15-854: 02/02/00

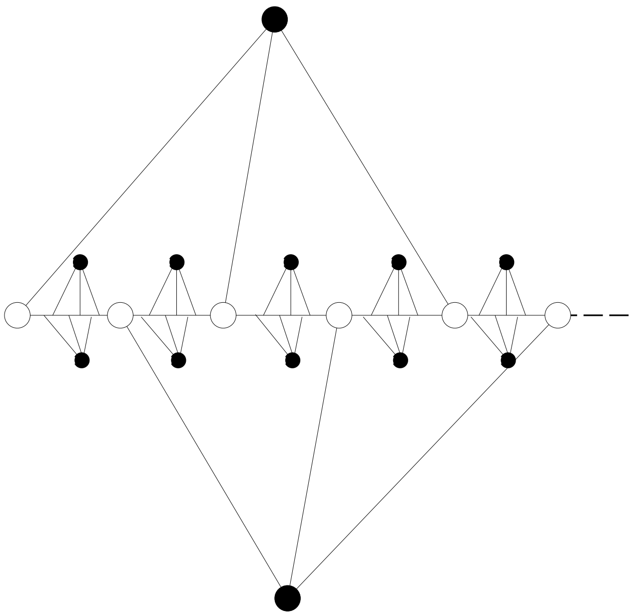
```
-- Recap
-- A little more about approx hardness
-- PTAS and FPTAS
```

Recap from last time

Routing paths to minimize congestion. Idea:

- Solve fractionally (multicommodity flow)
- For each commodity, view flows as probabilities: pick path s.t. each edge has $p_e = f_e$.
- For any given edge, choices independent over different commodities. So, can use Chernoff bounds.
  - W.h.p., no worse than $O(\log n / \log\log n)$ times optimal.
  - Better if optimal is large: $1+o(1)$ if optimal is $\omega(\log n)$.

A tricky example



- Recursion goes k levels deep.
- What's the minimum congestion?

About approx hardness

PCP results give a poly time transformation T from 3CNF formulas to 3CNF formulas with the following property:

- If formula ϕ is satisfiable, then $\phi' = T(\phi)$ is satisfiable.
- If ϕ is not satisfiable, then not only is ϕ' not satisfiable, but also any assignment to the variables of ϕ' satisfies at most a $(1-\varepsilon)$ fraction of the clauses of ϕ'.

This transformation is the [ALMSS92] result.

Natural Approx Classes

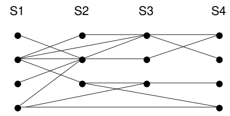
[From Arora-Lund]

Hard to approx to	Representative problem
$(1+\varepsilon)$	MAX-3SAT
$O(\log n)$	Set-Cover
$2^{\log^{1-\varepsilon} n}$	Label-Cover
n^{ε}	Clique

Clique approx hardness

Goal: show that if you could approximate Clique to $N^{\varepsilon/6}$ then could determine which case for ϕ'.

- Let S_1 be $\lg n$ clauses iid at random from ϕ'.
- Repeat n^3 times to create $S_2, \ldots, S_{n^3}$.
- For each S_i, create one node for each assignment to vars in S_i that satisfies S_i.
- Put edge between two vertices if partial assignments are consistent.



- $N \leq n^6$ nodes total.

If ϕ' was satisfiable $\Rightarrow$ clique of size n^3.

Else, each of 2^n assignments satisfies expected $n^3(1-\varepsilon)^{\log_2(n)} < \frac{1}{2}n^{3-\varepsilon}$ of S_i. Chernoff: prob it satisfies twice this is $< e^{-n^{3-\varepsilon}/6} < e^{-n^2}$.

So, w.h.p., no clique of size $> n^{3-\varepsilon}$.

PTAS and FPTAS

For some problems you can get a smooth time $\leftrightarrow$ approx tradeoff.

A Polynomial-Time Approximation Scheme (PTAS) is an algorithm that for any given $\varepsilon > 0$ achieves a $(1+\varepsilon)$ approx, in time polynomial in the problem size (but dependence on ε may be outrageous).

A Fully Polynomial-time Approximation Scheme (FPTAS) is an algorithm that does this in time polynomial in the problem size and in $1/\varepsilon$.

These make sense even if the underlying problem isn't necessarily NP-hard (e.g., multicommodity flow).

List scheduling

- Given: m identical machines, and n jobs. Job i has processing time p_i.
- Goal: put jobs on machines to minimize completion time of last job.

  I.e., partition $\{p_i\}$ into m sets to minimize the largest sum.

NP-complete even for $m = 2$ (partition problem).

Factor of $2 - 1/m$ approx

[Graham '66]

- Look at jobs in order from 1 to n.
- Put each job onto machine with least total so far.

What's a bad case?

Proof of claim:

- WLOG, last job to finish (for us) is job n.
- Let $\text{avg}_i = \frac{1}{m}(p_1 + \ldots + p_i)$.
  - Our cost $\leq \text{avg}_{n-1} + p_n$.
  - OPT $\geq p_n$.
  - OPT $\geq \text{avg}_n = \text{avg}_{n-1} + p_n/m$.

our cost $\leq$ OPT $+ (1 - 1/m)p_n \leq (2 - 1/m)$OPT.

A PTAS (running time $n^{\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}}$)

Preliminaries:

- Assume we know optimal time needed T up to multiplicative $(1 + \varepsilon)$.

  (Use prev alg to get range $[x, 2x]$ and do binary search)
- Ignore jobs with $p_i < \varepsilon T$.

  (Fill them in at end using prev alg)
- Round down all remaining p_i to nearest power of $(1 + \varepsilon)$.

(we'll end up with something like a $(1+3\varepsilon)$ approx)

PTAS, contd

Recap: n jobs, m machines. All processing times p_i are power of $(1 + \varepsilon)$ in range $[\varepsilon T, T]$.

Now, use dynamic programming:

- Only $k = \log_{1+\varepsilon}(1/\varepsilon) \approx \frac{1}{\varepsilon}\ln(1/\varepsilon)$ different p_i.
- So, at most n^k functionally distinct sets of jobs.
- Build big $n^k \times m$ table: best way to pack given set of jobs on first i machines.
- Each entry in col i can be filled in using values in col $i - 1$.

For constant m, can get FPTAS

Can solve in time poly(n, T) — so, poly time if processing times are given in unary — using DP. E.g., $m = 2$:

- Create $n \times T$ table: what sums are achievable using only jobs $1, \ldots, i$.
- $M(i, t) = 1$ if $M(i-1, t) = 1$ or $M(i-1, t-p_i) = 1$.

To make FPTAS for general case, just change units to multiples of $\varepsilon T/n$. (E.g., microseconds to milliseconds).

Changes time on a machine by at most $n(\varepsilon T/n)$.