CS1510 Greedy Problems 8 & 12

Rebecca Negley, Sean Myers

September 12, 2011

8. Consider the following problem. The input consists of n skiers with heights $p_1,\ldots,p_n$, and n skis with heights $s_1,\ldots,s_n$. The problem is to assign each skier a ski to minimize the average difference between the height of a skier and his/her assigned ski. That is, if the i^{th} skier is given the $\alpha(i)^{th}$ ski, then you want to minimize:

$$\frac{1}{n}\sum_{t=1}^{n}|p_t - s_{\alpha(t)}|$$

   (a) Consider the following greedy algorithm. Find the skier and ski whose height difference is minimized. Assign this skier this ski. Repeat the process until every skier has a ski. Prove or disprove that this algorithm is correct.

   Solution: Consider two skiers with heights 4 and 7 and two skis with heights 6 and 9. The minimal difference first algorithm will first pair the height 7 skier with the height 6 ski. Then, the height 4 skier will be paired with the height 9 ski. This gives an average height difference of $\frac{1}{2}(|7-6|+|4-9|) = \frac{1}{2}(1+5) = 3$. However, if we pair the height 4 skier with the height 6 ski and the height 7 skier with the height 9 ski, we get an average height difference of $\frac{1}{2}(|4-6|+|7-9|) = \frac{1}{2}(2+2) = 2$. Thus, the minimal difference first algorithm does not produce an optimal solution for every input and hence does not solve the problem.

   (b) Consider the following greedy algorithm. Give the shortest skier the shortest ski, give the second shortest skier the second shortest ski, give the third shortest skier the third shortest ski, etc. Prove or disprove that this algorithm is correct.

   Solution: Suppose there exists an input I for which the shortest to shortest (STS) algorithm fails to give an optimal solution. Suppose there exists an optimal solution $OPT(I)$. Order both $STS(I)$ and $OPT(I)$ in order of shortest skier height to tallest skier height. Let k be the first position in this order where $OPT(I)$ and $STS(I)$ differ. Suppose $OPT(I)$ chooses ski height s_a for the k^{th} shortest skier, and $STS(I)$ chooses ski height s_b for the k^{th} shortest skier. We know that, in the optimal solution, ski height s_b is given to some skier after the k^{th} position because it matches $STS(I)$ for the first $k-1$ steps. Similarly, ski height s_a is given to some skier after the k^{th} position in $STS(I)$.

   Construct a new solution $OPT'(I)$ that initially agrees with $OPT(I)$. Then, swap s_a with s_b in $OPT'(I)$. It is apparent that $OPT'(I)$ agrees with $STS(I)$ for one more step. We must show that $OPT'(I)$ is still optimal. By the definition of $STS(I)$, $s_b \leq s_a$. Say p_i is the skier height in position k, and p_j is the skier height originally matched with s_b in $OPT(I)$ (so that p_j is now matched with s_a in $OPT'(I)$). Since p_j comes after p_i in the shortest to tallest ordering, $p_i \leq p_j$.

These two inequalities give that one of the following must hold:

$$(0 <)p_i \leq p_j \leq s_b \leq s_a \quad (1)$$
$$(0 <)p_i \leq s_b \leq p_j \leq s_a \quad (2)$$
$$(0 <)p_i \leq s_b \leq s_a \leq p_j \quad (3)$$
$$(0 <)s_b \leq p_i \leq p_j \leq s_a \quad (4)$$
$$(0 <)s_b \leq p_i \leq s_a \leq p_j \quad (5)$$
$$(0 <)s_b \leq s_a \leq p_i \leq p_j \quad (6)$$

Since (average height difference of $OPT'(I)$)=(average height difference of $OPT(I)$)$-(|p_i - s_a| + |p_j - s_b|) + (|p_i - s_b| + |p_j - s_a|)$, showing

$$|p_i - s_a| + |p_j - s_b| \geq |p_i - s_b| + |p_j - s_a| \quad (7)$$

is sufficient to show $OPT'(I)$ is still an optimal solution.

