Advanced Algorithmics

Strategies for Tackling Hard Problems

Sebastian Wild

Markus Nebel

Lecture 21

2017-07-03

Definition 5.37 (APX, PTAS, FPTAS)

$$\begin{aligned} \mathcal{APX} &= \{U \in \mathcal{NPO} \;:\; \exists \text{ constant } c : \exists c\text{-approx for } U\}, \\ \mathcal{PTAS} &= \{U \in \mathcal{NPO} \;:\; \exists \text{ PTAS for } U\}, \\ \mathcal{FPTAS} &= \{U \in \mathcal{NPO} \;:\; \exists \text{ FPTAS for } U\}, \end{aligned}$$ ◀

Obviously, we have

$$\mathcal{PO} \subseteq \mathcal{FPTAS} \subseteq \mathcal{PTAS} \subseteq \mathcal{APX} \subseteq \mathcal{NPO}$$

Theorem 5.38 (Approximation Classes)

Unless $\mathcal{P} = \mathcal{NP}$, all of the above inclusions are *strict*. ◀

FPTAS for Knapsack

n items $w_1, \dots, w_n$ $v_1, \dots, v_n$ all integers
b capacity of knapsack

$\max \sum_{i \in I} v_i$ s.t. $\sum_{i \in V} w_i \le b$

Assumption: any item fits in the knapsack alone, i.e., $w_i \leq b$

greedy $\frac{v_i}{w_i}$ $\neq$ approx

```
procedure approxKnapsack(w,v,b,ε)
    V̂ = max_{i=1,...,n} v_i
    K = εV̂/n
    ṽ = ⌊v/K⌋        // round values to certain precision
    return DPKnapsack(w,ṽ,b)
```

Knapsack

weight of
$A[i,v] =$ min-weight subset with value $= v$

$n \cdot V$

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

Theorem 5.39

approxKnapsack is an FPTAS for 0/1-Knapsack

Running Time: DPKP : $O(n \cdot \tilde{V} \cdot \log \tilde{V})$

$\le n \cdot \hat{V}$

$$\sum_{i=1}^{n} \tilde{v}_i \le \frac{\sum v_i}{K} = \frac{V}{K} \le \frac{n^2}{\varepsilon}$$

$O\left(n^3 \log\left(\frac{1}{\varepsilon}\right) \frac{1}{\varepsilon}\right)$

poly-time wrt $\frac{n}{\varepsilon}$

$(1+\varepsilon)$-approx : $\mathrm{cost}_{AKP} \overset{!}{\geq} (1-\varepsilon)\,\mathrm{OPT}$

T optimal solution $\rightsquigarrow$ cost $\mathrm{OPT} = \sum_{i \in T} v_i$

$\tilde{T}$ approx. solution $\rightsquigarrow$ $\mathrm{cost}_{AKP} = \sum_{i \in \tilde{T}} v_i$

- $\forall i \in [n]$ $\quad \tilde{v}_i = \left\lfloor \frac{v_i}{K} \right\rfloor \in \left(\frac{v_i}{K} - 1, \frac{v_i}{K} \right] \qquad (*)$

- for any $I \subseteq [n]$

$$\sum_{i \in I} \tilde{v}_i \geq \sum_{i \in I} \left(\frac{v_i}{K} - 1 \right) = \frac{1}{K} \sum_{i \in I} v_i - \underbrace{|I|}_{\leq n} \qquad (1)$$

$$\sum_{i \in I} \tilde{v}_i \leq \sum_{i \in I} \frac{v_i}{K} = \frac{1}{K} \sum_{i \in I} v_i \qquad (2)$$

$$\mathrm{cost}_{AKP} = \sum_{i \in \tilde{T}} v_i \underset{(2)}{\geq} K \cdot \sum_{i \in \tilde{T}} \tilde{v}_i \underset{\substack{\text{DPKP} \\ \text{optimal}}}{\geq} K \cdot \sum_{i \in T} \tilde{v}_i \underset{(1)}{\geq} \sum_{i \in T} v_i - n \cdot K$$

$$= \mathrm{OPT} - \varepsilon \hat{V} \qquad // \ \mathrm{OPT} \geq \hat{V}$$

$$\geq (1 - \varepsilon) \cdot \mathrm{OPT}$$

$\square$

FPTAS asks for much

Theorem 5.40 (FPTAS → FPT and pseudopolynomial)

