Advanced Algorithm: Note #1

February 21, 2017

Prof. Ho-Lin Chen

電機三 B03901018 楊程皓

Algorithm

input → Algorithm → output (optimal solution)
algorithm: fixed amount of resource requirement.

Approximation Algorithm

input → Approximation Algorithm → output (near-optimal solution)
if cost(OPT) $\leq$ cost(alg. output) $\leq$ k $\times$ cost(OPT)
$\Rightarrow$ k-approx. algorithm

Randomized Algorithm

input → Randomized Algorithm (random bit string 0001101 ...) → output

1. Las Vegas Alg.
   Always produces desired (optimal) output.
   Different resource requirement.
   ex: Quicksort

2. Monte Carlo Alg.
   Deterministic time/memory ...
   Output may be incorrect/not optimal.

Online Algorithm

input → Online Algorithm → output
(arriving one by one)

Streaming Algorithm

10, → Streaming Alg.
2,
6,
8,
...

Knapsack Problem

n items, weight w_i, value v_i
Weight limit W
Goal: Select set of items S,
s.t. $\sum_S v_i$ is maximized, subject to $\sum_S w_i \leq W$

1. fractional Knapsack
   (items can be split)
   Algorithm for fractional Knapsack Sort items s.t.
   $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_n}{w_n}$ Choose items 1,2,3, …, i until i+1 doesn't fit.
   Total value = OPT = $v_1 + v_2 + \ldots v_i + c \cdot v_{i+1}$

2. 0-1 Knapsack
   (items cannot be split)
   2-Approx. Alg. for 0-1 Knapsack
   Sort items s.t.
   $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_n}{w_n}$ Choose items 1,2,3, …, i until i+1 doesn't fit.

   Pick max$(v_1 + v_2 + \ldots + v_i,\ v_{i+1})$ (Assume $w_i \leq W, \forall i$)
   $$\geq \frac{(v_1 + v_2 + \ldots + v_i) + (v_{i+1})}{2}$$
   $$\geq \frac{v_1 + v_2 + \ldots + v_i + c \cdot v_{i+1}}{2} = \frac{OPT for fractioinal Knapsack}{2}$$
   $$\geq \frac{OPT}{2}$$

Recap: Complexity classes

(decision problems only.)
A problem is in P iff *exists* polynomial-time alg. which solves it.
A problem is in NP iff *exists* polynomial-time verifier.
A problem is in NP-hard iff $\begin{cases} \forall\ Y \in NP, \\ Y(poly-time\ reduction) \rightarrow X \end{cases}$.
NP-complete = NP ∩ NP-hard

If we know: $\begin{cases} Z is NP-hard \\ Z(poly-time\ reduction) \rightarrow X \end{cases}$, then X is NP-hard.

Exact Set Cover (NP-hard)

Given S=$x_1, \ldots, x_n$
m subsets $S_1, S_2, \ldots, S_m$
Q: Is there a collection of subsets $S_{i_1}, S_{i_2}, \ldots, S_{i_k}$, s.t. every element $x_i \in S$ is contained in exactly one subset.

0-1 Knapsack

Given n items with weights w_i, value v_i and W, V.
Is there a set of items S s.t. $\sum_S w_i \leq W$, $\sum_S v_i \geq V$

Reduction

$S_i \rightarrow$ item i, $v_i = w_i =$ n digit number (base m+1)
$x_j \in S_i \Leftrightarrow$ the j-th digit=1
ex: $x_1, x_2, x_4 \rightarrow 110100 \ldots 0$

W=V = 11 ... 1 (n digits)

Dynamic Programming Algorithm for 0-1 Knapsack

A[i,v]: min weight set chosen from items 1,2, ..., i s.t. $\sum value \geq v$

A[0,v]=$\begin{cases} 0, if\ v \leq 0 \\ \infty, if\ v > 0 \end{cases}$

for i=1 to n
    for v=0 to $\sum v_i$
        A[i,v]=min(A[i-1,v], A[i-1,v-v_i] + w_i)

Find v^* s.t.
A[n, v^*] $\leq W$
A[n, $v^* + 1$] $\geq W$
$\Rightarrow$ total time: $\Theta(nv^*)$

input: v_i, w_i, W
v_i: 5,8,10
binary: 101, 1000, 1010
unary: 11111, 1111111, 1 ... 1

Pseudo-polynomial time algorithm

Running is polynomial in the input size under unary representation (but not binary)

Strongly NP-hard

NP-hard even when all numerical values are poly(size of input)

0-1 Knapsack

Alg. Given any $\epsilon > 0$
Want: Find (1+ϵ)-approx. alg.
Let $b = \frac{\epsilon}{2n} max_i v_i$

Consider 0-1 $Kn\hat{a}psack$

$\hat{w_i} = w_i$, $\hat{W} = W$
$\hat{v_i} = \lceil \frac{v_i}{b} \rceil \times b$
Run dynamic programming on $Kn\hat{a}psack$

Running time = $\Theta(n \times \frac{v^*}{b})$
$\frac{v^*}{b} \leq \frac{n \cdot max_i v_i}{\frac{\epsilon}{2n} max_i v_i} = \frac{2n^2}{\epsilon}$
Time = $O(n^3 \cdot \epsilon^{-1})$

OPT: original 0-1 Knapsack
$\hat{OPT}$: algorithm output.
value(OPT) in 0-1 Knapsack
$\leq$ value(OPT) in $\hat{Knapsack}$
$\leq$ value($\hat{OPT}$) in $\hat{Knapsack}$
$\leq$ value($\hat{OPT}$) in $Knapsack$ + nb

In the original 0-1 Knapsack
value(OPT) $\leq$ value($\hat{OPT}$) + nb
= value($\hat{OPT}$) + $n \cdot \frac{\epsilon}{2n} max\ v_i$
$\leq$ value($\hat{OPT}$) + $\frac{\epsilon}{2}$ value(OPT)
$\Rightarrow$ value($\hat{OPT}$) $\geq$ (1-$\frac{\epsilon}{2}$) value(OPT)
$\Rightarrow$ value($\hat{OPT}$) $\geq$ $\frac{1}{1+\epsilon}$ value(OPT)

Def: PTAS

polynomial-time approximation scheme

Def: FPTAS

Given any $\epsilon > 0$ can approximate within a factor of $1 + \epsilon$ in time poly(n,$\frac{1}{\epsilon}$)