CS7800: Advanced Algorithms

Class 16: Approximation Algorithms I

- Knapsack
- Maximum Coverage

Jonathan Ullman

October 31, 2025

Approximation Algorithms

Defined by the "data"

Objective function $f : \mathcal{X} \to \mathbb{R}$ real numbers

Set of feasible solutions $\mathcal{X}$

~~Goal: find $x \in \mathcal{X}$ that maximizes $f(x)$~~

$$\boxed{\text{Goal: find } x \text{ s.t. } f(x) \geq c \cdot \max_{x^*} f(x^*)}$$

Sometimes called c-approximation

Does not (necessarily) contradict NP-hardness of exact maximization/minimization

Approximation Algorithms

- Many NP-hard optimization problems have interesting approx algorithms
  - Knapsack, Set/Vertex Cover, Traveling Salesman, ...
  - Some do not! (But that's for another course)

- Many interesting techniques
  - Greedy, discretization, LPs, ...

- Useful way of analyzing natural heuristics

Knapsack

Input:

- n items with values $v_i \geq 0$, weights $w_i \geq 0$
- capacity constraint W

Output:

- subset $S \subseteq \{1, \dots, n\}$ s.t. $\sum_{i \in S} w_i \leq W$
- Goal: maximize $\sum_{i \in S} v_i$

- NP-hard to solve exactly in polynomial time
- Can solve exactly in time $O(n 2^n)$, $O(nW)$, $\boxed{O(nV)}$

How?

$\sum_{i=1}^{n} v_i$

Input size in bits: $(2n+1)\log W$

Greedy Knapsack

> Add items in decreasing order of ___???___ until you run out of room

① Decreasing value $v_1 \geq v_2 \geq \ldots \geq v_n$

Bad input $v_1 = B \quad v_2 = \ldots = v_n = B-1$

$w_1 = W \quad w_2 = \ldots = w_n = 1$

optimal value $(B-1)\cdot W$ greedy value B approx ratio $\frac{B}{(B-1)W} \approx 1/W$

② Decreasing "bang-for-buck" $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \frac{v_n}{w_n}$

