Metaheurísticas

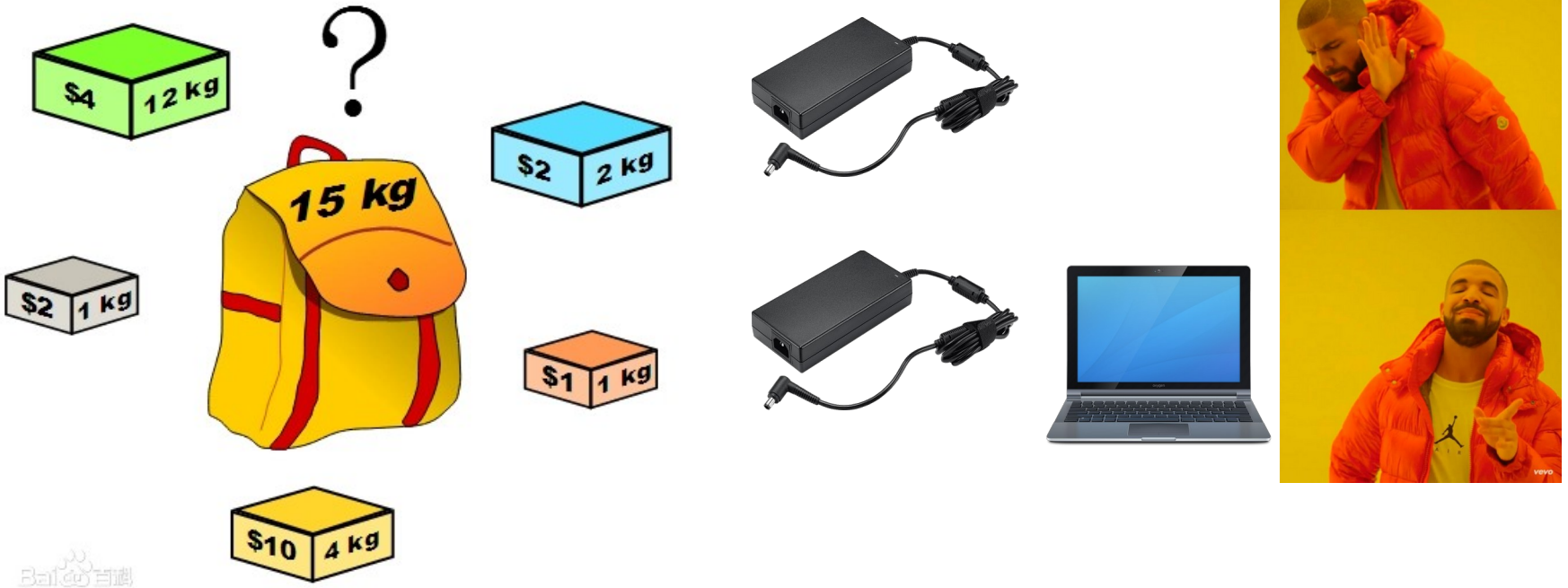
Seminario 2. Problemas de optimización con técnicas basadas en búsqueda local

Problema Cuadrático de la Mochila (QKP)

- Definición del Problema
- Representación y Ejemplo
- Solución Greedy
- Búsquedas por Trayectorias Simples
- Formato de los datos

Definición del Problema

- El Problema Cuadrático de la Mochila (QKP) está considerado como uno de los problemas de optimización combinatoria más complejos
- El problema general consiste en encontrar la selección óptima de entre *n* objetos dentro de una capacidad máxima, existiendo un beneficio no solo por los objetos de forma individual, también por pares de objetos.



Definición del Problema

- Es NP-completo. Mientras que el la mochila no cuadrática se puede alcanzar el óptimo, en esta versión no se pueden garantizar con tamaños de más de 100 objetos. Hay versiones que se ejecutan en un tiempo cercano a polinomial, pero necesitan un gran uso de memoria.

$$\text{X debe cumplir que} \left(x \in \{0,1\}^n : \sum_{i=1}^{n} w_i \cdot x_i \leq W \right)$$

Y se desea maximizar:

$$\max \left(\sum_{i=1}^{n} p_i \cdot x_i + \sum_{i=1}^{n} \sum_{j=1, j \neq i} p_{ij} \cdot x_i \cdot x_j \right)$$

Definición del Problema

- **Problema de la asignación cuadrática, *QKP***:

  Dadas n objetos y capacidad W, el problema consiste en maximizar la suma de los valores de los artículos en la mochila de modo que la suma de los pesos sea menor o igual a la capacidad de la mochila.

