Lecture 4

Friday, March 24, 2017 10:42 AM

Problem

① Makespan Minimization on Identical Machines

Input:

- n jobs with processing times $p_1, p_2, \ldots, p_n$
- m machines.

Identical: Each job j takes same time on each machine.

If a set $S \subseteq \{1, \ldots, n\}$ is scheduled on a machine, the total processing time is the sum of processing times.

Output: Allocation of jobs to machines.

Objective: Minimize makespan : max proc. time on any m/c.

Formally, find a partition $S_1, S_2, \ldots, S_m$ of $\{1, \ldots, n\}$ s.t.

$$\max^{m} \sum \quad \text{is minimized}$$

$$\max_{i=1}^{m} \sum_{j \in S_i} p_j \quad \text{is minimized.}$$

Graham Notation

This problem is $P \,||\, C_{max}$

Identical machines

no extra constraints.

Makespan

For instance, we could have a machine dependent running time, p_{ij} – job j's proc. time on machine i, and we could've wanted to minimize the same obj. That prob is called $R \,||\, C_{max}$.

Today, we stick to $P \,||\, C_{max}$

Algorithm (The "oldest" appx alg) '1966 [Graham]

- Order the jobs in any order.

- Schedule job j on the machine which has the least load so far.

Analysis

- **Lower Bounds on OPT**

$$\left.\begin{aligned} P_{max} &:= \max_j p_j \\ \overline{P} &:= \frac{1}{m}\sum_{j=1}^{n} p_j \end{aligned}\right\} L := \max(P_{max}, \overline{P})$$

Obs: $OPT \geq L$

We will use this easier lower bnd to compare our algo. It may seem like we're taking a hit, but we have a better handle on it.

Back to analysis:

Let i be the most loaded m/c. Consider the last job j being added to i. Let P be the load on m/c i just before j was added.

- $ALG = P + p_j \leq P + L$

- $P \leq$ load on any other m/c when j was added

$$\Rightarrow P \leq \frac{1}{m} \sum_{j' < j} P_{j'} \leq L$$

$$\therefore \quad ALG \leq 2L$$

$\boxed{\text{Theorem: The above algorithm is a factor 2 appx. alg. for } P||C_{max}.}$

A slightly "simpler" problem: $P2||C_{max}$

So, $m = 2$. Only 2 m/c.
The problem is still NP-hard.
(Do you see this?)

Classification:

- Fix any $\varepsilon > 0$.
- Job j is big if $p_j > \varepsilon L$
  small, o/w.

Partitioning Output Space:

· B be the set of big jobs.

- Any schedule in particular partitions $B = B_1 \cup B_2$
- Let (B_1^*, B_2^*) be the partition of B in the optimal schedule

(*) We can afford to guess this partition.

Why? How many partitions are there?

$$2^{|B|}$$

How large is $|B|$?

$$2L \geq \sum_{j \in B} p_j \geq |B| \cdot \varepsilon L \Rightarrow |B| \leq \frac{2}{\varepsilon}$$

$\therefore$ The # of partitions of $|B| \leq 2^{2/\varepsilon}$

$$= O_\varepsilon(1)$$

fancy not^n for saying it's a const dependent only on ε.

$\therefore$ By enumerating, we may assume we have guessed / know (B_1^*, B_2^*)

Note: in the future, whenever we say

"guess", be aware how much time this guess takes. In this case, the guess is via enumeration and takes $2^{O(1/\varepsilon)}$ time.

Algo:

1. Guess (B_1^*, B_2^*) // $2^{O(1/\varepsilon)}$ time via enum.

2. Order small jobs in any order and schedule on the least loaded m/c .... exactly like before.

Analysis

Two cases.

1. The max-loaded m/c gets no small jobs. In that case,

$$ALG \le \max\left(p(B_1^*), p(B_2^*)\right) \le OPT$$

2. The max-loaded m/c gets at least one small job j. Now we can repeat the above argument ...

$$ALG \le P + p_j \le P + \varepsilon L$$

$$\le (1+\varepsilon) L$$

gain

$\leq (1+\varepsilon) L$

$\leq (1+\varepsilon) OPT$

gain

Theorem: For any $\varepsilon > 0$, the above algorithm is an $(1+\varepsilon)$-approximation running in time $\text{poly}(n) \cdot 2^{O(1/\varepsilon)}$

Definitions

Polynomial Time Appx Scheme (PTAS):

An alg. which takes an extra parameter $\varepsilon > 0$ as input and returns an $(1+\varepsilon)$-factor appx in time which is a polynomial in n as long as ε is a constant.