Bad input $v_1 = 1 \quad v_2 = W-1$

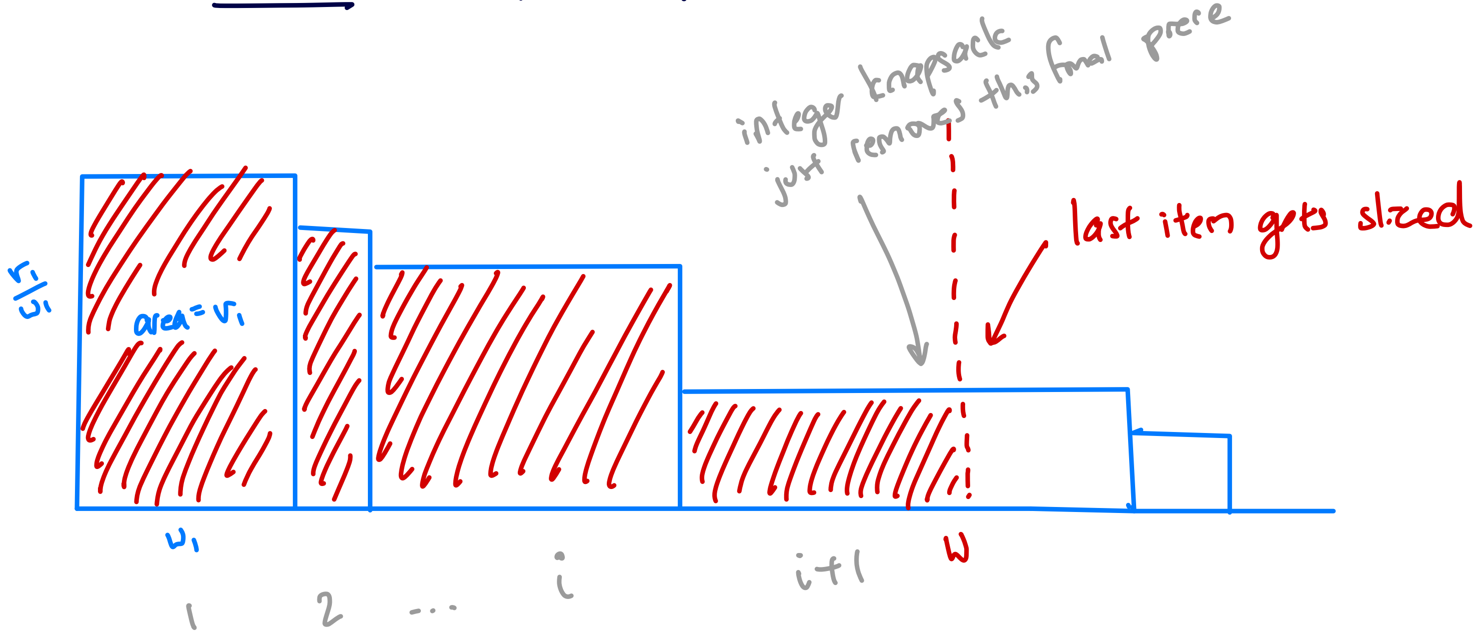
$w_1 = 1 \quad w_2 = W$

$\frac{v_1}{w_1} = 1 \quad \frac{v_2}{w_2} = 1 - 1/W < 1$

optimal value $W-1$ greedy value 1 approx ratio $\frac{1}{W-1}$

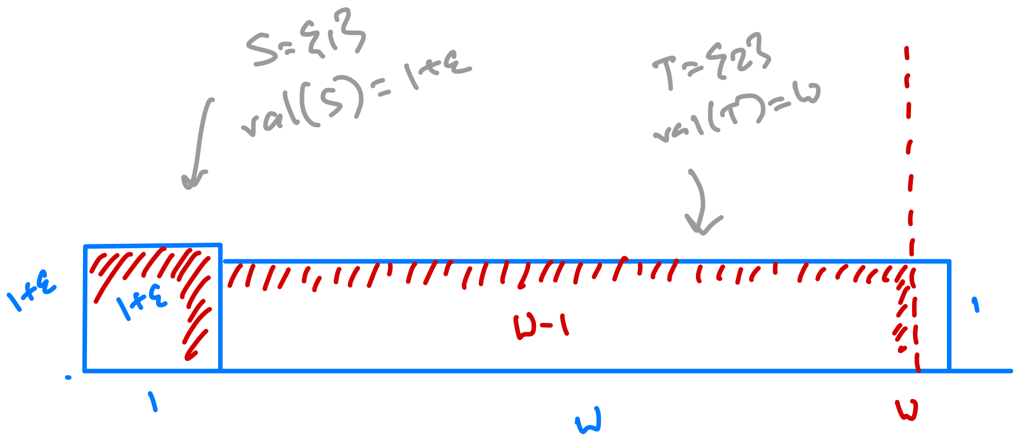
Aside: Fractional Knapsack

Claim: Greedy in descending order of bang-for-buck is optimal for the fractional knapsack problem.



Bad input $v_1 = 1+\varepsilon$ $v_2 = W$

$w_1 = 1$ $w_2 = W$



Modified Greedy Knapsack

① Sort by bang-for-buck $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \dots \geq \frac{v_n}{w_n}$

② Add items $1, 2, \dots, k$ until you run out of space

③ Take better of $S = \{1, 2, \dots, k\}$ and $T = \{k+1\}$

Thm: ModGreedy is a $\frac{1}{2}$-approximation

Proof:

- opt is the optimal integer value
- fracopt is the optimal fractional value

$$\text{opt} \leq \text{fracopt}$$

WTS: $\text{opt} \leq 2 \cdot \text{greedy}$

$$\text{opt} \leq \text{fracopt} \leq \sum_{i=1}^{k} v_i + v_{k+1}$$

$$= \text{val}(S) + \text{val}(T)$$

$$\leq 2 \cdot \text{greedy}$$

$$\Rightarrow \text{greedy} \geq \tfrac{1}{2} \cdot \text{opt}$$

Take 2: Faster DP for Knapsack

$V = \sum_{i=1}^{n} v_i$

Fact: There is a DP algorithm with running time $O(nV)$

↳ Maybe we can "change units" to make values small

1. Let $v_{max} = \max_i v_i$ and $\alpha = \frac{n}{\varepsilon \cdot v_{max}}$
2. For $i=1,\dots,n$ let $v_i' = \lfloor \alpha v_i \rfloor$ // $v_i' \in \{0, 1, \dots, n/\varepsilon\}$
3. Run DP on inputs $\{(v_i', w_i)\}_{i\in[n]}$ and W