- Se puede representar una solución como un vector binario sol.
- La única restricción es que cumpla que:

$$\sum_{i=1}^{n} x_i \leq W$$

Representación y Ejemplo

- Supongamos que tenemos los siguientes parámetros:
- W =145.
- Wi = [34, 33, 12, 3, 43, 26, 10, 2, 48, 39]
- Pi = [91, 78, 22, 4, 48, 85, 46, 81, 3, 26]

$$P_{ij} = \begin{pmatrix} 0 & 55 & 23 & 35 & 44 & 5 & 91 & 95 & 26 & 40 \\ 55 & 0 & 92 & 11 & 20 & 43 & 71 & 83 & 27 & 65 \\ 23 & 92 & 0 & 7 & 57 & 33 & 38 & 57 & 63 & 82 \\ 35 & 11 & 7 & 0 & 100 & 87 & 91 & 83 & 44 & 48 \\ 44 & 20 & 57 & 100 & 0 & 69 & 57 & 79 & 89 & 21 \\ 5 & 43 & 33 & 87 & 69 & 0 & 9 & 40 & 22 & 26 \\ 91 & 71 & 38 & 91 & 57 & 9 & 0 & 50 & 6 & 7 \\ 95 & 83 & 57 & 83 & 79 & 40 & 50 & 0 & 71 & 52 \\ 26 & 27 & 63 & 44 & 89 & 22 & 6 & 71 & 0 & 17 \\ 40 & 65 & 82 & 48 & 21 & 26 & 7 & 52 & 17 & 0 \end{pmatrix}$$

Representación y Ejemplo

- Generamos una solución aleatoria:
- Solución Xi= [1, 0, 1, 0, 0, 0, 1, 1, 1, 1]
- Wi = [34, 33, 12, 3, 43, 26, 10, 2, 48, 39], W = 145
- Peso = $\sum_{i=1}^{n=10} w_i \cdot x_i = 34+12+10+2+48+39 = 145$

¿Es válida? => Sí

- Beneficio de la solución:
- Pi= [91, 78, 22, 4, 48, 85, 46, 81, 3, 26]]
- Beneficio = $\sum_{i=1}^{n=10} P_i \cdot x_i = 91+22+46+81+3+26 = 269$

¿Es el único beneficio? => No

Representación y Ejemplo

- Solución Xi= [1, 0, 1, 0, 0, 0, 1, 1, 1, 1]
- Pi= [91, 78, 22, 4, 48, 85, 46, 81, 3, 26]
- Beneficio individual: $\sum_{i=1}^{n=10} P_i \cdot x_i = 91+22+46+81+3+26 = 269$

$$P_{ij} = \begin{pmatrix} 0 & 55 & 23 & 35 & 44 & 5 & 91 & 95 & 26 & 40 \\ 55 & 0 & 92 & 11 & 20 & 43 & 71 & 83 & 27 & 65 \\ 23 & 92 & 0 & 7 & 57 & 33 & 38 & 57 & 63 & 82 \\ 35 & 11 & 7 & 0 & 100 & 87 & 91 & 83 & 44 & 48 \\ 44 & 20 & 57 & 100 & 0 & 69 & 57 & 79 & 89 & 21 \\ 5 & 43 & 33 & 87 & 69 & 0 & 9 & 40 & 22 & 26 \\ 91 & 71 & 38 & 91 & 57 & 9 & 0 & 50 & 6 & 7 \\ 95 & 83 & 57 & 83 & 79 & 40 & 50 & 0 & 71 & 52 \\ 26 & 27 & 63 & 44 & 89 & 22 & 6 & 71 & 0 & 17 \\ 40 & 65 & 82 & 48 & 21 & 26 & 7 & 52 & 17 & 0 \end{pmatrix}$$

$$P_{ij} \cdot x_i = \begin{pmatrix} 0 & 23 & 91 & 95 & 26 & 40 \\ 23 & 0 & 38 & 57 & 63 & 82 \\ 91 & 38 & 0 & 50 & 6 & 7 \\ 95 & 57 & 50 & 0 & 71 & 52 \\ 26 & 63 & 6 & 71 & 0 & 17 \\ 40 & 82 & 7 & 52 & 17 & 0 \end{pmatrix}$$