1. $U \in \mathcal{FPTAS} \implies p\text{-}U \in \mathcal{FPT}$ canonical parametrization
2. $U \in \mathcal{FPTAS}$ and $\overset{\in \mathbb{N}}{cost}(u,x) < p\big(MaxInt(x)\big)$ for some polynomial p
   $\implies$ $\exists$ pseudopolynomial algorithm for U.

Proof: 1 assume goal = min

$A(x,\varepsilon)$ FPTAS for U running time $\le q(|x|, \frac{1}{\varepsilon})$ q polynomial

Construct algo. B for $p\text{-}U$

$B(x,k)$ // output $\exists y \in M(x) : cost(y,x) \le k$

$y = A(x, \frac{1}{k+1})$

return $cost(y,x) \le k$

- $cost(y,x) \le k$ (Yes) obviously correct
- $cost(y,x) \ge k+1$

$$\underset{A\text{ approx.}}{OPT \ge} \frac{cost(y,x)}{1+\frac{1}{k+1}} \ge \frac{k+1}{1+\frac{1}{k+1}} = \frac{k+1}{k+2}(k+1) > k \implies \nexists\, y \in M(x) : cost \le k$$

$\Rightarrow$ No-instance

Running Time: $A(x, \frac{1}{k+1})$ runs

in $g(|x|, k+1) = \text{fpt}$

- if $|x| \leq k+1$ $\quad g(|x|, k+1) \leq g(k+1, k+1) = f(k)$
- if $|x| > k+1$ $\quad g(|x|, k+1) \leq g(|x|, |x|) = g'(|x|)$

w.l.o.g. g increasing

(2) $y = A(x, \varepsilon)$ with $\varepsilon = \dfrac{1}{p(\mathrm{MaxInt}(x))}$

assume goal = min

$$\mathrm{cost}(y, x) \leq (1+\varepsilon)\,\mathrm{OPT}$$

$$< \mathrm{OPT} + \varepsilon \cdot p(\mathrm{MaxInt}(x))$$

$$= \mathrm{OPT} + 1$$

$$\Rightarrow \mathrm{cost}(y, x) = \mathrm{OPT}$$

Running Time: $g(|x|, \frac{1}{\varepsilon}) = g(|x|, p(\mathrm{MaxInt}(x)))$

$=$ polynomial $|x|$, $\mathrm{MaxInt}(x)$

$\Rightarrow$ pseudo polynomial

$\square$

Bin Packing

min k

Bin-Packing strongly NP-hard $\Rightarrow$ no FPTAS (unless P = NP)

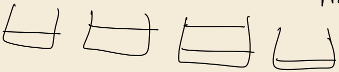
Recall: BIN-PACKING

Given: $w_1, \dots, w_n \in \mathbb{N}$, $b \in \mathbb{N}$, $\cancel{k \in \mathbb{N}}$

Question: $\exists a : [n] \to [k] \;:\; \forall j \in [k] : \sum_{\substack{i=1,\dots,n \\ a[i]=j}} w_i \le b$?

Theorem 5.41 (First fit 2-approx)

The first-fit heuristic is a 2-approximation for BIN-PACKING.



Proof: ≤ 1 active bin $<$ half full

- item $< \frac{1}{2}$
  - exists $<$ half full bin $\to$ put in
  - $\neg$ exist $<$ half full bin $\to$ start new
- item $\geq \frac{1}{2}$
  - start new one

obviously $OPT \geq \left\lceil \frac{\sum w_i}{b} \right\rceil$

$\square$

A first inapproximability result

Theorem 5.42

There is no poly-time $(\frac{3}{2} - \varepsilon)$-approximation for Bin-Packing for any $\varepsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.

