CS 341: Algorithms
Module 5: Greedy Algorithms

Armin Jamshidpey, Eugene Zima

Based on lecture notes by many previous CS 341 instructors

David R. Cheriton School of Computer Science, University of Waterloo

Spring 2019

Scheduling to minimize lateness

Given n jobs each requiring processing time t_i and has a deadline d_i. Scheduling assigns to each job start time s_i, such that jobs do not overlap.

Example:

assignments	time required	deadline
CS341	4 hrs	in 9 hrs
Math1000	2hrs	in 6 hrs
Philosophy	3 hrs	in 14 hrs
CS350	10 hrs	in 25 hrs

Define lateness l_i of job i to be $s_i + t_i - d_i$.

Find the schedule that minimizes $\max_i l_i$.

Note 1. It does not make sense to have breaks.
Note 2. Each job should be done continuously. (Why?)
Possible greedy strategy.

1. Schedule the short jobs first (find a counterexample).
2. Schedule the jobs with smaller "slack"($d_i - t_i$) first (find a counterexample).
3. Schedule jobs in increasing order of deadlines: sort them such that

$$d_1 \le d_2 \le \ldots \le d_n$$

   and schedule in this order. (Might work....)

Correctness.

In order to prove that greedy solution is optimal we use "exchange" argument. Supposed there is an optimal scheduling $i_1, i_2, \ldots, i_n$ which is not in the increasing deadline order. This means that there are two consecutive jobs i_j and i_{j+1} in the optimal scheduling such that $d_{i_j} > d_{i_{j+1}}$.
We exchange those two jobs to obtain new scheduling
$i_1, \ldots, i_{j-1}, i_{j+1}, i_j, i_{j+2} \ldots, i_n$
This only changes the finish time for two jobs i_{j+1} and i_j.

Suppose i_{j-1} finishes at time u. Then the lateness of i_j in the optimal scheduling is

$$l_{i_j} = u + t_{i_j} - d_{i_j},$$

and the lateness of i_{j+1} in the optimal scheduling is

$$l_{i_{j+1}} = u + t_{i_j} + t_{i_{j+1}} - d_{i_{j+1}}.$$

After exchange the lateness of i_{j+1} in new scheduling is

$$l^*_{i_{j+1}} = u + t_{i_{j+1}} - d_{i_{j+1}},$$

and the lateness of i_j in the new scheduling is

$$l^*_{i_j} = u + t_{i_j} + t_{i_{j+1}} - d_{i_j}.$$

Now, $l_{i_{j+1}} \geq l_{i_j}$, because $d_{i_j} > d_{i_{j+1}}$; $l_{i_{j+1}} \geq l^*_{i_{j+1}}$ and $l_{i_{j+1}} \geq l^*_{i_j}$. Therefore, changing the order of jobs i_j and i_{j+1} in the optimal scheduling we did not increase maximal lateness. Repeating exchange enough times we will get the scheduling in greedy order and it is not worse than the optimal non-greedy scheduling.

Knapsack Problems

Problem

Instance: Profits $P = [p_1, \ldots, p_n]$; weights $W = [w_1, \ldots, w_n]$; and a capacity, M. These are all positive integers.

Feasible solution: An n-tuple $X = [x_1, \ldots, x_n]$ where

$$\sum_{i=1}^{n} w_i x_i \leq M.$$

In the $0-1$ Knapsack problem (often denoted as Knapsack), we require that $x_i \in \{0, 1\}, 1 \leq i \leq n$.
In the Rational Knapsack problem, we require that $x_i \in \mathbb{Q}$ and $0 \leq x_i \leq 1, 1 \leq i \leq n$.

Find: A feasible solution X that maximizes $\sum_{i=1}^{n} p_i x_i$

Possible Greedy Strategies for Knapsack Problems

- Consider the items in decreasing order of profit (i.e., the local evaluation criterion is p_i).
- Consider the items in increasing order of weight (i.e., the local evaluation criterion is w_i).
- Consider the items in decreasing order of profit divided by weight (i.e., the local evaluation criterion is p_i/w_i).

Does one of these strategies yield a correct greedy algorithm for the Rational Knapsack problem?

A Greedy Algorithm for Rational Knapsack

Algorithm 1: GreedyRationalKnapsack(P,W: array; M: integer)

sort the items so that $p_1/w_1 \geq \ldots \geq p_n/w_n$
$X \leftarrow [0, \ldots, 0]$
$i \leftarrow 1$
$CurW \leftarrow 0$
while $(CurW < M)$ **and** $(i \leq n)$
    if $CurW + w_i \leq M$ **then**
        $x_i \leftarrow 1$
        $CurW \leftarrow CurW + w_i$
        $i \leftarrow i + 1$
    else
        $x_i \leftarrow (M - CurW)/w_i$
        $CurW := M$
return(X)

Correctness Proof

For simplicity, assume that the profit / weight ratios are all distinct, so

$$\frac{p_1}{w_1} > \frac{p_2}{w_2} > \cdots > \frac{p_n}{w_n}.$$

Suppose the greedy solution is $X = (x_1, \ldots, x_n)$ and the optimal solution is $Y = (y_1, \ldots, y_n)$.

We will prove that $X = Y$, i.e., $x_j = y_j$ for $j = 1, \ldots, n$. Therefore there is a unique optimal solution and it is equal to the greedy solution.

Suppose $X \neq Y$.

Pick the smallest integer j such that $x_j \neq y_j$.

It is impossible that $x_j < y_j$, so we have $x_j > y_j$.

There exists an index $k > j$ such that $y_k > 0$ (otherwise Y is not optimal).

Correctness Proof (cont.)

Let $\delta = \min\{w_k y_k, w_j(x_j - y_j)\}$; note that $\delta > 0$. Define

$$y_j' = y_j + \frac{\delta}{w_j} \text{ and } y_k' = y_k - \frac{\delta}{w_k}.$$

Then let Y' be Y with y_j and y_k updated to y_j' and y_k', respectively. The idea is to show that

- Y' is feasible, and
- profit(Y') >profit(Y).

This contradicts the optimality of Y and proves that $X = Y$.

Correctness Proof (cont.)

To Show Y' is feasible, show that $y'_k \geq 0$, $y'_j \leq 1$ and weight$(Y') \leq M$. First, we have

$$y'_k = y_k - \frac{\delta}{w_k} \geq y_k - \frac{w_k y_k}{w_k} = 0.$$

Second,

$$y'_j = y_j + \frac{\delta}{w_j} \leq y_j + \frac{w_j(x_j - y_j)}{w_j} = x_j \leq 1.$$

Third,

$$\text{weight}(Y') = \text{weight}(Y) + \frac{\delta}{w_j} w_j + \frac{\delta}{w_k} w_k = weight(Y) \leq M.$$

Finally, we compute

$$\text{profit}(Y') = \text{profit}(Y) + \frac{\delta p_j}{w_j} - \frac{\delta p_k}{w_k} = \text{profit}(Y) + \delta \left(\frac{p_j}{w_j} - \frac{p_k}{w_k} \right) > \text{profit}(Y),$$

since $\delta > 0$ and $\frac{p_j}{w_j} > \frac{p_k}{w_k}$