25 April 2018 Approximation Algorithms (Chapter 11)

Efficient algorithms for (typically NP-Hard) optimization problems that may fail to produce an optimal solution, but one can prove they produce something "close to optimal".

Optimization Problem: Problems where the goal is to maximize or minimize some function of the input instance & the output called the objective function.

E.g. MAX INDEPENDENT SET problem: output is a subset of vertices of graph with no edge between any pair, objective function is cardinality of the set.

An approximation algorithm with approximation factor α (a.k.a. α-approximation algorithm) is an algorithm that outputs a solution $ALG(x)$ to input instance x, s.t.

$$\forall x \qquad ALG(x) \leq OPT(x) \leq \alpha \cdot ALG(x) \qquad \text{[maximization]}$$

defn of OPT approximation guarantee

$$\forall x \qquad OPT(x) \leq ALG(x) \leq \alpha \cdot OPT(x) \qquad \text{[minimization]}$$

Note $\alpha \geq 1$ for both maximization & minimization.

E.g. 2-approximation is "at least half of the opt" for maximization "at most twice the opt" for minimization.

Examples of Greedy Apx Algorithms and How to Analyze

Commentary 1: Typically what makes it hard to analyze an approximation algorithm is that in the end you must prove $OPT \leq \alpha \cdot ALG$ or $ALG \leq \alpha \cdot OPT$ but you don't know how to find OPT!!

Trick is to avoid directly comparing with this mystery quantity by comparing instead with some "surrogate value" ~~that~~ can easily be shown to be lower bound (or upper bound) on OPT.

E.g. $OPT \leq |V(G)|$ for graph problems that require maximizing cardinality of a vertex set, eg: IND SET.

Commentary 2. Although today's lecture is "introductory" the proofs in this lecture are the most ad hoc, hence hardest to come up with.

Example 1. Load Balancing.

Given jobs of sizes $s_1, s_2, \ldots, s_n$.
and m identical machines.
Partition jobs into sets $J_1, \ldots, J_m$ (J_i = jobs assigned to machine i)
to minimize max load on any machine.

$$\text{minimize} \quad \max_{1 \le i \le m} \left\{ \sum_{j \in J_i} s_j \right\}.$$

This is NP-hard, even when $m = 2$.
Reduction from SUBSET SUM: given $n_1, \ldots, n_K$ is there a subset that adds up to W?
Assume WLOG $W \le \frac{1}{2} N$. Else replace W with $N - W$.

$$s_1 = n_1 \quad s_2 = n_2 \quad \ldots \quad s_K = n_K \qquad \left(N := \sum_{i=1}^{K} n_i \right)$$

$$s_{K+1} = N - 2W$$

Can we partition the jobs into two sets J_1, J_2 s.t.

$$\max \left\{ \sum_{j \in J_1} s_j, \ \sum_{j \in J_2} s_j \right\} \le N - W$$

If such a partition exists, the two sums must both be equal to $N - W$. Because $s_1 + \cdots + s_{K-1} = 2N - 2W$, so

$$\sum_{j \in J_1} s_j + \sum_{j \in J_2} s_j = 2N - 2W$$

$$\max \{ \text{these 2 sums} \} \ge \frac{1}{2}(2N - 2W) = N - W.$$

Only way for $\max\{\text{two sums}\} = N - W$ is if the 2 sums are equal.
If the 2 sums are equal, then the jobs that accompany s_{K+1} in its piece of the partition sum up to W.
Converse: if $\sum_{j \in J} s_j = W$ then $J_1 = J \cup \{K+1\}$, $J_2 = \overline{J_1}$

and this attains max load $N - W$.

Greedy approx alg for Load balancing

Initialize $J_i = \emptyset$, $L_i = 0$ $\forall i = 1, \dots, m$

for $j = 1, 2, \dots, n$

  assign j to machine i with smallest load L_i.
  (break ties arbitrarily.)

  $J_i \leftarrow J_i \cup \{j\}$

  $L_i \leftarrow L_i + s_j$

endfor

$(m=2)$

Example where this is suboptimal: $s_1 = \frac{1}{2}$ $s_2 = \frac{1}{2}$ $s_3 = 1$.

$OPT = 1$ $J_1 = \{1, 2\}$, $J_2 = \{3\}$

Greedy instead does:

Iteration	Machine 1	Machine 2
1	$1/2$	
2		$1/2$
3	1	

Ends with loads $\{3/2, 1/2\}$ $ALG = 3/2$ while $OPT = 1$.

<u>Theorem.</u> For all input instances, $GREEDY \leq 2 \cdot OPT$.

<u>Proof.</u> Suppose machine i is the max loaded machine when you run GREEDY, and j is the last job that gets put into J_i.