29 April 2024

$(1-\varepsilon)$ - Approximate Knapsack.

Plan.

* Knapsack Problem
* Announcements
* Approximation Scheme
* Dynamic Program.

Knapsack Problem

* Given:
  - list of n items.
    - ↳ Each item $i \in \{1, \ldots, n\}$ has weight w_i & value v_i
  - weight bound W

* Find: Subset $S \subseteq \{1, \ldots, n\}$ maximizing

$$\sum_{i \in S} v_i$$

subject to

$$\sum_{i \in S} w_i \leq W.$$

5	7	4	2	11	3
4	3	10	6	15	4

$W = 16$

5	7	4	2	11	3
4	3	10	6	15	4

$W = 16$

$S = \{5\}$

$V_s = 11$

$W_s = 15$

5	7	4	2	11	3
4	3	10	6	15	4

$W = 16$

$S = \{5\}$ $\quad V_s = 11$

$W_s = 15$

$S = \{1, 2, 6\}$ $\quad V_S = 15$

$W_s = 11$

Two facts about Knapsack

(1) Knapsack is NP-Hard

(2) Knapsack is solvable by dynamic programming in time $O(n^2 v^*)$

where $v^* = \max_{i \in \{1, \dots, n\}} v_i$

Two facts about Knapsack

(1) Knapsack is NP-Hard

(2) Knapsack is solvable by dynamic programming in time $O(n^2 v^*)$

where $v^* = \max_{i \in \{1,\dots,n\}} v_i$

Knapsack is "weakly" NP-Hard

Weak NP-Hardness

* Hardness arises due to solving problem exactly on <u>large</u> value.

Weak NP - Hardness

* Hardness arises due to solving problem exactly on large value.

Idea.

* Reduce Knapsack on Large values to Knapsack on small values

Weak NP-Hardness

* Hardness arises due to solving problem exactly on large value.

Idea.

* Reduce Knapsack on Large values to Knapsack on small values

* Solution on "noisy instance" w/ small values approximates optimal solution on original instance

Announcements

* HW9 out now.
* Fall 2024 TA Application due tonight!
* Student Survey about CIS courses
  WIC/URMC

DP Exact Algorithm.

* pseudo-polynomial time

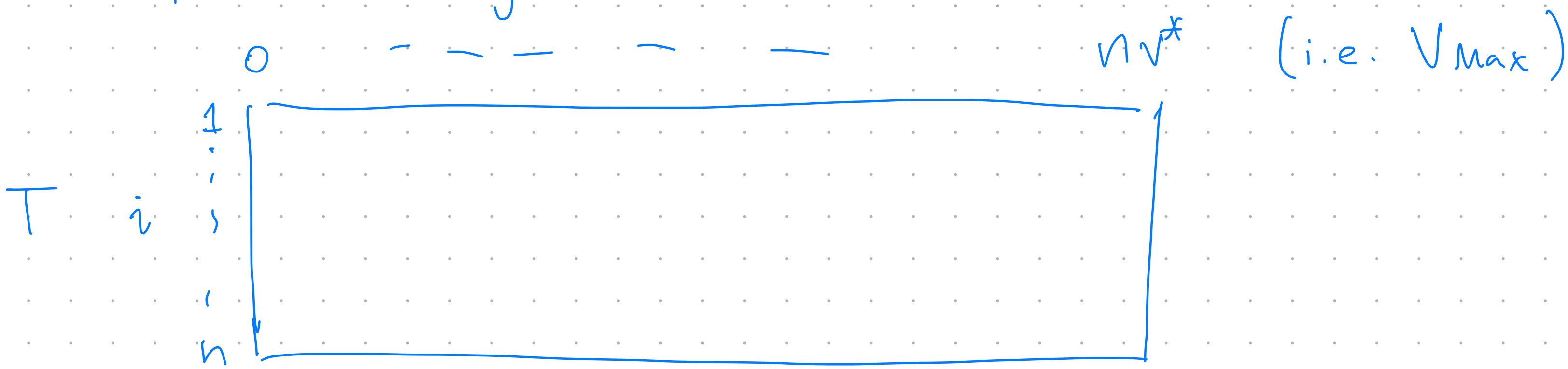
Dynamic Program

DP Exact Algorithm.

* pseudo-polynomial time

$v^* = \max_{i \in \{1, \dots, n\}} v_i$

Dynamic Program



↳ n entries per <u>value</u>

$\boxed{O(n^2 v^*) \text{ RT}}$

DP Exact Algorithm.