- Beneficio por pares: 1436
- Beneficio final: 269+1426 = 1705

Solución Greedy

- Dado lo bueno que es para la mochila normal tiene mucho sentido aplicarlo para esta nueva versión.
- A diferencia de la versión original, da resultado muy competitivo pero superable, y es $O(2^n)$.
- Analizando la función objetivo podemos determinar que una buena fórmula heurística para resolver el problema es:

Escoger los objetos que aporten el mejor ratio beneficio/peso

Solución Greedy

1) Inicialmente asignamos la solución todo a cero:

$$X=[0,0,0,0,0,0,0,0,0]$$

2) Escogemos como primer elemento aquel con mejor beneficio individual: $p1\in\{0,n-1\},\forall i\in 0,n-1,i\neq p1,P[i]/w[i]\leq P[p1]/w[p1],x[p1]\Leftarrow 1$

3) El siguiente elemento p2 es aquel que maximice el siguiente ratio:

$$p2\in\{0,n-1\},\forall i\in\{0,n-1\},i\neq p2,\frac{P[i]+P[i,p1]}{w[i]}\leq\frac{P[p2]+P[p2,p1]}{w[p2]},x[p2]\Leftarrow 1$$

4) En general vamos añadiendo siempre:

$$pi\in\{0,n-1\},\forall i\in\{0,n-1\},i\neq pi,\frac{P[i]+\sum_{j\in X}P[i,j]}{w[i]}\leq\frac{P[pi]+\sum_{j\in X}P[pi,j]}{w[i]},x[pi]\Leftarrow 1$$

Solución Greedy

En general el proceso sería en cada paso calcular para cada solución:

$$r_i = \frac{p_i + \sum_{j \in X} p_{ij}}{w_i}$$

Y asignar siempre aquel objeto que maximice r^i.

Solución Greedy

```
S ← ∅;
pending ← {0, 1, ..., n − 1};
capacidad ← W;
borrar de pending objetos que pesen más que capacidad;
for i ← 1 to n do
    r_i ← p_i / w_i
end
for times ← 1 to n do
    max ← 0;
    for j ∈ pending do
        if r_j > max then
            max ← r_j;
            posi ← j;
        end
    end
    S ← S ∪ {j};
    capacidad ← capacidad − w_j;
    borrar de pending objetos que pesen más que capacidad;
    for i ← 1 to n do
        actualizar r_i;
    end
end
```

Búsquedas por Trayectorias Simples

- **Representación**: Problema de asignación: una vector $\pi=[\pi(1), ..., \pi(n)]$ en el que los valores $\pi(1), ..., \pi(n)$ representa un 0 si no se elije el objeto correspondiente, y un 1 si se escoje. Permite verificar las restricciones

- **Operador de vecino de intercambio y su entorno**: El entorno de una solución π está formado por las soluciones accesibles desde ella a través de un movimiento de intercambio

  Para cumplir las restricciones primero eliminaremos un objeto, y luego intentaremos añadir un objeto no seleccionado que queda (si no cabe se elimina otro segundo objeto, y se vuelve a intentar de nuevo).

Búsquedas por Trayectorias Simples

- El vecinario cumple las restricciones, si la solución original π es factible siempre genera una solución vecina π' factible

- Su aplicación provoca que el tamaño del entorno (en el peor caso) sea:

$$|E(\pi)| = \frac{n \cdot (n-1)}{2}$$

- Las instancias del QKP no suelen ser demasiado grandes y el cálculo factorizado del coste de una solución se realiza de forma eficiente ($O(n)$), permitiendo explorar el entorno completo

  Aún así, dicha exploración requería $O(2^n)$ por lo que es recomendable utilizar una estrategia avanzada, considerando una modalidad de lista de candidatos y seleccionado primero los movimientos más prometedores.

