Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica
División de Posgrado en Ingeniería de Sistemas

NP-Completeness:

Complejidad del problema de la Mochila
(Knapsack problem)

Ing. Jonás Velasco Álvarez
Optimización Combinatoria

Agenda:

- *Definición del problema de la mochila (KP)*
- *Complejidad computacional (NP-Completo)*
- *Algoritmos pseudo-polinomiales*
- *Problemas fuertemente NP-Completos*
- *Algoritmos de Aproximación (PTAS y FPTAS)*
- *Algunas aplicaciones del KP*
- *Algo sobre mi tesis*

Knapsack problem: *"Empacado de objetos dentro de la mochila"*

El problema de la mochila es definido formalmente como:

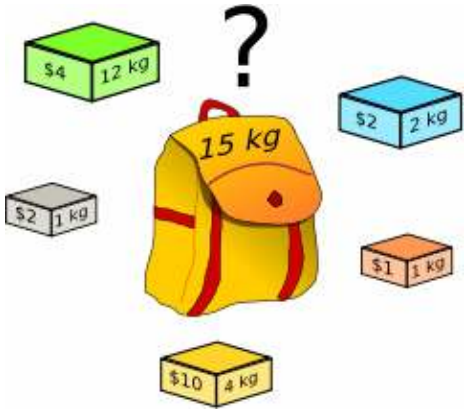
Se tiene una determinada instancia de *KP* con un conjunto de objetos *N*, que consiste de *n* objetos *j* con ganancia *pj* y peso *wj*, y una capacidad *c*. (Usualmente, los valores toman números enteros positivos).

El objetivo es seleccionar un subconjunto de *N* tal que la ganancia total de esos objetos seleccionados es maximizado y el total de los pesos no excede a c.

$$(kp) \quad \text{maximize} \sum_{j=1}^{n} pj\ xj$$

$$\text{subject to} \sum_{j=1}^{n} wj\ xj \leq c$$

$$xj \in \{0,1\}, \quad j = 1,\ldots\ldots,n$$



Teorema 9.10: Knapsack es NP-completo

Lo que se muestra por un problema de conjuntos *EXACT COVER BY 3-SET* que es un caso especial del *SET COVERING* y son generalizaciones del *TRIPARTITE MATCHING*.

También el *KNAPSACK* es un caso especial del SUBSET SUM y que viene por la reducción de 3-SAT, VERTEX COVER.

Reducción de SUBSET SUM:

La entrada de la suma de subconjuntos es dado un conjunto de números enteros $\{a_1, a_2, a_3, \ldots\ldots, a_n\}$ y un numero k, la pregunta es ¿hay un subconjunto de los numero a añadir tal que la suma sea exactamente k?

Esto es como un problema de la mochila mediante la introducción de n objetos en la mochila, cuando el peso y la ganancia tienen un mismo valor a_j.

Ahora, la suma de subconjuntos tiene respuesta sí, si y solo si el problema de la mochila tiene una solución con valor igual a k. (Dado que el peso y la ganancia son iguales , el knapsack puede lograr un valor k si pueden ser añadidos los objetos por completo).

En el *KP* se tiene como instancia una lista de *N* diferentes objetos $j \in \Phi$ que consiste de *n* objetos *j* y cada objeto tiene una ganancia *pj* y peso *wj*.

La pregunta es qué conjunto $M \subseteq \Phi$ de objetos debería uno elegir para tener un valor total por lo menos *k* si se tiene una mochila que solamente soporta peso hasta un cierto limite superior Ψ . entonces con la restricción:

$$\psi \geq \sum_{j \in M} w(j)$$

Se aspira maximizar la utilidad total:

$$\sum_{j \in M} p(j) \geq k$$

El algoritmo

Definimos variables auxiliares V (w, i) que es el valor total máximo posible seleccionando algunos entre los primeros i artículos tal que su peso total es exactamente w. Cada uno de los V (w, i) con w = 1, . . . , Ψ y i = 1, . . . ,N se puede calcular a través de la ecuación recursiva siguiente:

$$V(w, i+1) = \text{máx}\{V(w, i), v_{i+1} + V(w - w_{i+1}, i)\}$$

donde V (w, 0) = 0 para todo w y V (w, i) = $-\infty$ si w ≤ 0.