Proof: PARTITION

NP-complete

Input: $x_1, \dots, x_n \in \mathbb{N}$

Question: $\exists I \subseteq [n]: \sum_{i \in I} x_i = \sum_{i \in \bar{I}} x_i$?

$\bar{I} = [n] \setminus I$

reduce PARTITION to BIN-PACKING

$w = x$

$b = \left\lfloor \frac{\sum x_i}{2} \right\rfloor$ $\quad k = 2$

If we had $(\frac{3}{2} - \varepsilon)$-approx poly-time

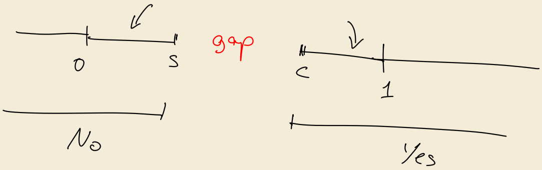
$\rightsquigarrow$ would optimally solve this

(distinguish 2 and 3) $\square$

How can we transfer this result to other problems?

Is it tight?

5.9 Inapproximability



Assume in this section: $goal = \max$.

Definition 5.43 (Gap problem)

Let c, s with $0 \leq s \leq c \leq 1$ be given and let $U = (\Sigma_I, \Sigma_O, L, L_I, M, cost, goal)$ an optimization problem form $\mathcal{NPO}$. We define the $\mathrm{GAP}_{c,s}$-U decision problem as follows:

- Input: $x \in L_I$ such that either $Opt_U(x)/|x| \geq c$ or $Opt_U(x)/|x| < s$ holds.
- Output:
  - Yes, in case $Opt_U(x)/|x| \geq c$;
  - No, in case $Opt_U(x)/|x| < s$.

Note: We will interpret $|x|$, the length of an encoding of the instance, a bit more freely and use a more natural unit of size for the input, e.g., the number of clauses for 3SAT or the number of nodes in Independent-Set.

Lemma 5.44 (Hard Gap → no approx)

Let $U \in \mathcal{NPO}$ and c, s with $0 \leq s \leq c \leq 1$ two constants.
If $\mathrm{GAP}_{c,s}$-U is $\mathcal{NP}$-hard then under the assumption $\mathcal{P} \neq \mathcal{NP}$, then there is no polynomial time $\frac{c}{s}$-approximation algorithm for U.

Proof: Assume A computes $\frac{c}{s} = \delta$ -approx. poly-time.

Build decider B for $\mathrm{GAP}_{c,s}$-U

assume goal = max

$y = A(x)$

$\mathrm{cost}(y) < s \cdot |x|$

$\mathrm{cost}(y) < s|x| \overset{?}{\iff} \mathrm{Opt}_U(x) < s \cdot |x|$

"$\Leftarrow$" $\mathrm{Opt}_U(x) < s|x| \implies \mathrm{cost}(y) \leq \mathrm{Opt}_U(x) < s|x|$

"$\Rightarrow$" $\mathrm{Opt}_U(x) \geq s|x| \underset{\text{Gap!}}{\implies} \mathrm{Opt}_U(x) \geq c \cdot |x|$

$A \quad \frac{c}{s}\text{-approx} \qquad \frac{Opt_U(x)}{cost(y)} \leq \frac{c}{s}$

$\Rightarrow \quad cost(y) \geq \frac{s}{c} Opt_U(x) \geq s|x|$

$\square$

Definition 5.45 (Gap reduction)

Let U_1 and U_2 be two maximization problems with potentially different input and output alphabets. U_1 is GP-reducible to U_2 (notation $U_1 \leq_{GP} U_2$) with parameters (c, s) and (c', s') if and only if there is a polynomial time algorithm A with:

1. For every input $x \in L_{I,1}$ we have $A(x) \in L_{I,2}$.
2. $\frac{Opt_{U_1}(x)}{|x|} \geq c$ implies $\frac{Opt_{U_2}(A(x))}{|A(x)|} \geq c'$.
3. $\frac{Opt_{U_1}(x)}{|x|} < s$ implies $\frac{Opt_{U_2}(A(x))}{|A(x)|} < s'$.