Búsqueda Local para el QKP

V. Cacchiani, M. Iori, A. Locatelli, y S. Martello, «Knapsack problems — An overview of recent advances. Part I: Single knapsack problems», Computers & Operations Research, vol. 143, p. 105692, jul. 2022, doi: 10.1016/j.cor.2021.105692.

- Algoritmo de búsqueda local del primer mejor: en cuanto se genera una solución vecina que mejora a la actual, se aplica el movimiento y se pasa a la siguiente iteración.
  - Se detiene la búsqueda cuando se ha explorado el vecindario completo sin obtener mejora

- Se considera una factorización para calcular el coste de π' a partir del de π considerando sólo los cambios realizados por el movimiento de intercambio.

Búsquedas por Trayectorias Simples

Function `Nuevo-Vecino(`π`)` **is**

$capacidad \leftarrow W - \sum_{i \in \pi} w_i \cdot \pi_i$

$possible \leftarrow \{0, 1, ..., n-1\} - \pi$

elegir $posi \in \pi$ aleatoriamente

$capacidad \leftarrow capacidad + w_{posi}$

eliminar de *possible* aquellos con peso mayor que la capacidad

eliminar $posi$ de π

while $possible \neq \emptyset$ **do**

elegir $posi2 \notin \pi$ aleatoriamente tal que $w_{posi2} \leq capacidad$

$capacidad \leftarrow capacidad - w_{posi2}$

eliminar de *possible* aquellos con peso mayor que la capacidad

end

end

BL-QKP: Factorización del Movimiento de Intercambio

- Sea $C(\pi)$ el coste de la solución original π. Para generar π', si el operador de vecino eliminar p y añade un elemento o $Int(\pi,p,o)$ se puede calcular de forma eficiente.
- La capacidad es fácil de calcular, añadiendo y borrando los pesos.
- El coste del beneficio de cada objeto es costoso, es el que hay que optimizar.

$$b_i = p_i + \sum_{j \in S} p_{ij}$$

Tras eliminar el objeto p es $$b_i' = p_i + \sum_{j \in S} (p_{ij} - p_{ip}) = b_i - p_i - \sum_{j \in S} p_{ip}$$

Tras añadir el objeto n es $$b_i'' = p_i + \sum_{j \in S} (p_{ij} + p_{io}) = b_i' + p_{io} + \sum_{j \in S} p_{io}$$

Resultando en

$$\Delta C(\pi, p, o) = P_o - P_p + \sum_{i \in S, i \neq p, i \neq o} (p_{io} - p_{ip}), fitness(S') = fitness(S) + \Delta C(\pi, p, o)$$

BL-QKP: Factorización del Movimiento de Intercambio

- Si $\Delta C(\pi,p,o)$ es positivo ($\Delta C(\pi,p,o)>0$), la solución vecina π' es mejor que la actual π (el QKP es un problema de maximización) y se acepta. Si no, se descarta y se genera otro vecino

- El pseudocódigo de la BL del Primer Mejor del Tema 2 de Teoría quedaría:

Repetir

$\pi' \leftarrow$ GENERA_VECINO(π_{act});

Hasta ($\Delta C(\pi,r,s)>0$) **O**
(se ha generado E(π_{act}) al completo)

- El coste $C(\pi')$ de la nueva solución vecina es: $C(\pi')=C(\pi)+\Delta C(\pi)$. Sólo es necesario calcularlo para la solución vecina aceptada

Los datos QKP

- Los datos del QKP se han obtenido de la https://cedric.cnam.fr/~soutif/QKP/QKP.html
- Poseen datos de tamaño 100, 200 y 300.
- Para cada tamaño tiene resultados con distinto nivel de densidad (en la matriz de beneficios por pares).
- Aunque de ahí se pueden obtener, los datos están en PRADO para mayor comodidad.

Los datos QKP

- El formato de los ficheros de datos es el siguiente:

Nombre fichero
Tamaño (N)
Pi
Pij (solo valores encima diagonal principal, i < j)
Línea en blanco
0
Capacidad de la mochila (W)
Vector de pesos Wi
Información textual

Más en detalle en el guión de prácticas.