Ejemplo

M={1,2,3}, v_j={2, 3, 5}, w_j={3, 5, 7}

$$V(w, i+1) = \text{máx}\{V(w, i), v_{i+1} + V(w - w_{i+1}, i)\}$$

w_j=3
V(3, 1)= max { V(3, 0), 2 + (3 - 3, 0) }
V(3, 2)= max { V(3, 1), 3 + (3 - 5, 1) }
V(3, 3)= max { V(3, 2), 5 + (3 - 7, 2) }

w_j=5
V(5, 1)= max { V(5, 0), 2 + (5 - 3, 0) }
V(5, 2)= max { V(5, 1), 3 + (5 - 5, 1) }
V(5, 3)= max { V(5, 2), 5 + (5 - 7, 2) }

i \ w	0	3	5	7	8	10	12	15
1	0	2	2	2	2	2	2	2
2	0	2	3	3	5	5	5	5
3	0	2	3	5	5	7	8	10

Entonces, podemos calcular en tiempo constante un valor de V (w, i) conociendo algunos otros y en total son N Ψ elementos.

Este algoritmo tiene tiempo de ejecución O(N $\cdot\Psi$).

La respuesta del problema de decisión es “sí” únicamente en el caso que algún valor V (w, i) sea mayor o igual a k.

Entonces, ¿cómo puede ser esto un problema **NP**-completo?

Para pertenecer a P, necesitaría tener un algoritmo polinomial *en el tamaño de la instancia*, que es más como $N \cdot \log \Psi$ y así menor que el parámetro obtenido $N \cdot \Psi$ (tomando en cuenta que = $\Psi = 2^{\log \Psi}$).

Tal algoritmos donde la cota de tiempo de ejecución es polinomial en los enteros de la entrada y no sus logaritmos se llama un algoritmo *pseudo-polinomial*.

Un problema es *fuertemente NP-completo* si permanece *NP-completo* incluso en el caso que toda instancia de tamaño *n* está restringida a contener enteros de tamaño máximo *p(n)* para algún polinomial *p*.

Un problema fuertemente *NP-completo* no puede tener algoritmos pseudo-polinomiales salvo que si aplica que P sea igual a NP.

Los problemas SAT, MAXCUT, TSPD y HAMILTON PATH, por ejemplo, son fuertemente *NP-completos*, pero KNAPSACK no lo es.

En situaciones donde todos los algoritmos conocidos son lentos, vale la pena considerar la posibilidad de usar una solución aproximada, o sea, una solución que tiene un valor de la función objetivo cerca del valor optimo, pero no necesariamente el óptimo mismo.

En muchos casos es posible llegar a una solución aproximada muy rápidamente mientras encontrar la solución óptima puede ser imposiblemente lento.

Un algoritmo de aproximación puede ser determinista o no determinista (Algoritmo aleatorizado).

Un algoritmo de aproximación bien diseñado cuenta con un análisis formal que muestra que la diferencia entre su solución y la solución óptima es de un factor constante (factor de aproximación).

Se dice que Π es un problema de optimización con una función objetivo f_Π. Decimos que un algoritmo A es un esquema de aproximación para Π si la entrada (I, ϵ), donde I es una instancia de Π y $\epsilon > 0$ es el parámetro de error, su salida da como solución *s* talque:

$$f_\Pi(I, s) \leq (1 + \epsilon) \cdot \mathrm{OPT}$$ (minimización)

$$f_\Pi(I, s) \geq (1 - \epsilon) \cdot \mathrm{OPT}$$ (maximización)

Si existe un método sistemático para aproximar la solución a factores arbitrarios, ese método se llama un *esquema de aproximación (de tiempo polinomial)*.

PTAS (Polynomial time approximation scheme):

Se dice si para cada fijo $\epsilon > 0$, su tiempo de corrida es acotado por un polinomial en el tamaño de Instancia I.

FPTAS (Fully polynomial time approximation scheme):

Se requiere que en el tiempo de corrida del problema A es acotado por un polinomial en el tamaño de la instancia I y $1/\epsilon,$

FPTAS para el KP.

