Problém batohu

Zdeněk Hanzálek
hanzalek@fel.cvut.cz

ČVUT FEL Katedra řídicí techniky

5. dubna 2011

Formulace problému

Problém batohu (Knapsack problem)

- **Instance:** Nezáporná celá čísla $n, c_1, \ldots, c_n, w_1, \ldots, w_n, W$, kde n je počet předmětů, $c_1, \ldots, c_n$ jsou ceny předmětů, $w_1, \ldots, w_n$ jsou hmotnosti předmětů a W je nosnost batohu.
- **Cíl:** Nalézt podmnožinu $S \subseteq \{1, \ldots, n\}$ takovou, že $\sum_{j \in S} w_j \leq W$ a $\sum_{j \in S} c_j$ je maximální.

- Jde o jeden z "nejjednodušších" NP-obtížný problém.
- Též se někdy nazývá **$\mathbf{0/1}$ Knapsack problem**.

Relaxace na nedělitelnost předmětů

Fractional Knapsack problem

- **Instance:** Nezáporná celá čísla $n, c_1, \ldots, c_n, w_1, \ldots, w_n, W$, kde n je počet předmětů, $c_1, \ldots, c_n$ jsou ceny předmětů, $w_1, \ldots, w_n$ jsou hmotnosti předmětů a W je maximální hmotnost batohu.
- **Cíl:** Nalézt racionální čísla $x_1, \ldots, x_j, \ldots, x_n$ taková, že $0 \leq x_j \leq 1$ a $\sum_{j=1}^{n} x_j \cdot w_j \leq W$ a $\sum_{j=1}^{n} x_j \cdot c_j$ je maximální.

- díky možnosti dělit předměty (reálné proměnné x_j) je tento problém řešitelný v polynomiálním čase

Řešení Fractional Knapsack problému - Dantzing [1957]

- pokud platí $\sum_{j=1}^{n} w_j > W$ (opačný případ je triviální)
- setřiď a přeindexuj předměty podle relativní ceny, neboli aby platilo $\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \ldots \geq \frac{c_n}{w_n}$
- v setříděné sekvenci nalezni index prvku, který se do batohu nevejde, neboli $h := min \left\{ j \in \{1, \ldots, n\} : \sum_{i=1}^{j} w_i > W \right\}$
- pro optimální řešení je oddělena ta část h-tého prvku, která se do batohu vejde:
  - $x_j := 1$ pro $j = 1, \ldots, h-1$
  - $x_h := \frac{W - \sum_{i=1}^{h-1} w_i}{w_h}$
  - $x_j := 0$ pro $j = h+1, \ldots, n$
- třídění prvků zabere $O(nlogn)$ času, výpočet h se provede jednoduchým průchodem v $O(n)$, takže tento postup vyřeší **Fractional Knapsack problém** v $O(nlogn)$
- existuje i algoritmus ([1] str. 440), který dokáže tento problém vyřešit v $O(n)$ převodem na **Weighted Median problem**

2-aproximační algoritmus pro Knapsack

r-aproximační algoritmus pro maximalizaci

Algoritmus A pro problém s maximalizací kriteriální funkce J se nazývá r-aproximační pokud existuje číslo $r \geq 1$ takové, že $J^A(I) \geq \frac{1}{r} J^*(I)$ pro všechny instance I daného problému.

Věta

Nechť $n, c_1, \ldots, c_n, w_1, \ldots, w_n, W, h$ jsou nezáporná celá čísla, pro která platí:

- $w_j \leq W$ pro $j = 1, \ldots, n$
- $\sum_{i=1}^{n} w_i > W$
- $\frac{c_1}{w_1} \geq \frac{c_2}{w_2} \geq \ldots \geq \frac{c_n}{w_n}$
- $h = min \left\{ j \in \{1, \ldots, n\} : \sum_{i=1}^{j} w_i > W \right\}$

Potom výběr lepšího ze dvou řešení $\{1, \ldots, h-1\}$ a $\{h\}$ je **2-aproximační algoritmus pro Knapsack problém** s časovou náročností $O(n)$.