Example running times: $n^{1/\varepsilon}$, $n^{2^{2^{1/\varepsilon}}}$, $n^2 * 2^{1/\varepsilon}$, n/ε

Fully polynomial Time Appx Scheme (FPTAS)

These are PTAS .. but their

These are PTASes, but their running times are poly$(n, 1/\varepsilon)$.

So, even if $\varepsilon \approx \frac{1}{n^{10}}$, the scheme runs in polynomial time.

An FPTAS is the best you can hope for for an NP-hard problem.

So our above theorem can be succinctly stated as

$$\boxed{\text{Thm: } P2||C_{max} \text{ has a PTAS}}$$

$P||C_{max}$ with many machines

① (Multiplicative Scaling)

"Round up" all job sizes p_j to the smallest power of $(1+\varepsilon)$ which is larger than p_j.

- $OPT(\mathcal{I}_{old}) \leq OPT(\mathcal{I}_{new}) \leq (1+\varepsilon) \cdot opt(\mathcal{I}_{old})$
- Given any soln to $\mathcal{I}_{new}$ of makespan

T $\Rightarrow$ a soln to I_{orig} of $\leq T$ mkspn.

② (Guessing OPT) Till now we assumed we didn't know OPT.... the number. But we keep working with a 'guess' and bump it by a factor $(1+\varepsilon)$ if the final soln. isn't within $\leq (1+\varepsilon)$ times guess.

③ (Bucketing Big Jobs)

note this is a power of $(1+\varepsilon)$..

- Job j is big if $p_j > \varepsilon \cdot OPT$

guess.

- B = Set of big jobs.
- $B_t \subseteq B$ with the same proc. time

$$1 \leq t \leq N$$

- How big is N? $\log_{1+\varepsilon}(1/\varepsilon)$

$$\approx \frac{1}{\varepsilon}\log\frac{1}{\varepsilon}$$

$\lg(1+x) \approx x$ if x is small...

④ (Profile of big jobs)

Every feasible schedule gives each machine a "profile"

$$\vec{V} = (v_1, v_2, \ldots, v_N)$$

where $v_t \in \mathbb{Z}_{\geq}$ denotes the # of jobs of B_t is given to this m/c.

Each big job $\geq \varepsilon \cdot OPT \Rightarrow \sum_{t=1}^{N} v_t \leq \frac{1}{\varepsilon}$

How many feasible profiles are there?

If $\mathbb{V}$ is collection of all feas. profiles,

$$M = |\mathbb{V}| \underset{\text{super crude}}{\leq} \left(\frac{1}{\varepsilon}\right)^N \approx \underbrace{2^{\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon}}}_{\text{is a constant, believe it or not.}}$$

⑤ (Histogram of profiles)

Let's order the profiles of $\mathbb{V}$ as

$$\vec{v}^{(1)}, \vec{v}^{(2)}, \dots, \vec{v}^{(M)}$$

Any feasible schedule allocates each machine one of these profiles.
Let h_i be the # of m/cs which pick the profile $\vec{v}^{(i)}$.

Here we use the "parallelness" of machines. It doesn't matter who chooses which of these h_i profiles of type $\vec{v}^{(i)}$

Consistency Conditions:

- $$\sum_{i=1}^{M} h_i = m$$
- $$\underbrace{\sum_{i=1}^{M} h_i \vec{v}_t^{(i)}}_{\text{\# of big jobs of type } B_t \text{ alloc. among the m/c's.}} = |B_t| \qquad \forall\ 1 \le t \le N$$

(6) (Guessing OPT's profile)

Let $(h_1^*, \dots, h_M^*)$ be the histogram of OPT. <u>Claim</u>: We can 'guess' h^* by enumeration.

How many consistent histograms are there?

$$\leq m^M$$

$$\leq m^{2^{\frac{1}{\varepsilon}\log^2\frac{1}{\varepsilon}}} \Big\} \text{ constant!}$$

Once we guess OPT's histogram, we give h_i^* <u>arbitrary</u> m/c's the profile $\vec{v}^{(i)}$. Since its consistent, all big jobs are allocated.

The small jobs we allocate GREEDily. Just as in the $P2||C_{max}$ case, we get an $(1+\varepsilon)\cdot OPT$ makespan.