

Algoritmos Greedy

Problema QAP

DANIEL BOLAÑOS MARTÍNEZ

JOSÉ MARÍA BORRAS SERRANO

FERNANDO DE LA HOZ MORENO

SANTIAGO DE DIEGO DE DIEGO

A solid orange horizontal bar at the bottom of the slide.

Análisis del problema

El problema P, está basado en el Problema de Asignación Cuadrática.

Nuestro ejemplo consiste en asignar a cada oficinista de un grupo de oficinistas, una habitación de un grupo de habitaciones de forma que se minimice el coste de asignar a cada habitación i el oficinista $p(i)$.

$$p^* = \min_p H(p) = \min_p \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{p(i)p(j)} d_{ij}$$

Análisis del problema

Matriz de distancias

0	7	14	20	3
4	0	10	17	49
51	1	0	43	71
7	3	10	0	20
90	101	47	3	0

Distancia potencial: (44,80,160,40,241)

Matriz de flujos

0	4	7	4	1
0	0	10	3	21
0	0	0	47	3
41	21	7	0	9
21	43	32	0	27

Flujo potencial: (16,34,50,78,123)

Componentes Greedy

- **Lista de candidatos**: en este caso son las habitaciones y los oficinistas.
- **Función solución**: cuando el conjunto de candidatos se encuentra vacío se tiene la solución al problema.
- **Función selección**: se escoge la habitación con menor distancia potencial y el oficinista con mayor flujo potencial.
- **Función de factibilidad**: en este caso siempre se da la factibilidad.
- **Función objetivo**: asignar a oficinistas con la máxima carga de trabajo a habitaciones con la mínima distancia.

Pseudocódigo

`S ← 0`

`Mientras (habitaciones != 0 && oficinistas != 0) hacer:`

`x1 = Selección de la habitación con mínima
 distancia potencial entre las habitaciones.`

`x2 = Selección del oficinista con máximo
 flujo potencial entre los oficinistas.`

