CS 218 Design and Analysis of Algorithms

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Module 4: Coping with NP-hardness

Coping with NP-hardness

Understanding hard problems.

- Heuristics. No provable guarantees, but may work well in practice.
- Special cases and efficient algorithm for special cases.
- Better-than-brute-force algorithms.
- Approximation algorithms.
  - For optimization problems.
  - Maximization problem. Find a solution not less than OPT/c for some $c \geq 1$.
  - Minimization problem. Find a solution not more than $c \cdot \mathrm{OPT}$ for some $c \geq 1$.
  - A c-approximation algorithm. Ideally should run in polynomial time.

Approximation Algorithms

We will present an approximation algorithm for the scheduling problem.

Given: processors $p_1, \ldots, p_m$ and jobs $j_1, \ldots, j_n$ with durations $d_1, \ldots, d_n$ respectively

Find: a schedule for these jobs on m processors that minimises the total completion time.

Let $\mathcal{S}$ be a schedule with the following specifications.

Let A_i be the set of jobs scheduled on processor $i \in [m]$.

Let $T_i = \sum_{j \in A_i} d_j$.

Completion time $= \max_{i \in [m]} T_i$.

The problem is NP-complete. Even the version with 2 processors is NP-complete.

A greedy strategy is a 2-approximation

Recall one of the strategies we discussed.

Arrange the jobs in any arbitrary order.

Schedule jth job on the machine with the smallest load so far.

Let T be the completion time achieved by this schedule.

We know that there are instances where this will not be optimal.

Let T^* be the completion time for the OPT schedule.

We will show that $T \leq 2 \cdot T^*$.

A greedy strategy is a 2-approximation

Strategy for the proof of 2-approximation.

We want to show $T \leq 2T^*$. Without knowing what T^* is!

Suppose we can come up with some quantity T' such that

$T' \leq T^*$ but it is not too much smaller than T^*.

This will give a guarantee that, no matter how small T^* is, it cannot be smaller than T'.

Now we will show that $T \leq 2T'$.

This will finish the proof.

Lower-bounding T^*

Multiple ways of doing it.

$T^* \geq \frac{1}{m} \sum_{j \in [n]} d_j$ (A)

Can be quite weak. Can you think of a situation where it will be weak?

$T^* \geq \max_j d_j$. (B)

We will use both of these.

Proof of 2-approximation

We now prove that the greedy achieves a 2-approximation.

- Say ith processor be such that $T = T_i$.
- Let j be the last job to be scheduled on it.
- This means, each processor $i' \neq i$ had at least $T_{i'} \geq T_i - d_j$ on it when j was scheduled on i.
- Hence we have $\sum_{i \in [n]} T_i \geq m(T_i - d_j)$.

$$
\begin{aligned}
(T_i - d_j) &\leq \frac{1}{m} \sum_{i \in [n]} T_i \\
&\leq T^* && \text{(Using (A))}
\end{aligned}
$$

Proof of 2-approximation

We now prove that the greedy achieves a 2-approximation.

- Say ith processor be such that $T = T_i$.
- Let j be the last job to be scheduled on it.
- This means, each processor $i' \neq i$ had at least $T_{i'} \geq T_i - d_j$ on it when j was scheduled on i.
- Hence we have $\sum_{i \in [n]} T_i \geq m(T_i - d_j)$.

$$
\begin{aligned}
(T_i - d_j) &\leq \frac{1}{m} \sum_{i \in [n]} T_i \\
&\leq T^* && \text{(Using (A))}
\end{aligned}
$$

$$
d_j \leq T^* \qquad \text{(Using (B))}
$$

Proof of 2-approximation

We now prove that the greedy achieves a 2-approximation.

Say ith processor be such that $T = T_i$.

Let j be the last job to be scheduled on it.

This means, each processor $i' \neq i$ had at least $T_{i'} \geq T_i - d_j$ on it when j was scheduled on i.

Hence we have $\sum_{i \in [n]} T_i \geq m(T_i - d_j)$.

$$\begin{aligned} (T_i - d_j) &\leq \frac{1}{m} \sum_{i \in [n]} T_i \\ &\leq T^* && \text{(Using (A))} \\ \\ d_j &\leq T^* && \text{(Using (B))} \\ \therefore (T_i - d_j) + d_j &\leq 2T^* && \text{Using the above two} \end{aligned}$$