

Homework 1 Solutions

1. (a) This algorithm is not optimal.

Consider three jobs with starts $\{1, 1, 3\}$, ends $\{4, 2, 4\}$, and profits $\{3, 2, 2\}$. The given algorithm schedules the first job from time 1 to time 4, at a profit of 3, and must discard the other two jobs. The optimal solution schedules the other two jobs, for a profit of 4.

- (b) The given algorithm is indeed optimal.

First, observe that, without loss of generality, we need consider only schedules that use all slots $[0, 1, \dots, n - 1]$. Any schedule without this property can be changed by moving jobs back in time to fill any empty slots, which can only increase the number of jobs that meet their deadlines.

Second, we will consider the slightly more general problem in which the input is a list J of jobs *and* a list T of blocked slots. Formulating the general problem on $J \times T$ will make our substructure argument easier.

Greedy Choice Property: Let $J \times T$ be any problem instance. Let j_1 be the first job scheduled, and let t_1 be the slot in which it is scheduled. Then there exists an optimal solution to $J \times T$ that schedules j_1 at time t_1 .

Pf: Let S^* be any optimal solution. If S^* schedules j_1 at time t_1 , we are done. Otherwise, S^* puts j_1 at some time $t \neq t_1$ and schedules some other job $j_k \neq j_1$ at time t_1 .

Suppose we create a new solution S from S^* by moving the jobs j and j_1 to each others' slots. S makes the greedy choice, but we must show that it is still optimal. We proceed by cases:

- i. Suppose $t_1 \leq d_1$, and that j_1 incurs no penalty in S^* . Then $t < t_1$, since j_1 cannot be scheduled without penalty in any slot $> t_1$. Conclude that exchanging j_k with j_1 leaves j_1 without penalty and moves j_k to an earlier slot, which leaves its penalty the same or less than before. Hence, S 's penalty is no greater than that of S^* .
- ii. Suppose $t_1 \leq d_1$, and that j_1 incurs a penalty in S^* . Then $t > t_1$, and the exchange moves j forward. The total penalty of S is therefore at most that of S^* plus $p_k - p_1$, since j_k may miss its deadline in S , but j_1 does not. By the greedy choice, $p_1 \geq p_k$, so S 's penalty remains at most that of S^* .
- iii. Suppose $t_1 > d_1$. Then there is *no* free slot in which to schedule j_1 without penalty, and so j_1 incurs a penalty in every solution to $J \times T$. Moreover, by the greedy choice, t_1 is the latest free slot, so $t < t_1$. Hence, exchanging j_k with j_1 does not change the penalty for j_1 and may eliminate the penalty for j_k . Hence, S 's penalty remains at most that of S^* .

Conclude that S is always at least as good as S^* and so is optimal. QED

Substructure 1: After making the greedy choice, we are left with a smaller instance of the same problem.

Pf: Let $J' = J - \{j_1\}$, and let $T' = T \cup \{t_1\}$. Then $J' \times T'$ is a smaller instance of the original problem. QED

Substructure 2: Let S' be an optimal solution to the subproblem $J' \times T'$. Then the solution S that adds j_1 at time t_1 to S' optimally solves the original problem $J \times T$.

Pf: First, since T' blocks slot t_1 , we can add j_1 at time t_1 to S' . Second, observe that

$$\text{penalty}(S) = \text{penalty}(S') + p(j_1, t_1),$$

where $p(j_1, t_1)$ is the penalty incurred by placing j_1 at t_1 . The usual contradiction argument completes the proof. QED

- Our greedy algorithm is as follows: beginning at time S , repeatedly select a babysitter who overlaps the covered time so far and whose interval extends furthest in time toward F (i.e. the b_i with maximum f_i).

Here is some pseudocode that takes a set B of sitters b_i and an interval $[S, F)$ and selects sitters according to the above rule. It will break if no sitter is available to cover some time between S and F , but we'll ignore that case here.

```

HIRE( $B, [S, F)$ )
  sort  $B$  in nondecreasing order of  $s_i$ 
   $T \leftarrow \emptyset$  // solution set
   $i \leftarrow 1$ 
   $e \leftarrow S$ 
  while  $e < F$ 
     $f_i^* = S$ 
    while  $s_i < e$ 
      if  $f_i > f_i^*$ 
         $f_i^* \leftarrow f_i$ 
         $i^* \leftarrow i$ 
       $i++$ 
     $T \leftarrow T \cup \{b_{i^*}\}$ 
     $e \leftarrow f_{i^*}^*$ 
  return  $T$ 

```

The code looks at each sitter exactly once. For each successive end e of the covered part of $[S, F)$, it considers all sitters who start before e and chooses the one that ends furthest to the right. That sitter is added to the solution, and the endpoint e is updated. The sitters not chosen all end at time $\leq e$, so they may subsequently be ignored.