If $|p_i - s_a| \geq |p_i - s_b|$ and $|p_j - s_b| \geq |p_j - s_a|$, we have (7) automatically.
Suppose $|p_i - s_a| < |p_i - s_b|$. Then, we must have (4), (5), or (6). If (4), $|p_j - s_b| + |p_i - s_a| = (p_j - p_i) + (p_i - s_b) + (s_a - p_j) + (p_j - p_i) = |p_i - s_b| + |p_j - s_a| + 2|p_j - p_i| \geq |p_i - s_b| + |p_j - s_a|$, so we have (7). If (5), $|p_j - s_b| + |p_i - s_a| = (p_j - s_a) + (s_a - p_i) + (p_i - s_b) + (s_a - p_i) = |p_j - s_a| + |p_i - s_b| + 2|p_i - s_a| \geq |p_i - s_b| + |p_j - s_a|$, so we again have (7). If(6), $|p_j - s_b| + |p_i - s_a| = (p_j - s_a) + (s_a - s_b) + (p_i - s_a) = (p_j - s_a) + (p_i - s_b) = |p_j - s_a| + |p_i - s_b|$. Since the two sides are equal, (7) is true. Therefore, $|p_i - s_a| \geq |p_i - s_b|$ implies (7).
Suppose $|p_j - s_b| \geq |p_j - s_a|$ Then, we must have (1), (2), or (3). If (1), $|p_j - s_b| + |p_i - s_a| = (s_b - p_j) + (s_a - s_b) + (s_b - p_i) = (s_a - p_j) + (s_b - p_i) = |p_j - s_a| + |p_i - s_b|$. Since the two sides are equal, we have (7). If (2), $|p_j - s_b| + |p_i - s_a| = (p_j - s_b) + (s_a - p_j) + (p_j - s_b) + (s_b - p_i) = |p_j - s_a| + |p_i - s_b| + 2|p_j - s_b| \geq |p_j - s_a| + |p_i - s_b|$, so we have (7). If (3), $|p_j - s_b| + |p_i - s_a| = (p_j - s_a) + (s_a - s_b) + (s_a - s_b) + (s_b - p_i) = |p_j - s_a| + |p_i - s_b| + 2|s_a - s_b| \geq |p_j - s_a| + |p_i - s_b|$, so we have (7). Therefore, $|p_j - s_b| \geq |p_j - s_a|$ implies (7).

Since we must have both $|p_i - s_a| \geq |p_i - s_b|$ and $|p_j - s_b| \geq |p_j - s_a|$ or at least one of $|p_i - s_a| < |p_i - s_b|$ and $|p_j - s_b| \geq |p_j - s_a|$, and each of these things implies (7), we know that (7) must hold. Hence, $OPT'(I)$ is still an optimal solution and matches $STS(I)$ for one more step. We can continue to construct $OPT''(I)$, $OPT'''(I)$, ... until we find an optimal solution that is identical to $STS(I)$. Hence, we contradict that fact that STS is not optimal for I, so the exchange argument gives that the STS algorithm is correct.

12. We consider the following scheduling problem:

INPUT: A collection of jobs J_1,...,J_n, where the i^{th} job is a tuple (r_i, x_i) of non-negative integers specifying the release time and size of the job.
OUTPUT: A preemptive feasible schedule for these jobs on one processor that minimizes the total completion time $\sum_{t=1}^{n} C_1$.

Consider greedy algorithms of the following form:
$t = 0$
While there are jobs left not completely scheduled:

> Among those jobs J_1 such that $r_1 \leq t$, and that have previously been scheduled for less than x_1 time units, pick a job J_m to schedule at time t according to some rule.
> Increment t.

(a) **SJF:** Pick J_m to be the job with minimal size x_t. Ties may be broken arbitrarily.

Solution: Consider the input $I = \{J_1 = (0, 70), J_2 = (65, 60)\}$. SJF would pick job J_1 to run from $t = 0$ to $t = 65$. Since J_2 has a smaller size than J_1, SJF would then pick J_2 to run from $t = 65$ to $t = 125$, so J_2 would complete at time $C_2 = 125$. J_1 would run from $t = 125$ to $t = 130$ and complete at time $C_1 = 130$. Therefore, the total completion time of $SJF(I)$ is $130 + 125 = 255$. However, if J_1 runs from $t = 0$ to $t = 70$, $C_1 = 70$. Then, J_2 can run from $t = 70$ to $t = 130$, giving $C_2 = 130$. Then, the total completion time would be $70 + 130 = 200$. Since $200 < 255$, $SJF(I)$ is not optimal. Hence, the SJF algorithm does not solve the problem.

