CS420+500: Advanced Algorithm Design and Analysis

Lecture: March 17, 2017

Prof. Will Evans *Scribe: Stephen Kan*

In this lecture, we discussed approximation algorithms for:

- Minimum Vertex Cover
- List Scheduling Approximation

Suggested Reading: http://jeffe.cs.illinois.edu/teaching/algorithms/notes/31-approx.pdf

1 Minimum Vertex Cover

Definition. *Minimum vertex cover:* given an undirected graph $G=(V,E)$, find the smallest set of vertices $S \subseteq V$ such that all edges in G have at least one endpoint in S

Greedy Algorithm for Minimum Vertex Cover [$GreedyVC$]:

1. Include in S the vertex with the highest degree (largest number of connected edges)
2. Remove the vertex and all incident edges from G
3. Repeat Steps 1-2 until no edges are left in G

Guarantee: the size of $GreedyVC \leq \log n \bullet OPTVC$ for all graphs

Matching Vertex Cover [MVC]:

1. Set $S = \varnothing$
2. Pick any edge in the graph: $(u,v) \in G$
    a. Remove (u,v) from G and all edges that are adjacent to u or v
    b. Add u,v to S
3. Repeat 2 until no edges are left in G

Claim: MVC is a 2-approximation for minimum vertex cover

- We don't know how big $OPTVC(G)$ is but we can get the lower bound for its size

Proof:

1. $|OPTVC(G)| \geq \#edges$ picked by MVC because the edges form the matching; no vertex covers more than one edge in a matching
2. # vertices picked by MVC is $2 \times \#edges\ picked \rightarrow |MVC(G)| \leq 2|OPTVC(G)|$

2 List Scheduling Approximation

(1966 – Ronald Graham of Graham's Scan)

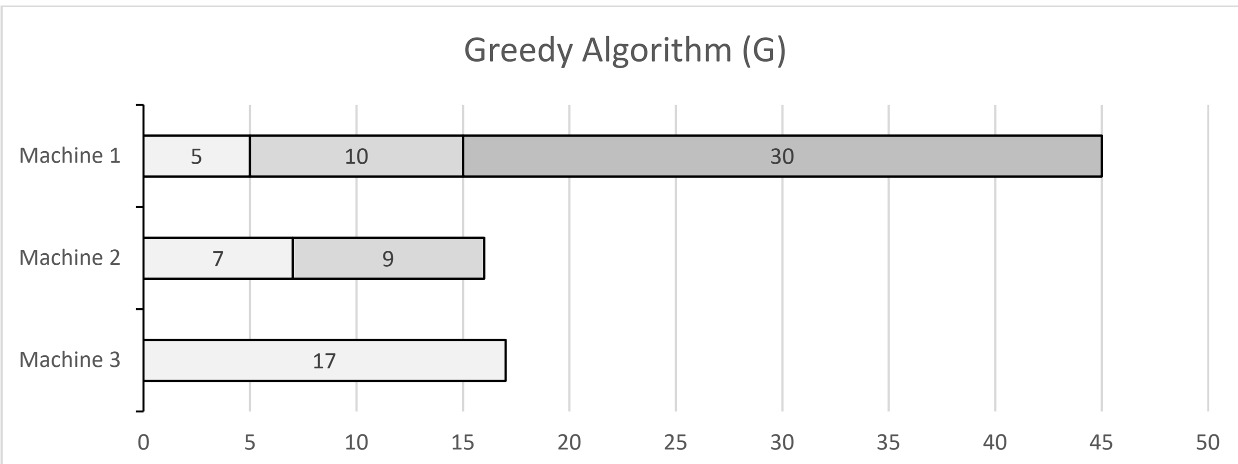
Definition. *List Scheduling:*

- Given n jobs, job i must execute uninterruptibly for P_i time units.
- Given m identical machines, each machine can work on one job at a time
- Find a schedule of jobs that minimizes the overall completion time

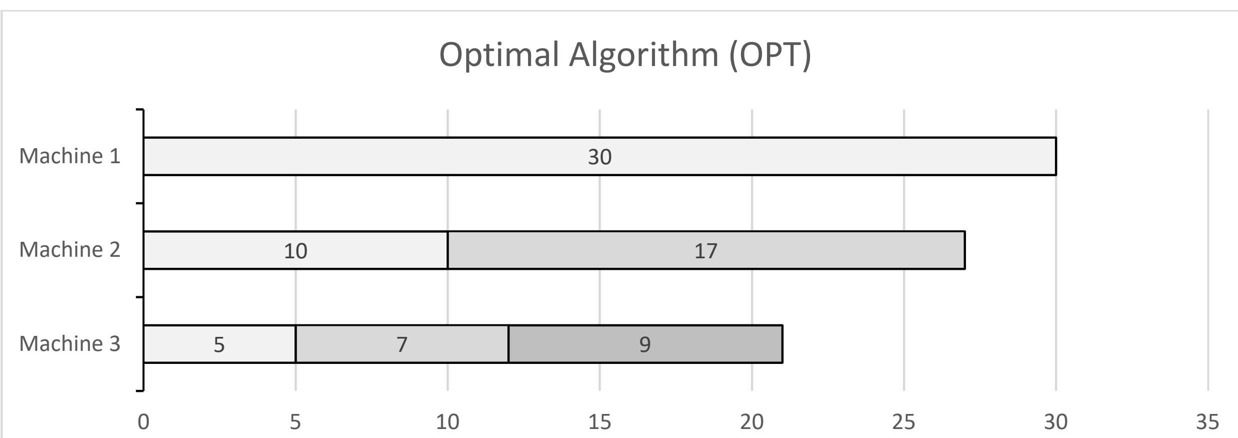
Example:

P_i : 5, 7, 17, 10, 9, 30: 6 jobs among 3 machines

Greedy Algorithm: Whenever a machine becomes idle, assign the next job to it.



Optimal Algorithm: the best possible job allocation



Another possible variant (not depicted): sort the jobs first from smallest to largest, then add to machines in reverse size order.

Greedy Algorithm (G):

Whenever a machine becomes idle, assign the next job to it.

Claim: $\left|G\left(P_1, P_2, \ldots, P_n\right)\right| \le \left(2 - \frac{1}{m}\right)\left|OPT\left(P_1, P_2, \ldots, P_n\right)\right|$ (A little better than 2-approximation)

Proof:

1. $OPT \ge P_i$ for all i
2. $OPT \ge \frac{1}{m}\sum_i P_i$

- Note: $\frac{1}{m}\sum_i P_i$ is the perfect division of P_i among machines, assuming jobs are interruptible

Let job k be the last job to finish. $P_k \le OPT$ by (1)

Goal: show that the sum of jobs before P_k on that machine is smaller than OPT, then the sum of the P_i s for all jobs on that machine is no more than $\left(2 - \frac{1}{m}\right)OPT$

1. Let S_k be the sum of the P_i s for all jobs on that machine before P_k.
2. Up to time S_k (when work starts on job k), all machines have been busy. That means the total amount of work that has been done up to time S_k is mS_k. This work is on jobs other than job k. So $\sum_{i \ne k} P_i \ge mS_k$ or, after rearranging, $S_k \le \frac{1}{m}\sum_{i \ne k} P_i$
3. Combining (1) and (2):

$$\begin{aligned} S_k + P_k &\le \frac{1}{m}\sum_{i \ne k} P_i + P_k = \frac{1}{m}\sum_i P_i + \left(1 - \frac{1}{m}\right)P_k \\ &\le OPT + \left(1 - \frac{1}{m}\right)OPT = \left(2 - \frac{1}{m}\right)OPT \end{aligned}$$

$\therefore$