2-aproximační algoritmus pro Knapsack

Důkaz:

- V libovolné instanci **Knapsack problému** mohou být vyřazeny prvky jejichž hmotnost je větší než nosnost batohu.
- Pokud je $\sum_{i=1}^{n} w_i \leq W$, potom celá množina předmětů tvoří optimální řešení.
- Jelikož $\sum_{i=1}^{h} c_i$ je horní mez optima, tak lepší ze dvou řešení $\{1, \ldots, h-1\}$ a $\{h\}$ dosahuje více než poloviny optima.

Poznámka k aproximačním algoritmům:

- Aproximační algoritmy dávají garanci, že i v nejhorším případě bude hodnota kriteriální funkce nalezeného řešení v určité proporci k optimu. Četnost výskytu tohoto nejhoršího případu aproximační algoritmus nestuduje.
- Někdy namísto **výkonnostního poměru** r (asymptotic performance ratio) uvádíme **poměrnou odchylku od optima** ϵ, přitom platí $r = 1 + \epsilon$.
- Uvádět absolutní chybu nemá smysl. Proč?

Dynamické programování (celočíselné ceny) pro Knapsack

Vstup: Ceny $c_1, \dots, c_n \in \mathbb{Z}_0^+$, váhy $w_1, \dots, w_n, W \in \mathbb{R}_0^+$.
Výstup: $S \subseteq \{1, \dots, n\}$ taková, že $\sum_{j \in S} w_j \leq W$ a $\sum_{j \in S} c_j$ je maximální.
Nechť C je libovolná horní mez optima, např. $C = \sum_{j=1}^{n} c_j$;

$x_0^0 := 0$; $x_k^0 := \infty$ pro $k = 1, \dots, C$;
for $j := 1$ **to** n **do**
  for $k := 0$ **to** C **do** $s_k^j := 0$; $x_k^j := x_k^{j-1}$;
  for $k := c_j$ **to** C **do**
    if $x_{k-c_j}^{j-1} + w_j \leq min\{W, x_k^{j-1}\}$ **then**
      $s_k^j := 1$; $x_k^j := x_{k-c_j}^{j-1} + w_j$;
    end
  end
end
$i := max\{k \in \{0, \dots, C\} : x_k^n < \infty\}$; $S := \emptyset$;
for $j := n$ **downto** 1 **do**
  if $s_i^j = 1$ **then** $S := S \cup \{j\}$; $i := i - c_j$;
end

Dynamické programování (celočíselné ceny) pro Knapsack

- Pseudopolynomiální algoritmus s časovou náročností $O(nC)$.
- Proměnná x_k^j představuje **minimální hmotnost o ceně k, jakou lze dosáhnout výběrem předmětů z množiny** $\{1, \ldots, j\}$
- (*) Předmět j je dán do výběru z předmětů $1, \ldots, j$, pokud pro danou cenu k dosáhne tento výběr **nižší nebo stejné hmotnosti než výběr z předmětů** $1, \ldots, j-1$**.** Algoritmus počítá tyto hodnoty s využitím rekurentní rovnice:

$$x_k^j = \begin{cases} x_{k-c_j}^{j-1} + w_j & \text{pokud předmět } j \text{ byl dán do výběru;} \\ x_k^{j-1} & \text{pokud předmět } j \text{ nebyl dán do výběru.} \end{cases}$$

- Do proměnné s_k^j zapisujeme, který z těchto dvou případů nastal. Použije se pro rekonstrukci výběru.