* pseudo-polynomial time

$v^* = \max_{i \in \{1..n\}} v_i$ (i.e. V_{max})

Dynamic Program

↳ n entries per value

$$\boxed{O(n^2 v^*) \text{ RT}}$$

* Reduction. solve a "coarse" instance using exact DP

Reduce Knapsack to "Coarse"-Knapsack.

$$\max_{i \in \{1,\dots,n\}} \sum_{i \in S} v_i \quad \longrightarrow \quad \max_{i \in \{1,\dots,n\}} \sum_{i \in S} \tilde{v}_i$$

s.t. $\sum_{i \in S} w_i \le W$ s.t. $\sum_{i \in S} w_i \le W$

Reduce Knapsack to "Coarse"-Knapsack.

$$\max_{i \in \{1,\dots,n\}} \sum_{i \in S} v_i \quad \text{s.t.} \quad \sum_{i \in S} w_i \leq W \longrightarrow \max_{i \in \{1,\dots,n\}} \sum_{i \in S} \tilde{v}_i \quad \text{s.t.} \quad \sum_{i \in S} w_i \leq W$$

Properties

* $\tilde{V} = \max_{i \in \{1,\dots,n\}} \tilde{v}_i \ll V^* = \max_{i \in \{1,\dots,n\}} v_i$

Reduce Knapsack to "Coarse"-Knapsack.

$$\max_{i \in \{1,\dots,n\}} \sum_{i \in S} v_i \qquad \longrightarrow \qquad \max_{i \in \{1,\dots,n\}} \sum_{i \in S} \tilde{v}_i$$

s.t. $\sum_{i \in S} w_i \leq W$ $\qquad$ s.t. $\sum_{i \in S} w_i \leq W$

Properties

* $\tilde{V} = \max_{i \in \{1,\dots,n\}} \tilde{v}_i \ll V^* = \max_{i \in \{1,\dots,n\}} v_i$

* $\exists\, b \in \mathbb{R}$ s.t. $\forall\, S \subseteq \{1,\dots,n\}$ $\quad b \cdot \sum_{i \in S} \tilde{v}_i \approx \sum_{i \in S} v_i$

<u>Reduction</u>

$$\tilde{v}_i = \left\lfloor \frac{v_i}{b} \right\rfloor \quad \forall i \in \{1 \ldots n\}$$

Reduction

$$\tilde{v}_i = \left\lceil \frac{v_i}{b} \right\rceil \qquad \forall i \in \{1 \ldots n\}$$

Approximation

$$v_i \underset{(\ell)}{\leq} b \cdot \tilde{v}_i = b \cdot \left\lceil \frac{v_i}{b} \right\rceil \underset{(u)}{\leq} v_i + b$$

Leave weights & W the same.

Reduction

$$\tilde{v}_i = \left\lceil \frac{v_i}{b} \right\rceil \qquad \forall i \in \{1 \ldots n\}$$

Approximation

$$v_i \underset{(l)}{\leq} b \cdot \tilde{v}_i = b \cdot \left\lceil \frac{v_i}{b} \right\rceil \underset{(u)}{\leq} v_i + b$$

Claim. Suppose S is maximum b-"coarse" solution, and S^* is maximum to original Knapsack instance.

Then,

$$\sum_{i \in S} v_i \geq \sum_{j \in S^*} v_j - bn$$

Claim. Suppose S is maximum b-"coarse" solution, and S^* is maximum to original Knapsack instance.

Then,
$$\sum_{i \in S} v_i \geq \sum_{j \in S^*} v_j - bn$$

Pf.

1
$$\sum_{i \in S^*} \tilde{v}_i \leq \sum_{i \in S} \tilde{v}_i \qquad // \text{ by maximum } S.$$

Claim. Suppose S is maximum b-"coarse" solution, and S^* is maximum to original Knapsack instance.

Then,

$$\boxed{\sum_{i \in S} v_i} \geq \boxed{\sum_{j \in S^*} v_j} - bn$$

Pf.

(i) $$\sum_{i \in S^*} \tilde{v}_i \leq \sum_{i \in S} \tilde{v}_i \qquad // \text{ by maximum } S.$$

By the inequalities established:

$$\boxed{\sum_{i \in S^*} v_i} \leq b \cdot \sum_{i \in S^*} \tilde{v}_i \leq b \cdot \sum_{i \in S} \tilde{v}_i \leq \boxed{\sum_{i \in S} v_i} + b|S|$$

by (l) by (i) by (ii)

So. if we can solve the b-coarse Knapsack problem, we can obtain approximation.

$$V_{approx} = \sum_{i \in S} v_i$$

$$V_{exact} = \sum_{j \in S^*} v_j$$

So. if we can solve the b-coarse Knapsack problem, we can obtain approximation.

$$V_{approx} = \sum_{i \in S} v_i$$

$$V_{exact} = \sum_{j \in S^*} v_j$$

Approximation Ratio

$$\frac{V_{approx}}{V_{exact}} \geq \frac{V_{exact} - bn}{V_{exact}} = 1 - \frac{bn}{V_{exact}}$$

So. if we can solve the b-coarse Knapsack problem, we can obtain approximation.

$$V_{approx} = \sum_{i \in S} v_i$$

$$V_{exact} = \sum_{j \in S^*} v_j$$

Approximation Ratio

$$\frac{V_{approx}}{V_{exact}} \geq \frac{V_{exact} - bn}{V_{exact}} = 1 - \frac{bn}{V_{exact}}$$

$r \geq 1 - \varepsilon$ for all $\varepsilon \geq \dfrac{bn}{\sum_{j \in S^*} v_j}$

Polynomial-Time Approximation Scheme

↳ An r-approximation algorithm for any $r < 1$.