Notar que si los valores de ganancias de los objetos son números pequeños, ellos son acotados por un polinomial en *n*. Entonces el tiempo de corrida debería ser acotado por un polinomial en tamaño de instancia.

Algoritmo (FPTAS para el KP):

1.- dado $\epsilon > 0$, $K = \frac{\epsilon P}{n}$.

2.- Para cada objeto i, define una ganancia $p'(a_i) = \left\lfloor \frac{p(a_i)}{K} \right\rfloor$.

3.- Con estas ganancias de objetos usamos el algoritmo pseudo-polinomial y encontrar el conjunto con mayor ganancia S´.

4.- salida S´.

Tarea: ejecutar el algoritmo con la instancia de ejemplo, $\varepsilon = 0.20$.

Lema 8.3: El conjunto S´ salida del algoritmo satisface que:

$$P(S') \geq (1 - \epsilon) \cdot \mathrm{OPT}$$

El tiempo de corrida del algoritmo es $O(n^2 \lfloor \frac{P}{K} \rfloor)$ = $O(n^2 \lfloor \frac{n}{\epsilon} \rfloor)$ la cual es polinomial en *n* y $1/\epsilon$, .

Mejoras al algoritmo:

Cuando obtenemos unas solución aproximada de manera cualquiera a un problema de optimización, podemos intentar mejorarla por búsqueda local.

Solo aplicamos operaciones pequeñas y rápidamente realizadas para causar cambios pequeños en la solución así que se mantenga factibles y pueda ser que se mejore.

Tarea (mas puntos):
A la solución (FPTAS) anterior implementar un algoritmo de búsqueda local.

Este problema tiene numerosas aplicaciones tales como:

• La denominada *Cutting Stock*, en donde hay que cortar una plancha de acero en diferentes piezas.

• Determinar los artículos que puede almacenar un depósito para maximizar su valor total.

• Maximizar el beneficio en asignación de inversiones cuando sólo hay una restricción.

Interés práctico:

• Problemas en los cuales una inversión es proporcional a la suma de lo invertido con una variación pequeña.

• Cuando se tiene una inversión que genera una mayor ganancia pero también tiene un mayor costo.

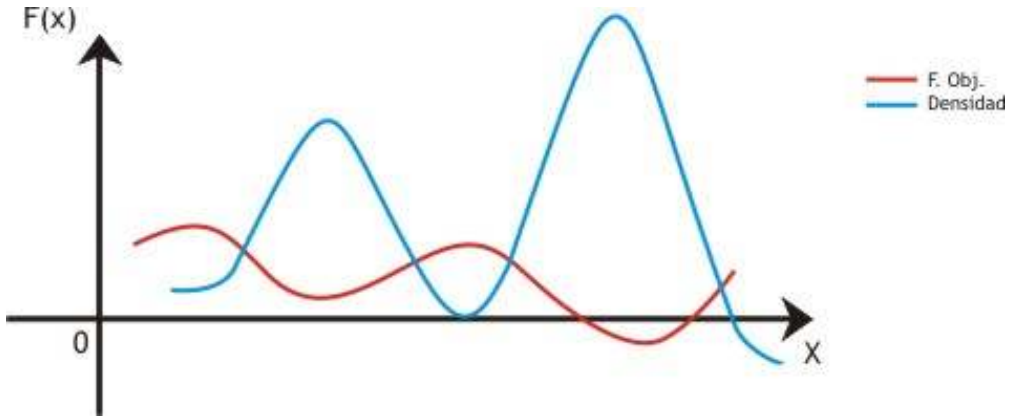
SFPL

Adaptación

Diversificación (Ruido aditivo)

$$\dot{x}_n = -\frac{\partial V}{\partial x_n} + \varepsilon(t),$$

El proceso difusivo tiene una densidad de probabilidad asociada:



Los máximos en la densidad está asociados con los mínimos del problema.

Referencias:

- Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, Reading, MA, USA, 1994.
- Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag GmbH, Berlin, Germany, 2001.
- Hans Kellerer, Ulrich Pferschy, David Pisinger. *Knapsack Problem*. Springer-Verlag GmbH, Berlin, Germany, 2004.