CS 270: Combinatorial Algorithms and Data Structures Spring 2023

Lecture 19 — March 21, 2023

Prof. Jelani Nelson *Scribe: Calvin Yan, Young Jin Park*

1 Overview

In the last lecture we expanded upon primal/dual analysis for approximation algorithms, introduced the integrality gap as an obstacle to approximation, and proposed a solution in the form of polynomial time approximation schemes (PTAS) and fully polynomial time approximation schemes (FPTAS).

In this lecture we will continue analysis of PTAS and FPTAS, specifically with regards to knapsack, and conclude approximation algorithms with the techniques of fully polynomial-time randomized approximation schemes (FPRAS) and SDP-rounding.

2 Last Time: Greedy 2-Approximation For Knapsack

Last lecture we claimed without proof that a greedy solution provides a 2-approximation of integral knapsack. This lecture, we will prove that claim.

First, a recap of the algorithm. Intuitively, we can sort each item by by $\frac{v_i}{w_i}$, which represents the cost-effectiveness for some item i, then take all the elements that we can starting by the most cost-effective item.

This is optimal for the fractional case, where we can take a fraction of each element. *However*, it is not optimal for integral knapsack. We can make a very simple optimization:

(1) Take as much of each item as we can, in descending order of their effectiveness $\frac{v_i}{w_i}$.

(2) Take $v_{\max}$, the largest element we can afford with our space, and nothing else.

(3) Use the strongest result between (1) and (2).

Lemma 2.1. *Suppose the greedy algorithm takes items $1, 2, \ldots, k-1$ but not the k-th item from our sorted sequence where $\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \ldots \geq \frac{v_k}{w_k}$. Then the sum $\sum_{i=1}^{k} v_i > \mathsf{OPT}(ILP)$.*

Proof. Recall that we can write the LP

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} x_i v_i \\ s.t. \quad & \sum_{i=1}^{n} x_i w_i \leq W \\ & \forall i;\ 0 \leq x_i \leq 1 \end{aligned}$$

LP relaxation tells us that

$$\mathsf{OPT} = \mathsf{OPT}(ILP) \leq \mathsf{OPT}(LP)$$

Since fractional knapsack took all of the first $k-1$ items and some fraction (maybe 0) of the k-th item, we know that the remaining items it took contribute less value than if it took the entirety of the k-th item (since it has more weight and contributes more value per weight by our sorting order). We have the strict inequality

$$\sum_{i=1}^{k} v_i > \mathsf{OPT}(LP)$$

$$\mathsf{OPT}(ILP) \leq \mathsf{OPT}(LP) < \sum_{i=1}^{k} v_i$$

□

Claim 2.2. The better of (1) or (2) achieves $\geq \frac{1}{2}\mathsf{OPT}$.

Proof. By Lemma 2.1,

$$\sum_{i=1}^{k} v_i = \underbrace{\sum_{i=1}^{k-1} v_i}_{\leq \text{greedy}} + \underbrace{v_k}_{\leq v_{\max}} > \mathsf{OPT}$$

It then follows that at least one of greedy or $v_{\max}$ is $> \frac{\mathsf{OPT}}{2}$. □

3 PTAS for Knapsack

As a reminder, a polynomial-time approximation scheme aims to find a result $\geq (1-\epsilon)\mathsf{OPT}$ in time $n^{f(1/\epsilon)}$.

Observation. The number of items of value $> \epsilon\mathsf{OPT}$ that the optimal solution takes is at most $\lfloor \frac{1}{\epsilon} \rfloor$.

Idea. Guess $S = \{1/\epsilon$ largest items that OPT takes$\}$.

For any particular guess S, we want to pack the remaining items well. To do this, we can use the greedy approach on the remaining elements, i.e. the elements smaller than $\epsilon\mathsf{OPT}$.

However, we don't know the value of $\epsilon\mathsf{OPT}$. How can we compare against $\epsilon\mathsf{OPT}$ if we don't even know its value? We can guess the lowest value of any item in S and set $\tau := \epsilon\mathsf{OPT}$ slightly lower than this, greedily packing the items with less than τ value. There are $O(n)$ such thresholds, so we can just try them all. An extra $O(n)$ factor of runtime is fine, since it's still polynomial in n.

How much do we lose from using greedy in this remaining portion? Recall from the proof for Claim 2.2 that $\mathsf{Greedy} > \mathsf{OPT} - v_k$, so we lose at most v_k which is $\leq v_{\max} \leq \epsilon\mathsf{OPT}$ by construction. This gives us the value $(1-\epsilon)\mathsf{OPT}$, which is exactly what we want.