(b) **SRPT:** Let $y_{1,t}$ be the total time that job J_1 has been run before time t. Pick J_m to be a job that has minimal remaining processing time, that is, that has minimal $x_1 - y_{1,t}$. Ties may be broken arbitrarily.

Solution: Suppose there exists an input I for which $SRPT(I)$ is not an optimal solution. Select an optimal solution, $OPT(I)$. Examine first time $t = k$ where $SRPT(I)$ and $OPT(I)$ differ. Say $OPT(I)$ runs job J_a at time k, and $SRPT(I)$ runs job J_b at time k. We know that both J_a and J_b are unfinished in both solutions before time $t = k$ because both solutions agree for the first $k-1$ steps. Therefore, there must be some time $j > k$ such that $OPT(I)$ runs job J_b at $t = j$.

Create a new solution $OPT'(I)$ that is initially identical to $OPT(I)$. Starting at time k and increasing in time, find and label all remaining time units $a_1,\ldots,a_l$ where $OPT(I)$ runs job J_a (so $l = x_a - y_{a,k}$). Similarly, find and label all remaining time units $b_1,\ldots,b_m$ where $OPT(I)$ runs job J_b (so $m = x_b - y_{b,k}$). We know $x_b - y_{b,k} \leq x_a - y_{a,k}$ by the definition of $SRPT$, so $m \leq l$. Observe that, by the way these series are constructed, $k = a_1 < a_2 < \ldots < a_l$ and $k < b_1 < b_2 < \ldots < b_m$.

If $b_m < a_l$, swap the contents of a_1 and b_1 in $OPT'(I)$ and leave every other interval identical to $OPT(I)$. We know $a_1 < b_1 \leq b_m < a_l$, so $a_1 < a_l$ and $b_1 < a_l$. Therefore, job J_a will complete at the same time ($t = a_l$) in $OPT'(I)$ as it did in $OPT(I)$. If $b_1 \neq b_m$, job J_b will complete at the same time ($t = b_m$) in $OPT'(I)$ as in $OPT(I)$. In this case, since no other jobs were shifted, $OPT'(I)$ will have the same total completion time as $OPT(I)$ and hence still be optimal while agreeing with $SRPT(I)$ for one more step. If $b_1 = b_m$, job J_b will complete at an earlier time ($t = a_1 < b_1 = b_m$) in $OPT'(I)$ than in $OPT(I)$. Since all other completion times remain the same, this would mean the total completion time of $OPT'(I)$ is less than the total completion time of $OPT(I)$. Since $OPT(I)$ is an optimal solution, $b_1 = b_m$ gives a contradiction and is not possible.

If $b_m > a_l$, swap the contents of a_1 with b_1, a_2 with b_2, ..., a_m with b_m in $OPT'(I)$. (We have from above that $m \leq l$, so $a_m \leq a_l$). Then, job J_a now completes at time b_m instead of time a_l (since b_m contains job J_a and is larger than a_l). Importantly, the completion time of job J_a in $OPT'(I)$ is the same as the completion time of J_b in $OPT(I)$. Job J_b now finishes at time $a_m \leq a_l$. (Not that a_l is the completion time of J_a in $OPT(I)$). All other jobs are not shifted and finish at the same time in both $OPT(I)$ and $OPT'(I)$, so (completion time of $OPT'(I)$)= (completion time of $OPT(I)$)$-(a_l + b_m) + (a_m + b_m)$ =(completion time of $OPT(I)$)$-(a_l - a_m)$ $\leq$(completion time of $OPT(I)$). If $m < l$, (completion time of $OPT'(I)$)$<$(completion time of $OPT(I)$). This contradicts the fact that $OPT(I)$ is an optimal solution, so we must have $m = l$ in this case. Then, $OPT'(I)$ is still optimal and agrees with $SRPT(I)$ for one more step.

Thus, in all cases where $OPT(I)$ is an optimal solution, we can construct an $OPT'(I)$ that is still an optimal solution and agrees with $SRPT(I)$ for one more step. We can continue in this pattern to construct $OPT''(I)$, $OPT'''(I)$, ... until we have an optimal solution that matches

$SRPT(I)$. Therefore, the exchange argument gives us a contradiction that there exists some input for which $SRPT$ fails to give an optimal solution. Hence, the $SRPT$ algorithm is correct.