`habitaciones = habitaciones \ {x1}`

`oficinistas = oficinista \ {x2}`

`S[x1] = x2`

`Fin-Mientras`

`Devolver S`

Ejemplo Greedy

	H1	H2	H3	H4	H5
H1	0	7	14	20	3
H2	4	0	10	17	49
H3	51	1	0	43	71
H4	7	3	10	0	20
H5	90	101	47	3	0



	1	2	3	4	5
dp	44	80	160	40	241

$$d_p(i) = \sum_{j=0}^{N-1} d_{ij}.$$

	O1	O2	O3	O4	O5
O1	0	4	7	4	1
O2	0	0	10	3	21
O3	0	0	0	47	3
O4	41	21	7	0	9
O5	21	43	32	0	27



	1	2	3	4	5
fp	16	34	50	78	123

$$f_p = \sum_{b=0}^{N-1} f_{ab}$$

Ejemplo Greedy

d _p	1	2	3	4	5
	44	80	160	40	241

f _p	1	2	3	4	5
	16	34	50	78	123

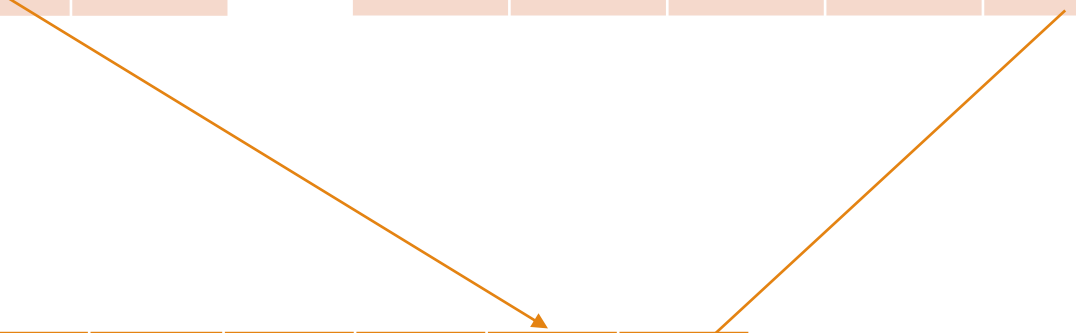
habitación	1	2	3	4	5
oficinista					

Ejemplo Greedy

d _p	1	2	3	4	5
	44	80	160	40	241

f _p	1	2	3	4	5
	16	34	50	78	123

habitación	1	2	3	4	5
oficinista				5	

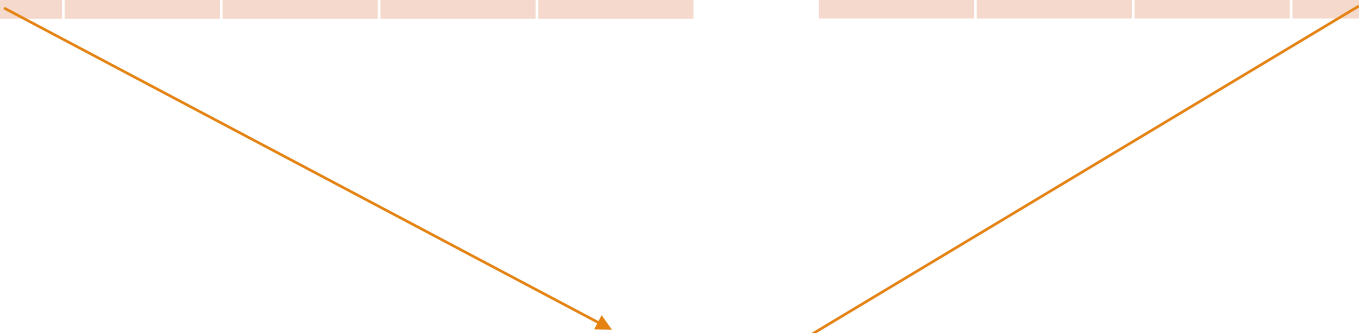


Ejemplo Greedy

d _p	1	2	3	4	5
	44	80	160	40	241

f _p	1	2	3	4	5
	16	34	50	78	123

habitación	1	2	3	4	5
oficinista	4			5	



Ejemplo Greedy

d _p	1	2	3	4	5
	44	80	160	40	241

f _p	1	2	3	4	5
	16	34	50	78	123

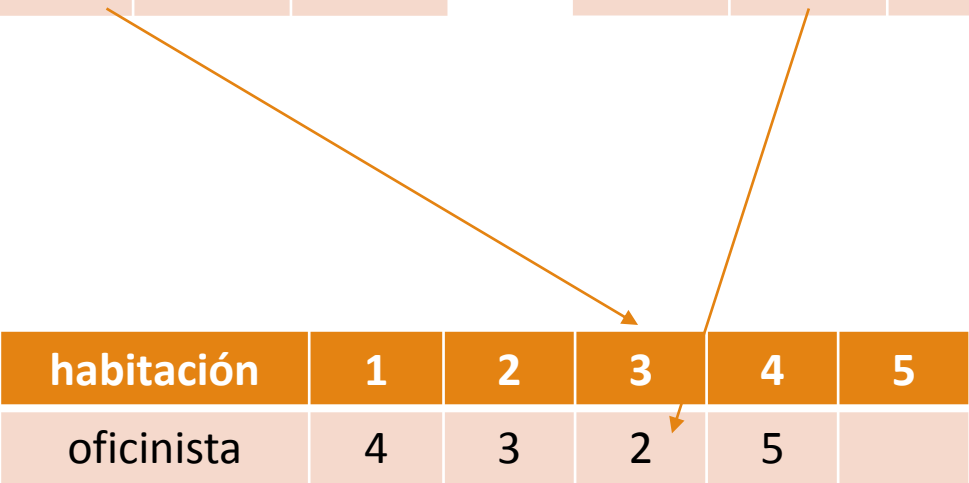
habitación	1	2	3	4	5
oficinista	4	3		5	

Ejemplo Greedy

d _p	1	2	3	4	5
	44	80	160	40	241

f _p	1	2	3	4	5
	16	34	50	78	123

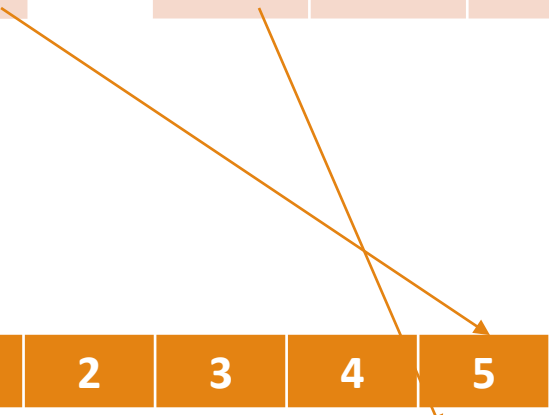
habitación	1	2	3	4	5
oficinista	4	3	2	5	



Ejemplo Greedy

d _p	1	2	3	4	5	f _p	1	2	3	4	5
	44	80	160	40	241		16	34	50	78	123

habitación	1	2	3	4	5
oficinista	4	3	2	5	1



Casos reales de aplicación del algoritmo

- Diseño de **centros comerciales** donde se quiere que el público recorra la menor cantidad de distancia para llegar a tiendas de mayor interés común.
- Diseño de **circuitos eléctricos**, en donde es de relevante importancia donde se ubican ciertas partes o chips con el fin de minimizar la distancia entre ellos, ya que las conexiones son de alto costo.

Casos reales de aplicación del algoritmo

- Haremos un pequeño esquema de cómo resolveríamos el ejemplo de los **centros comerciales** descrito tal y como hemos hecho con los oficinistas y habitaciones.
- **Lista de candidatos**: tiendas (distancias) y sector (interés).
- **Función solución**: cuando el conjunto de candidatos se encuentra vacío se tiene la solución al problema.
- **Función selección**: se escoge la tienda con menor distancia potencial y el sector con mayor flujo potencial.
- **Función de factibilidad**: en este caso siempre se da la factibilidad.
- **Función objetivo**: asignar a sectores con mayor interés a tiendas con la mínima distancia.

Orden de eficiencia teórica

Si n es el número de habitaciones y oficinistas el algoritmo tiene una eficiencia de $O(n^2)$.

Esto es debido a que el algoritmo tiene n etapas, en las que se elige la habitación de mínima distancia potencial y al oficinista con máximo flujo potencial en cada etapa. Esta elección de habitación y oficinista requiere recorrer dos vectores de tamaño n , buscando el mínimo y el máximo respectivamente, por lo que la eficiencia es $O(n^2)$.

Orden de eficiencia teórica

```
//Eficiencia:  $O(n^2)$ 
S <- 0
//Eficiencia:  $O(n)$ 
Mientras (habitaciones != 0 && oficinistas != 0) hacer:
    //Eficiencia:  $O(n)$ 
    x1 = Selección de la habitación con mínima
        distancia potencial entre las habitaciones.
    //Eficiencia:  $O(n)$ 
    x2 = Selección del oficinista con máximo
        flujo potencial entre los oficinistas.
    habitaciones = habitaciones \ {x1}
    oficinistas = oficinista \ {x2}
    S[x1] = x2
Fin-Mientras
Devolver S
```