CSC 791/495

Dr. Blair D. Sullivan

November 3, 2017

Optimization & Approximation

optimization problems

- instance → solution w/ cost.
- goal: find a solution w/ min(max) cost.

objective: max/min cost. solution

$OPT(x)$ = min/max possible cost of a valid/feasible solution for x.
instance

Ex: Vertex Cover → min. VC
Clique → max Clique
Ind. Set → max IS.
Makespan Scheduling (min)
cost ≠ soln size

NPO: (equiv of NP)

- size of soln $\in P$
- recognize instance/soln $\in P$
- cost function $\in P$

decision problem: Is $OPT(x) \le q$? (min)
$OPT(x) \ge q$? (max)
$\in$ NP

Approximation We say ALG is a c-approximation for a problem if $\forall x$
instance

$$\frac{cost(ALG(x))}{cost(OPT(x))} \le c \quad (min) \qquad \text{note: } c \ge 1$$

$$\frac{cost(ALG(x))}{cost(OPT(x))} \ge c \quad (max) \qquad c \le 1$$

usually require ALG to be poly-time.

Examples

① Vertex Cover:

Idea: greedy strategy. Not good to add all n, but what about one at a time?
Arbitrary order → could devolve to ↗ Star ⇒ probably high-deg is a good idea.

ALG: While $E(G_i) \neq \emptyset$, add v to cover C where v has max deg in G_i; delete $v \rightarrow G_{i+1}$

(*) Goal: show that after OPT steps, $\geq \frac{1}{2}$ edges are covered (removed)

$\Rightarrow$ run OPT $\Rightarrow \frac{1}{2}$ covered; run 2OPT $\Rightarrow \frac{1}{2} + \frac{1}{4}$ .... run for $O(\text{OPT} \log n) \rightarrow 1$

If we show (*) $\Rightarrow$ $O(\log n)$-approx alg.

Exercise: show (*).

(1b) greedily covering edges actually leads to a much better approximation.

ALG: while $E(G_i) \neq \emptyset$, pick an edge uv, add u and v to C. delete $u, v \rightarrow G_{i+1}$.

When considering edge uv, we know either $u \in C$ or $v \in C$. So by adding both, we could only have doubled the necessary size.

In-class Exercise

ALG 2: sort so $p_1 \geq p_2 \geq \ldots$ than apply ALG1. Show $3/2$ using similar analysis on last job.

Load balancing: Given jobs $j_1, \ldots, j_k$ w/ processing times $p_1, \ldots, p_k$ and two machines M_1, M_2. Goal: assign jobs to machines so that makespan ($\max_{i=1,2} \sum_{j \to M_i} p_j$) is minimized.

Goal: design a 2-approx. Improve it to $3/2$-approx.

Observe: $OPT \geq \max\{p_i\}$.

Observation 2: $OPT \geq \frac{1}{2} \sum p_i \longrightarrow$ ALG0: use 1 machine.

ALG1: Assign a job $j_1 \to$ machine M_1. Now assign jobs to M_2 until $\sum_{\ell \to M_2} p_\ell > p_1$ (M_2 has more work) $\Rightarrow$ assign another job to M_1. repeat until it has more work. Let $W_j[i]$ = amount of work assigned to machine j when job i is allocated.

Look @ machine w/ longest runtime. Consider the last job it was assigned.