Příklad řešený na tabuli (celočíselné ceny):
$n = 4, w = (21, 35, 52, 17), c = (10, 20, 30, 10), W = 100$

Dynamické programování pro Knapsack

Dynamické programování obecně

- pseudopolynomiální algoritmus
- stavový prostor je konstruován díky celočíselným vstupům
- stavový prostor není strom, jsou v něm "diamanty" (též se nazývá "overlaping structures", neboli v tomto místě st.prostoru si ze dvou možných řešení ponecháme pouze to výhodnější - viz (*)) tím nedochází k exponenciálnímu růstu jeho velikosti

Pokud jsou váhy celočíselné, můžeme problém řešit pomocí dynamického programování, kde pro danou váhu vybereme řešení s **větší** cenou (*).

Příklad (celočíselné váhy) na tabuli pomocí nejkratších cest a pomocí dynamického programování.

Redukce složitosti zaokrouhlením vstupních dat

Časová náročnost algoritmu Dynamického programování pro Knapsack je závislá na C.

Myšlenka

Vydělíme všechny ceny $c_1, \ldots, c_n$ číslem 2 a zaokrouhlíme je dolů.

To zrychlí algoritmus, ale může to vést na suboptimální řešení.
Takto můžeme volit mezi mírou rychlosti a optimality.
Zavedením

$$\bar{c}_j := \left\lfloor \frac{c_j}{t} \right\rfloor \; pro \; j = 1, \ldots, n$$

se časová náročnost algoritmu Dynamického programování pro Knapsack sníží t-krát.

Aproximační schéma pro Knapsack

Vstup: Nezáp. celá čísla $n, c_1, \ldots, c_n, w_1, \ldots, w_n, W$. Číslo $\epsilon > 0$.
Výstup: Podmnožina $S \subseteq \{1, \ldots, n\}$ taková, že $\sum_{j \in S} w_j \leq W$ a
$\sum_{j \in S} c_j \geq \frac{1}{1+\epsilon} \sum_{j \in S'} c_j$ pro všechny $S' \subseteq \{1, \ldots, n\}$ splňující $\sum_{j \in S'} w_j \leq W$.

1. proveď **2-aproximační algoritmus pro Knapsack**;
   řešení označ S_1 o ceně $c(S_1) = \sum_{j \in S_1} c_j$;
2. $t := max\{1, \frac{\epsilon c(S_1)}{n}\}$;
   $\bar{c}_j := \left\lfloor \frac{c_j}{t} \right\rfloor$ *pro* $j = 1, \ldots, n$;
3. proveď **algoritmus Dynamického programování pro Knapsack** na instanci $(n, \bar{c}_1, \ldots, \bar{c}_n, w_1, \ldots, w_n, W)$ s horní mezí $C := \frac{2c(S_1)}{t}$;
   řešení označ S_2 o ceně $c(S_2) = \sum_{j \in S_2} c_j$;
4. **if** $c(S_1) > c(S_2)$ **then** $S := S_1$ **else** $S := S_2$

Aproximační schéma pro Knapsack

- Důkaz toho, že volba dělitele t v kroku 2 ($t := max\{1, \frac{\epsilon c(S_1)}{n}\}$) vede na $(1+\epsilon)$-aproximační algoritmus lze nalézt v [1].
- Časová náročnost je $O(nC) = O(n\frac{c(S_1)}{t}) = O(n\frac{c(S_1)n}{\epsilon c(S_1)}) = O(n^2 \cdot \frac{1}{\epsilon})$.
- **Knapsack** je jeden z mála problémů pro něž existuje aproximační algoritmus s "libovolně malou" poměrnou odchylkou od optima ϵ. Pro určité malé $\epsilon = \frac{n}{c(S_1)}$ je $t = 1$ a jde v podstatě o **algoritmus Dynamického programování pro Knapsack** s horní mezí z **2-aproximační algoritmus pro Knapsack**.

Knapsack - Shnutí

- Jeden z "nejjednodušších" NP-obtížných problémů
- Základ pro řadu dalších optimalizačních problémů (bin packing, container loading, 1-D, 2-D, 3-D cutting problem)

Literatura

B. H. Korte and Jens Vygen.
Combinatorial Optimization: Theory and Algorithms.
Springer, fourth edition, 2008.