Polynomial-Time Approximation Scheme

↳ An r-approximation algorithm for any $r < 1$.

Given any $\varepsilon > 0$, algorithm produces a $(1-\varepsilon)$-approximate solution to Knapsack

in time $O(\text{poly}(n) \cdot f(1/\varepsilon))$

for some $f(\cdot)$.

Polynomial-Time Approximation Scheme

↳ An r-approximation algorithm for any $r < 1$.

Given any $\varepsilon > 0$, algorithm produces a $(1-\varepsilon)$-approximate solution to Knapsack in time $O(\text{poly}(n) \cdot f(1/\varepsilon))$ for some $f(\cdot)$.

Fully PTAS

$$O(\text{poly}(n, 1/\varepsilon))$$

Given ε,

$$b \leq \frac{\varepsilon \cdot V_{exact}}{n}$$

choose $\boxed{b \leq \frac{\varepsilon \cdot V^*}{n}}$

V_{exact} at least the max value
(pre-process to remove infeasible values)

Given ε,

choose $b \leq \dfrac{\varepsilon \cdot v^*}{n}$

$$\Rightarrow \tilde{v} \leq \frac{v^*}{\frac{\varepsilon \cdot v^*}{n}} = \boxed{\frac{n}{\varepsilon}}$$

max coarse value

<u>$(1-\varepsilon)$ - Approximate Knapsack</u> $(W, w_1, \ldots, w_n, v_1, \ldots, v_n)$

$$b = \frac{\varepsilon \cdot \max_{i \in \{1 \ldots n\}} v_i^*}{n}$$

$\tilde{v}_i \leftarrow \left\lceil \dfrac{v_i}{b} \right\rceil$ for all $i \in \{1 \cdots n\}$

Return $S \leftarrow$ Knapsack $(W, w_1, \ldots, w_n, \tilde{v}_1, \ldots, \tilde{v}_n)$

Dynamic Program for Knapsack

* 2D Table $\underbrace{T[i, U]}$

minimum weight $\sum_{j \in S_{i,U}} w_j$ where $S_{i,U} \subseteq \{1 \ldots i\}$ and $\sum_{j \in S_{i,U}} v_j \geq U$

Dynamic Program for Knapsack

* 2D Table $\underbrace{T[i, U]}$

minimum weight $\sum_{j \in S_{i,u}} w_j$ where $S_{i,u} \subseteq \{1 \dots i\}$ and $\sum_{j \in S_{i,u}} v_j \geq U$

- i tracks subset of elements considered.
- U tracks value

Eventually return max value U that doesn't exceed weight W

Dynamic Program for Knapsack

* 2D Table $\underbrace{T[i, U]}_{\text{minimum weight } \sum_{j \in S_{i,u}} w_j \text{ where}}$

$S_{i,u} \subseteq \{1 \ldots i\}$ and $\sum_{j \in S_{i,u}} v_j \geq U$

Cases.

① $i \notin S_{i,u} \implies T[i, U] = T[i-1, U]$

② $i \in S_{i,u} \implies T[i, U] = w_i + T[i-1, \underbrace{U - v_i}]$

Need to handle case where < 0.

Knapsack DP Values

Initialize $T[i, U] = +\infty$ for $i \in \{1, \dots, n\}$
$U \in \{0, \dots, nv^*\}$

For $i = 1, \dots, n$ $\quad T[i, u_0] = 0$ for $u_0 \leq 0$.

For $i = 1, \dots, n$.

For $U = 1, \dots, \sum_{j=1}^{i} v_j$

+∞ for infeasible U.

$$T[i, U] = \min \begin{cases} T[i-1, U], \\ w_i + \underbrace{T[i-1, U - v_n]}_{0 \text{ for } u_0 \leq 0.} \end{cases}$$

Return

$\max U$ s.t. $T[n, U] \leq W$.

__Claim.__ DP returns the correct value.

__Claim.__ DP runs in $O(n^2 \tilde{V})$ time.

$\Rightarrow$ __Theorem.__ Knapsack has an FPTAS.

For any $\varepsilon > 0$.

$(1-\varepsilon)$-approximate algorithm

running in $O(n^3/\varepsilon)$ time.