The final runtime is equal to

$$(\text{number of guesses for } S) \times \underbrace{(\text{time per } S)}_{\text{poly}(n)}$$

The number of guesses for S is the number of sets of $\leq \frac{1}{\epsilon}$ items we can take. A neat combinatorical trick for computing this is to add $\frac{1}{\epsilon}$ dummy items and compute the equivalent quantity

$$\underbrace{\binom{n+\frac{1}{\epsilon}}{\frac{1}{\epsilon}}}_{\substack{\text{Upperbounded by number of} \\ \text{subsets of } \{1,\ldots,n\} \text{ of size } \leq \frac{1}{\epsilon}}} \leq (e(\epsilon n+1))^{\frac{1}{\epsilon}}$$

Note the exponential dependence on $\frac{1}{\epsilon}$, which makes this PTAS, not FPTAS.

4 FPTAS for Knapsack

Recall that a fully polynomial time approximation scheme is a PTAS with worst-case poly($\frac{n}{\epsilon}$) runtime. Results of $\tilde{O}(n+\frac{1}{\epsilon^4})$[1] and $\tilde{O}(n+\frac{1}{\epsilon^{2.2}})$[2] have been found, and it remains an open problem to get this bound down to $\tilde{O}(n+\frac{1}{\epsilon^2})$, but these are beyond the scope of today's lecture. Instead, we will demonstrate a relatively simpler approach that achieves $O(\frac{n^3}{\epsilon})$ runtime.

Starting point: We know an exact DP algo with $O(nV)$ runtime, where $V := \sum_{i=1}^{n} v_i$

Construction 4.1. Construct a new knapsack problem with values v_i' and same weight w_i s.t. $v_i' := \lfloor \frac{n}{\epsilon} \cdot \frac{v_i}{v_{\max}} \rfloor$, then run the exact DP solution on the modified problem with (v_i', w_i), and return this optimal set. The runtime of this approximation is $O(nV') = O(n\frac{n^2}{\epsilon}) = O(\frac{n^3}{\epsilon})$

Analysis for FPTAS: Let A be the optimal set for v_i and B the optimal set for v_i'. Observe that for $\alpha = \frac{n}{\epsilon v_{\max}}$,

$$\begin{aligned} v_i' &= \lfloor \alpha v_i \rfloor \\ &\implies \alpha v_i - 1 \leq v_i' \leq \alpha v_i \\ &\implies \frac{1}{\alpha} v_i' \leq v_i \leq \frac{1}{\alpha}(v_i'+1) \end{aligned}$$

it follows that

$$\begin{aligned} \mathsf{val}(B) \geq \frac{1}{\alpha}\mathsf{val}(B) &\geq \frac{1}{\alpha}\mathsf{val'}(A) \\ &\geq \frac{1}{\alpha}(\alpha \mathsf{val'}(A) - |A|) \\ &= \underbrace{\mathsf{val}(A)}_{\mathsf{OPT}} - \frac{\epsilon v_{\max}}{n}|A| \\ &\geq \mathsf{OPT} - \epsilon v_{\max} \\ &\geq (1-\epsilon)\mathsf{OPT} \end{aligned}$$

5 FPRAS for DNF-counting

Definition 5.1. A condition is in conjunctive normal form (CNF) if it is an AND of a bunch of ORs: $(x_1 \vee \overline{x_3} \vee x_7 \vee x_2) \wedge (\overline{x_3}) \wedge (\dots$

Definition 5.2. A condition is in disjunctive normal form (DNF) if it is an OR of a bunch of ANDs: $(x_1 \wedge \overline{x_3} \wedge x_7 \wedge x_2) \vee (\overline{x_3}) \vee (\dots$

Remark. CNF satisfiability is merely the SAT problem which we know to be NP-complete. DNF is not (since we can just find an assignment that satisfies one clause!). However, we do have a harder problem regarding DNF.

Goal: Given DNF formula ϕ, count the number of x s.t. $\phi(x)$ is true.

Claim 5.3. DNF-counting is NP-hard (equivalently, DNF is $\#P$-complete).