Thm: DPApprox is a $(1-\varepsilon)$-approximation and runs in time $O(\frac{n^3}{\varepsilon})$

Faster DP for Knapsack

$\alpha = \frac{n}{\varepsilon \cdot v_{max}}$

Theorem: DPApprox is a $(1-\varepsilon)$-approximation

Pf: - We can assume $OPT \geq v_{max}$ ← Why?

- Key Claim: For every $S \subseteq \{1, \dots, n\}$ $\underbrace{\alpha v(S)}_{\text{value of sets}} \geq \underbrace{v'(S)}_{} \geq \alpha v(S) - n$

$\sum_{i \in S} \alpha v_i$ $\qquad \sum_{i \in S} v_i' = \sum_{i \in S} \lfloor \alpha v_i \rfloor$

WTS: If DP returns S' (opt for v_i'), S^* is the opt then $v(S') \geq (1-\varepsilon) v(S^*)$

$$v(S') \underset{\text{By Clm}}{\geq} \frac{1}{\alpha} \cdot v'(S') \underset{\text{By optimality for } v_i'}{\geq} \frac{1}{\alpha} \cdot v'(S^*) \underset{\text{By Clm}}{\geq} \frac{1}{\alpha} \cdot \left(\alpha v(S^*) - n\right) = v(S^*) - \frac{n}{\alpha}$$

$$= v(S^*) - \varepsilon \cdot v_{max}$$

$$\geq v(S^*) - \varepsilon v(S^*)$$

Maximum Coverage

Inputs: Sets $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$
A budget $k \geq 0$

Outputs/Objective: Choose sets $\{A_1, \ldots, A_k\} \subseteq \{S_1, \ldots, S_m\}$
maximizing $\left| \bigcup_{i=1}^{k} A_i \right|$

- Can solve in time $O\left(\binom{m}{k}\right) = O(m^k)$

- Problem is NP-hard to solve exactly

Why?

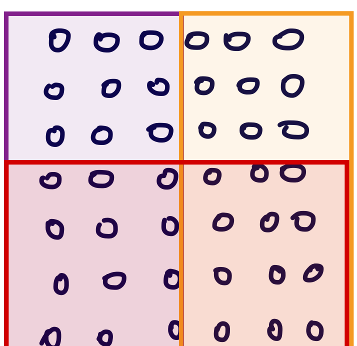
Greedy Max Coverage

For $i = 1, \dots, k$:

- Let A_i be the set maximizing $|A_1 \cup A_2 \cup \dots \cup A_i|$

Equivalent to maximizing $|A_i \setminus (A_1 \cup \dots \cup A_{i-1})|$

Bad Example ($k=2$):



is $1 \times \frac{1}{2}$

is $1 \times \frac{1}{2}$

is $(\frac{1}{2} + \varepsilon) \times 1$

For $k=2$, greedy is at best a $\frac{3}{4}$-approx

Greedy Max Coverage Analysis

Key Claim: Let c_i be the number of elts covered by the first i sets $A_1 \cup \ldots A_i$. Then at iteration i there exists a set that covers at least $\frac{OPT - c_i}{k}$ new elements

$\Longrightarrow$ for every i, $c_i - c_{i-1} \geq \frac{OPT - c_{i-1}}{k}$

Greedy Max Coverage Analysis

Greedy Max Coverage

For $i = 1, \ldots, k$:

- Let A_i be the set maximizing $|A_1 \cup A_2 \cup \ldots \cup A_i|$

Equivalent to maximizing $|A_i \setminus (A_1 \cup \ldots \cup A_{i-1})|$

Thm: Greedy MC gives a $(1-1/e)$-approximation in time $O(n^3)$

I didn't think hard about this