El problema Knapsack

Múltiples vistas

Rudik Rompich/ Alejandro Pallais

Universidad del Valle de Guatemala

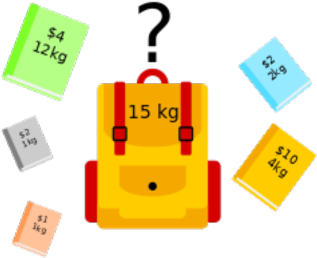
1. Introducción
2. Variaciones
3. Conclusiones
4. Referencias

£Qué es el problema Knapsack?

El problema Knapsack es un problema fundamental en la optimización combinatoria, el cual intuitivamente se describe como:

- Dado un conjunto de artículos, cada uno con un peso y un valor.
- Debemos determinar el número de cada artículo para incluir en una colección, de modo que el peso total sea menor o igual a un límite dado.
- Y el propósito del valor total sea maximizado.

Problema Knapsack en una dimensión



En una dimensión el problema knapsack se puede representar de la siguiente manera.

Problema Básico de la Mochila

Descripción:

- Mochila con capacidad máxima de peso W.
- n objetos, cada uno con un peso w_i y un valor v_i.
- Objetivo: Maximizar el valor total sin exceder W.

Formulación Matemática:

$$\text{Maximizar} \quad \sum_{i=1}^{n} v_i \cdot x_i$$

$$\text{Sujeto a} \quad \sum_{i=1}^{n} w_i \cdot x_i \leq W$$

$$x_i \in \{0, 1\} \quad \forall i$$

Comparaciones de las variaciones del problema

Variación	Descripción	Enfoque
0/1	Los objetos no se pueden dividir	Programación Dinámica
Fraccionaria	Los objetos se pueden dividir; se pueden tomar fracciones de los objetos	Algoritmo Greedy
Multi-dimensional	Múltiples restricciones de capacidad (p.ej., volumen, peso)	algoritmo genetico
Múltiples knapsacks	Más de un knapsack, cada una con su capacidad	Heurísticas/DP

Cuadro 1: Tabla comparativa de las variaciones

1. Introducción
2. Variaciones
3. Conclusiones
4. Referencias

Enfoque de Programación Dinámica

- PD aplica cuando los problemas se sobreponen y se compone de múltiples subproblemas.
- Surge como una mejora a los algoritmos de dividir y conquistar.
- PD resuelve un problema una única vez y lo guarda en una tabla. Evitando el desgaste computacional.

Enfoque de Programación Dinámica

La Programación Dinámica es un enfoque eficiente para el problema knapsack 0/1. Implica descomponer el problema en subproblemas más pequeños y resolver cada uno solo una vez.

- La PD construye una matriz $DP[0 \ldots n][0 \ldots W]$.
- $DP[i][w]$ representa el valor máximo alcanzable con los primeros i ítems y una capacidad de mochila de w.
- La solución se construye de manera iterativa.

Enfoque de Programación Dinámica

- Link a matriz de ejemplo.
- Demostración en código.

Knapsack Fraccional

Esta es una variante del problema donde se pueden seleccionar fracciones de los elementos. A diferencia del Problema 0/1, este problema permite un enfoque más flexible donde los elementos se pueden dividir en partes más pequeñas.

Formulación Matemática

Sea:

- n: Número total de objetos disponibles.
- w_i: Peso del objeto i, para $i = 1, 2, \ldots, n$.
- v_i: Valor del objeto i, para $i = 1, 2, \ldots, n$.
- W: Capacidad máxima de peso de la mochila.
- x_i: Fracción del objeto i que se incluye en la mochila, donde $0 \leq x_i \leq 1$.

Objetivo:

$$\text{Maximizar} \quad Z = \sum_{i=1}^{n} v_i \cdot x_i$$

Restricciones:

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

$$0 \leq x_i \leq 1 \quad \text{para todo } i = 1, 2, \ldots, n$$

Enfoque del Algoritmo greedy

1. Calcular la relación valor-peso para cada elemento.
2. Ordenar los elementos por esta relación en orden descendente.
3. Añadir los elementos a la mochila en este orden, tomando tanto de cada elemento como sea posible.

Multidimensional

Esta es una variante del problema donde se pueden poner múltiples restricciones a los elementos. A diferencia del Problema 0/1, este problema permite un enfoque más real donde podemos poner mas limitantes a los elementos.

Formulación Matemática

Sea:

- n: Número total de objetos disponibles.
- m: Número total de restricciones.
- w_{j_i}: Característica j del objeto i, para $i = 1, 2, \ldots, n, j = 1, 2, \ldots, m$.
- v_i: Valor del objeto i, para $i = 1, 2, \ldots, n$.
- W_j: Capacidad máxima de restricción j, para $j = 1, 2, \ldots, m$.

Objetivo:

$$\text{Maximizar} \quad Z = \sum_{i=1}^{n} v_i \cdot x_i$$

Restricciones:

$$\sum_{i=1}^{n} w_{j_i} \cdot x_i \leq W_j, \forall j \in \mathbb{Z} \cap [1, m]$$

$$x_i \in \{0, 1\} \quad \forall i$$

Enfoque del Algoritmo genético

1. Crea una población aleatoria (eligiendo objetos aleatorios)
2. Revisa si encontró la respuesta
3. Elimina las que no cumplen alguna restricción.
4. De las dos con mayor valor hace una nueva población agarrando una cantidad aleatoria de decisiones de una y el resto de la otra.
5. muta y vuelve a empezar

Ejemplo

- Demostración en código.

Conclusiones

- Programación dinámica busca descomponer el problema en problemas más pequeños.
- El algoritmo greedy busca optimizar en cada paso.
- El algoritmo genético para el problema multimensional tiene la capacidad de encontrar el resultado con un promedio de 3 iteraciones, la desventaja es que diverge un 30 % de las veces

Referencias

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to algorithms. MIT press.
- J. Doe and A. Smith, .^{o}ptimización de rutas utilizando algoritmos genéticos"Transactions on Evolutionary Computation, vol. 8, no. 4, pp. 347-360, Octubre 2010.