Proof. We aim to show that SAT reduces to DNF-counting.
Given a problem in SAT form,

$$\phi = \underbrace{(x_1 \vee \overline{x_3} \vee \dots)}_{C_1} \wedge \underbrace{(\overline{x_7} \vee \dots)}_{C_2} \wedge \dots \wedge \underbrace{(\dots)}_{C_m}$$

Then

$$\begin{aligned}\overline{\phi} &= \overline{(x_1 \vee \overline{x_3} \vee \dots) \wedge (\overline{x_7} \vee \dots) \wedge \dots \wedge (\dots)} \\ &= \overline{(x_1 \vee \overline{x_3} \vee \dots)} \vee \overline{(\overline{x_7} \vee \dots)} \vee \dots \vee \overline{(\dots)} \\ &= (\overline{x_1} \wedge x_3 \wedge \dots) \vee (x_7 \wedge \dots) \vee (\dots)\end{aligned}$$

which is in disjunctive normal form. If $\overline{\phi}$ has T solutions, then ϕ has $2^n - T$ solutions, since the statements are complementary and there are 2^n variable assignments in total. Then ϕ is satisfiable if and only if DNF-counting finds $T < 2^n$. □

New goal: Find $\tilde{T}$ such that $\mathbb{P}(|T - \tilde{T}| > \epsilon T) < \delta$. In more intuitive terms: $\tilde{T} = (1 \pm \epsilon)T$ with high probability.

Idea 1: Define $p := \frac{T}{2^n}$. We want to know $\tilde{p} = (1 \pm \epsilon)p$ and return $2^n\tilde{p}$. However, there is a problem with this idea: A small p results in high estimation variance.

Example: Consider $\phi = x_1 \wedge x_2 \wedge \dots \wedge x_n$

There is only one assignment that satisfies this. When the number of solutions is exponentially small, we will probably not sample any in polynomial time! There is a serious possibility that you will estimate $p = 0$ and return $2^n \times 0 = 0$.

Thankfully, there is a way to reexpress the estimation that ensures p isn't too small[3]:

$$
\begin{aligned}
B &:= \text{ the set of satisfying assignments} \\
B' &:= \{(i, x) : x \text{ satisfies } C_i\} \\
S_i &:= \{x : x \text{ satisfies } C_i\} \\
\implies B &= \bigcup S_i \\
B' &= \bigsqcup S_i
\end{aligned}
$$

Here $\sqcup$ refers to the disjoint union.

New goal: estimate $p' := \frac{|B|}{|B'|}$

Observation. $|B'| \leq m|B| \implies p' \geq \frac{1}{m}$
This is because each $x \in B$ appears in B exactly once, but may appear in B' up to m times (for $i = 1, 2, \ldots, m$).

Algorithm The estimation of p' can be done using Monte Carlo sampling.

- for $j = 1$ to l,
  - pick random $(i, x) \in B'$
  - check if $(i, x) \in B$
- return $\frac{\#\text{ of samples in } B}{l}$

Observation. B is in bijective correspondence with a subset of B'.

$$B \leftrightarrow \{(i, x) : C_i \text{ is the first clause that } x \text{ satisfies}\}$$

How do we sample an element of B' uniformly at random?

To sample a clause at random, pick i with probability $\frac{|S_i|}{\sum_{j=1}^{m} |S_j|}$. To sample an assignment at random, assign deterministically the variables in C_i such that it is satisfied, and all other variables uniformly at random.

Analysis: Define $X_1, \ldots, X_l$ as follows:

$$
\begin{aligned}
X_j &:= \begin{cases} 1 & \text{if } j\text{th sample} \in B \\ 0 & \text{otherwise} \end{cases} \\
X &:= \sum_{j=1}^{l} X_j,
\end{aligned}
$$

Observation. $\mathbb{E}(X) = p' \cdot l$ due to linearity of expectation.

Then apply the Chernoff bound:

$$\begin{aligned}
\mathbb{P}(|X-\mu| > \epsilon\mu) &\leq 2e^{-\epsilon^2\mu/3} \\
&= 2e^{-\epsilon^2 p' l/3} \\
&\leq 2e^{-\epsilon^2 l/3m} \leq \delta \\
l &= \lceil \frac{3m}{\epsilon^2} \ln(\frac{2}{\delta}) \rceil
\end{aligned}$$

References

[1] Eugene L. Lawler. Fast Approximation Algorithms for Knapsack Problems. *Math. Oper. Res.*, 4(4): 339-356, 1979.

[2] Mingyang Deng, Ce Jin, Xiao Mao. Approximating Knapsack and Partition via Dense Subset Sums. *Symposium on Discrete Algorithms*, 2961-2979, 2023.

[3] Richard M. Karp, Michael Luby, Neal Madras. Monte-Carlo Approximation Algorithms for Enumeration Problems. *J. Algorithms*, 10(3):429-448, 1989.