The code spends $\Theta(1)$ time per sitter, so the main loop is $\Theta(n)$. The dominant cost of the algorithm is therefore the sort, making it $\Theta(n \log n)$ overall.

Greedy Choice Property: Let $B, [S, F)$ be an instance of the babysitting problem, and let b_i be a sitter with $s_i \leq S$, such that f_i is maximal for b_i among all such sitters. Then there exists an optimal solution containing b_i .

Pf: Let T^* be any optimal solution to $B, [S, F)$. If $b_i \in T^*$, we are done. Otherwise, let b_j be the sitter in T^* with minimal f_j ; b_j must have $s_j \leq S$. Construct a new solution T from T^*

by discarding b_j and adding b_i . Because $f_i \geq f_j$ and $s_i \leq S$, the new solution T still covers the interval $[S, f_j]$. Moreover, T contains just as many sitters as T^* , so T is still optimal. QED

Substructure 1: after making the greedy choice b_i , we are left with a subproblem $B', [f_i, F)$, where B' is B with b_i and possibly some other b_j 's removed. This problem has no extra constraints compared to the original. (Note that we don't bother removing *all* useless sitters from B , since the problem does not specify that such sitters cannot exist, and the algorithm discards them provided that at least one non-useless sitter is found.)

Substructure 2: Let T' be an optimal solution to $B', [f_i, F)$, and let $T = T' \cup \{b_i\}$, where b_i is the greedy choice. Then T is optimal for $B, [S, F)$.

Pf: First, since b_i runs from $\leq S$ to f_i , T is a feasible solution, since it spans from S to F . Second, observe that $|T| = |T'| + 1$. The usual contradiction argument follows. QED

3. (a) Consider the problem instance with measured masses $\{3, 6\}$ and known masses $\{5, 10\}$. The given algorithm would match 6 with 5 and 3 with 10, for a total cost of 8. A better solution matches 3 with 5 and 6 with 10, for a total cost of 6.
- (b) Here is an algorithm for this problem. First, sort the measured masses in increasing order $\{x_1, x_2, \dots\}$ by size, and sort the known masses in increasing order $\{w_1, w_2, \dots\}$ by size. Then, assign the i th measured mass to the i th known mass. This algorithm is clearly $\Theta(n \log n)$, since it takes this much time to sort each list and $\Theta(n)$ to do the assignment.

Greedy Choice Property: Let (X, W) be an instance of the problem, with X the measured and W the known masses. Then there exists an optimal assignment that pairs the smallest measured mass x_1 to the smallest known mass w_1 .

Pf: let A^* be an optimal assignment of measured to known masses. If A^* pairs x_1 with w_1 , we are done. Otherwise, A^* pairs x_1 with some other w' and w_1 with some other x' . Note that, by construction, $x' \geq x_1$ and $w' \geq w_1$.

Let A be the assignment obtained from A^* by swapping these pairs, so that the greedy choice is made and x' is paired with w' . We claim that A 's cost is at most that of A^* . Observe that

$$\text{cost}(A) = \text{cost}(A^*) - |x_1 - w'| - |x' - w_1| + |x_1 - w_1| - |x' - w'|.$$

We need to show that the difference $\text{cost}(A) - \text{cost}(A^*)$ is zero or negative.

There are six possible orderings of the four values $x_1, w_1, x',$ and w' that are consistent with the inequalities $x_1 \leq x'$ and $w_1 \leq w'$. We will consider three of them; the other three are symmetric (just swap the roles of x and w):

- i. If $x_1 \leq x' \leq w_1 \leq w'$, then we have that

$$\begin{aligned} |x_1 - w_1| + |x' - w'| - |x_1 - w'| - |x' - w_1| &= w_1 - x_1 + w' - x' - (w' - x_1) - (w_1 - x') \\ &= 0. \end{aligned}$$

- ii. If $x_1 \leq w_1 \leq x' \leq w'$, then we have that

$$\begin{aligned} |x_1 - w_1| + |x' - w'| - |x_1 - w'| - |x' - w_1| &= w_1 - x_1 + w' - x' - (w' - x_1) - (x' - w_1) \\ &= 2(w_1 - x') \\ &\leq 0. \end{aligned}$$

iii. If $x_1 \leq w_1 \leq w' \leq x'$, then we have that

$$\begin{aligned} |x_1 - w_1| + |x' - w'| - |x_1 - w'| - |x' - w_1| &= w_1 - x_1 + x' - w' - (w' - x_1) - (x' - w_1) \\ &= 2(w_1 - w') \\ &\leq 0. \end{aligned}$$

Conclude that, in every case, A 's cost is at most that of A^* , and so A is also optimal. QED

Substructure 1: After making the greedy choice, we are left with a smaller instance of the same problem.