Los datos QKP

- Fichero del ejemplo:

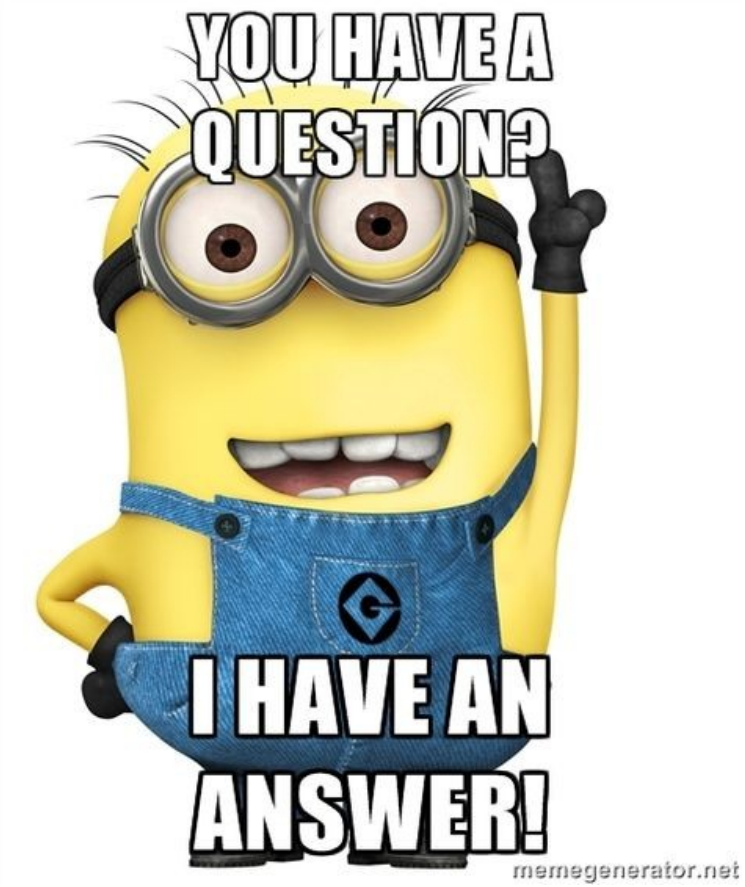
```
R_10_100_13
10
91 78 22 4 48 85 46 81 3 26
55 23 35 44 5 91 95 26 40
92 11 20 43 71 83 27 65
7 57 33 38 57 63 82
100 87 91 83 44 48
69 57 79 89 21
9 40 22 26
50 6 7
71 52
17

0
145
34 33 12 3 43 26 10 2 48 39

Comments

Density : 100.00 %
Seed : 13
```

W =145.

Wi = [34, 33, 12, 3, 43, 26, 10, 2, 48, 39]

Pi = [91, 78, 22, 4, 48, 85, 46, 81, 3, 26]

$$P_{ij} = \begin{pmatrix} 0 & 55 & 23 & 35 & 44 & 5 & 91 & 95 & 26 & 40 \\ 55 & 0 & 92 & 11 & 20 & 43 & 71 & 83 & 27 & 65 \\ 23 & 92 & 0 & 7 & 57 & 33 & 38 & 57 & 63 & 82 \\ 35 & 11 & 7 & 0 & 100 & 87 & 91 & 83 & 44 & 48 \\ 44 & 20 & 57 & 100 & 0 & 69 & 57 & 79 & 89 & 21 \\ 5 & 43 & 33 & 87 & 69 & 0 & 9 & 40 & 22 & 26 \\ 91 & 71 & 38 & 91 & 57 & 9 & 0 & 50 & 6 & 7 \\ 95 & 83 & 57 & 83 & 79 & 40 & 50 & 0 & 71 & 52 \\ 26 & 27 & 63 & 44 & 89 & 22 & 6 & 71 & 0 & 17 \\ 40 & 65 & 82 & 48 & 21 & 26 & 7 & 52 & 17 & 0 \end{pmatrix}$$


YOU HAVE A QUESTION?
I HAVE AN ANSWER!
memegenerator.net