M_1 step k

$W_1[k] \leq \frac{1}{2} \sum p_i \leq OPT$ $\boxed{W_1[k] \leq W_2[k]}$ $p_k \leq OPT$ total $\leq 2 \cdot OPT$

Approximation Schemes

parameter

Given $\varepsilon > 0$, give a $(1+\varepsilon)$-approximation algorithm with polynomial running time (in $|x|$) — for every **fixed** ε

in other words $O(f(\varepsilon) \cdot n^{g(\varepsilon)})$

This is a **PTAS** ← polynomial-time approx. scheme.

complexity class: **PTAS** = {NPO with a PTAS}

EPTAS: **efficient** PTAS $\quad O(f(\varepsilon) \cdot n^{O(1)})$ $\quad$ FPT w.r.t. ε

could be problematic e.g. $2^{1/\varepsilon}$

FPTAS: fully PTAS $\quad O(\mathrm{poly}(1/\varepsilon) \cdot n^{O(1)})$

PTAS Strategies

- we know that our problem is likely hard to solve exactly.

$$\underset{\text{instance}}{\boxed{I}} \Rightarrow \underset{\text{algorithm}}{\boxed{A}} \Rightarrow \underset{\text{solution}}{\boxed{A(I)}}$$

if A is exact, $\text{Cost}(A(I)) = OPT(I)$

solution: add "structure" that depends on ε. (ε big $\Rightarrow$ lots, ε small $\Rightarrow$ little)

Implies 3 strategies

1. structure on input
2. structure on output
3. structure during execution.

we'll talk about these

more approx / PTAS references:

1. Design of Approximation Algorithms by Williamson & Shmoys
2. Approximation Algorithms by Vazirani

both have free PDFs online :)

Structuring Input

(A) simplify $I \to I^*$ in poly. time.

(B) solve on I^* in poly time. $\leftarrow$ I^* had better be nice!

(C) translate solution back (exploit similarity)

Ex Load Balancing $\{\underbrace{p_1, \ldots, p_k}_{\text{proc. times}}\}$ & two machines M_1, M_2.

Ideas for (A):
- rounding
- merging
- cutting
- aligning

define $L = \max(\frac{1}{2} \sum p_i, \max(p_i))$

$L \leq OPT$

categorize jobs: **big** $p_i > \varepsilon L$

small $p_i \leq \varepsilon L$

(A): $I \to I^*$

bigjobs remain the same. (I^* has p_i $\forall i$ w/ $p_i > \varepsilon L$)

let $S = \sum_{i \text{ small job}} p_i \Rightarrow$ give I^* $\lfloor S/\varepsilon L \rfloor$ jobs w/ cost εL each.

(B) key: how many jobs in I^*? how big are I^*'s jobs? $\geq \varepsilon L$

all jobs take $\leq 2L$. $\Rightarrow$ #jobs $\leq \frac{2L}{\varepsilon L} = 2/\varepsilon$ brute force! $2^{2/\varepsilon}$

Structuring Input, cont

(c) translation of jobs in I^* to I.

- reserve $\begin{cases} S_1^* + 2\varepsilon L & \text{on } M_1 \\ S_2^* & \text{on } M_2 \end{cases}$

- obviously, schedule big jobs on same machine as in I^*.

say $S_i^* = \sum \underbrace{\text{small jobs}}_{\varepsilon L}$ on M_i in I^*

greedily assign small jobs in I to M_1 until no more fit in the reserved space. Observe: $\geq S_1^* + \varepsilon L$ is filled

claim less than S_2^* total small jobs remain.

$$\underbrace{S \leq S_1^* + S_2^* + \varepsilon L}_{\text{threw away } \leq \varepsilon L \text{ time } I \to I^*}$$

M_1 has $\leq \underline{B_1^* + S_1^*} + 2\varepsilon L$

M_2 has $\leq \underline{B_2^* + S_2^*}$

know: $S + B \leq 2L \leq 2\mathrm{OPT}$

$\max(B_1^* + S_1^*, B_2^* + S_2^*) = \mathrm{OPT}(I^*)$

claim: $\mathrm{OPT}(I^*) \leq (1+\varepsilon)\,\mathrm{OPT}(I)$.

$$\Downarrow$$

$(1+3\varepsilon)$ on entire scheme.

$\underline{\mathrm{OPT}(I^*) \leq \mathrm{OPT}(I) + \varepsilon L} \leq (1+\varepsilon)\mathrm{OPT}$