Pf: Let (X, W) be the original problem instance. After making the greedy choice, set $X' = X - \{x_1\}$ and $W' = W - \{w_1\}$. Then we are left with a subproblem (X', W') of matching $n - 1$ measured to $n - 1$ known masses. QED

Substructure 2: Suppose A' is an optimal assignment for subproblem (X', W') ; then $A = A' \cup (x_1, s_1)$ is an optimal solution to the original.

Pf: The cost of A can be decomposed as

$$\text{cost}(A) = \text{cost}(A') + |x_1 - s_1|.$$

Apply the usual contradiction argument. QED

4. Let G be our graph of precedence constraints. Say that a job j_k is *released* if there are no edges $j_k \rightarrow j_i$ in G . (This may be the opposite of what you expected; it is not a typo.) Also, let $T = \sum_i p_i$, the total duration of all jobs in the problem.

Our algorithm is as follows. Make a list of all released jobs, and for each such job j_k , compute its anger $f_k(T)$ at time T . Schedule the least angry such job j^i , to end at time T , and remove j_i and all its inbound edges from G . Finally, set $T \leftarrow T - p_i$ and repeat.

For each job scheduled, the algorithm spends $O(n)$ time computing anger values and finding the least angry job at time T . We can store G as an adjacency matrix, to make it easy to find every edge into a job j_k , and we can maintain for each job a count of its incoming edges. The counts for all remaining jobs can be updated in $O(n)$ total time each time a job is scheduled, and all released jobs can then be found in time $O(n)$. There are n scheduling passes, so the total cost of the algorithm is $O(n^2)$.

For correctness, we first observe that WLOG, an optimal schedule contains no gaps in which no job runs. For any schedule with gaps, we can fill them by moving all jobs after the gap back, which cannot increase their anger. hence, we may formulate the general problem as scheduling a set P of jobs in the interval $[0, T]$, where T is defined as above, with precedence constraints G .

Greedy Choice Property: Let j_i be the job with least anger when it ends at time T . Then there exists an optimal solution to problem (P, T, G) that schedules j_i at time T .

Pf: Let S^* be any optimal solution. If S^* schedules j_i at time T , we are done. Otherwise, S^* schedules some other job to end at time T and schedules j_i to end at some time $t_i < T$.

We construct a new solution S from S^* by moving j_i to the end of the schedule, then sliding it and all jobs after j back to close the resulting gap. Since T is the sum of all job durations, S

makes the greedy choice. Moreover, S preserves all the precedence constraints of G , since no job other than j_i is moved in the ordering of S , and no later job depends on j_i by construction.

We claim that the maximum anger of S is at most that of S^* . Let j_a be the “angriest” job in S ; that is, j_a ends at some time t_a in S , and $f_a(t_a)$ is maximal among all jobs. Note that $\text{cost}(S) = f_a(t_a)$.

If $j_a \neq j_i$, then j_a was at least as angry in S^* as it is in S , since it started later. Hence, the cost of S^* is at least that of S . If $j_a = j_i$, let j_k be the job that runs last in S^* . We have that

$$\text{cost}(S) = f_i(T) \leq f_k(T) \leq \text{cost}(S^*).$$

because j_i 's anger at time T is at most that of any other job. Conclude that in both cases, S costs no more than S^* and so is also optimal. QED

Substructure 1: after making the greedy choice j_i , we are left with a subproblem (P', T', G') of the original problem.

Pf: Set $P' \leftarrow P - \{j_i\}$, $T' \leftarrow T - p_i$, and $G' = G - \{j_i\}$. The result is a scheduling problem for all jobs P' with constraints on the interval $[0, T']$.

Substructure 2: let S' be an optimal schedule for (P', T', G') . Then adding j_i ending at time T to S' yields an optimal schedule S for (P, T, G) .

Pf: observe that

$$\text{cost}(S) = \max(\text{cost}(S'), f_i(T)).$$

It is not completely obvious that the usual contradiction argument applies, so let's do it.

Suppose S is not optimal, and let S^* be an optimal solution that schedules j_i at time T . By definition, $\text{cost}(S)$ is either $\text{cost}(S')$ or $f_i(T)$. We first observe that, in fact, $\text{cost}(S) = \text{cost}(S')$. Suppose not; then we have that

$$\text{cost}(S) = f_i(T) \leq \text{cost}(S^*)$$

because S^* also puts job j_i at time T , and this would contradict the suboptimality of S .

Finally, if $\text{cost}(S^*) < \text{cost}(S) = \text{cost}(S')$, then the subsolution of S^* obtained by removing job j_i , which solves (P', T', G') , is also better than $\text{cost}(S')$, contradicting the optimality of S' . QED