 28 , 4 , 36 , 19 , 69 , 59 , 70 , 35 , 46 , 20 , 47 , 29 , 73 , 27 , 61 , 1 , 3 , 12 , 53 , 18 , 58 , 56 , 14 , 68 , 60 , 21 , 41 , 40 , 42 , 22 , 55 , 48 , 23 , 17 , 49 , 24 , 54 , 30 , 63 , 0	662	92
INSERCIÓN	58 , 64 , 8 , 38 , 10 , 57 , 11 , 37 , 65 , 25 , 6 , 52 , 75 , 66 , 74 , 67 , 34 , 45 , 33 , 3 , 7 , 51 , 44 , 26 , 29 , 28 , 4 , 13 , 18 , 12 , 53 , 14 , 56 , 36 , 19 , 69 , 70 , 68 , 60 , 59 , 49 , 42 , 54 , 63 , 41 , 24 , 40 , 17 , 21 , 35 , 46 , 20 , 27 , 61 , 0 , 47 , 73 , 1 , 72 , 62 , 32 , 23 , 5 , 50 , 16 , 15 , 39 , 48 , 43 , 55 , 22 , 2 , 31 , 71 , 9 , 30 , 58	1292	4414
3ª ALTERNATIVA	54 , 68 , 60 , 21 , 63 , 41 , 40 , 42 , 0 , 72 , 61 , 27 , 73 , 29 , 47 , 20 , 46 , 35 , 70 , 59 , 69 , 19 , 36 , 4 , 28 , 44 , 26 , 51 , 33 , 45 , 7 , 34 , 6 , 52 , 13 , 18 , 53 , 12 , 56 , 14 , 3 , 74 , 75 , 66 , 25 , 11 , 39 , 16 , 50 , 5 , 67 , 1 , 32 , 62 , 15 , 2 , 43 , 31 , 8 , 38 , 71 , 57 , 9 , 37 , 64 , 10 , 65 , 58 , 30 , 24 , 49 , 17 , 23 , 48 , 22 , 55 , 54	660	152

- Más rápido → Vecino más cercano
- Más lento → Inserción
- Mejores resultados → 3ª alternativa ligeramente superior a vecino más cercano

Comparación Heurísticas (V): Caminos obtenidos para 76 ciudades

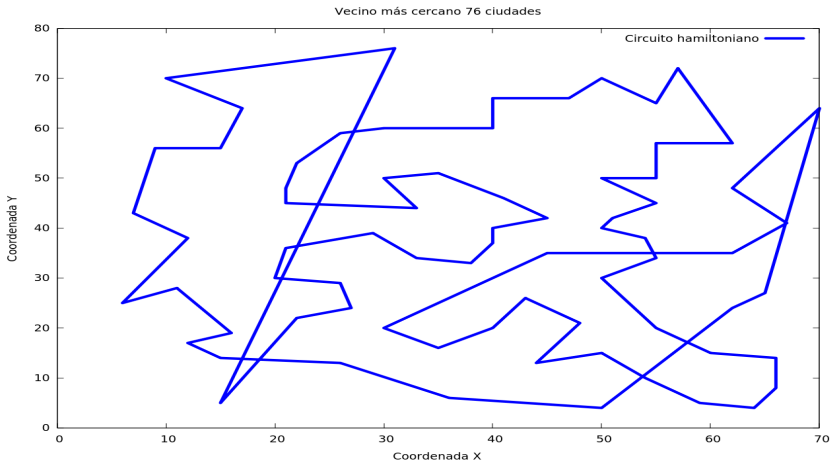


Figure 2: Vecino más cercano

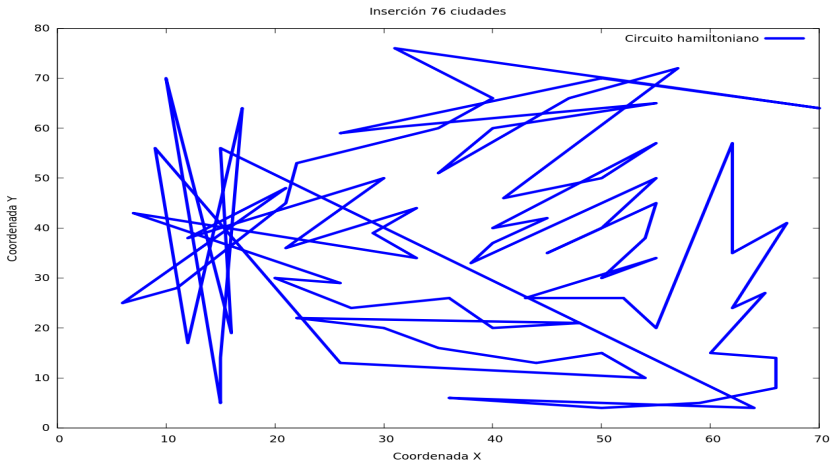


Figure 3: Inserción

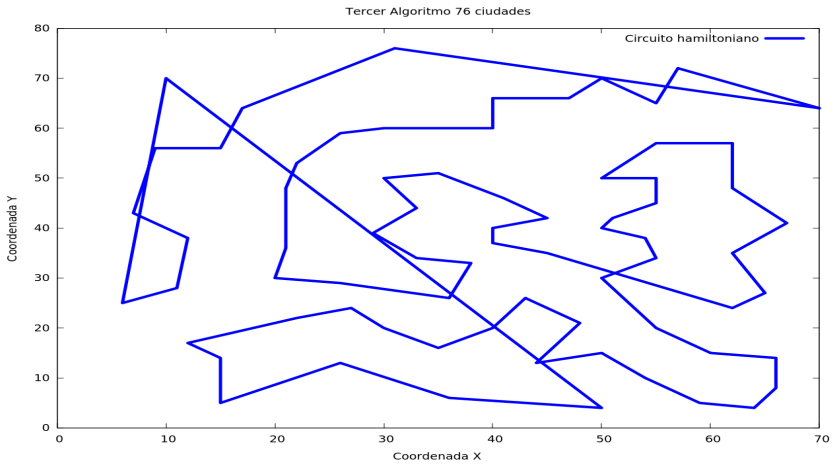


Figure 4: 3ª alternativa

Conclusión

- Hemos aprendido a verificar si se puede aplicar Greedy a un problema concreto
- Greedy $\rightarrow$ Soluciones muy eficientes $\rightarrow$ Soluciones óptimas o cercanas a óptimas (Importante comprobar en cada caso)
- Greedy es muy importante en problemas de grafos $\rightarrow$ soluciones aproximadas de problemas NP-Completos (TSP)
- Gran utilidad de heurísticas $\rightarrow$ Muy útiles en Algorítmica
- Hemos aprendido a trabajar con matrices de adyacencia (forma en la que el ordenador trabaja con grafos) y a hacer representaciones gráficas de grafos.