CS 1510 Homework 5

Brian Falkenstein, Brian Knotten, Brett Schreiber

September 6, 2018

11

a

SJF is not optimal on the following input I:
$J_1 = (0, 2)$
$J_2 = (1, 2)$
For each time t:

1. Run J_1 (it is the only choice)
2. Run J_2 (arbitrarily, since J_1 and J_2 are the same size.)
3. Run J_2 (arbitrarily). J_2 is completed now, so $C_2 = 3$
4. Run J_1 (it is the only choice). J_1 is completed now, so $C_1 = 4$

The total completion time for $SJF(I)$ is $C_1 + C_2 = 4 + 3 = 7$
But a more optimal solution $opt(I)$ exists. For each time t:

1. Run J_1
2. Run J_1. J_1 is now finished, so $C_1 = 2$.
3. Run J_2.
4. Run J_2. J_2 is now finished, so $C_2 = 4$.

The total completion time for $opt(I)$ is $C_1 + C_2 = 2 + 4 = 6$
$SJF(I)$ is not optimal therefore SJF is incorrect.

b

This proof does not consider the possibility of job j completing between times t and u. If this were the case, and j was initially scheduled to complete before time u, moving it back to u will increase its completion time, making its output not optimal.

c

Assume that A, the algorithm that implements $SRPT$, is incorrect and has some input I that makes it give the incorrect output. Define $Opt(I)$ to be the correct output that agrees with $A(I)$, the output from A on I, for the most steps. Also define the first "step", or time interval that $A(I)$ and $Opt(I)$ disagree, to be t. At time t, say that $A(I)$ schedules job J_A with tuple (r_A, x_A), and $Opt(I)$ schedules job J_O with tuple (r_O, x_O). We can construct $Opt'(I)$ by simply swapping J_O with the next instance of $Opt(I)$ scheduling J_A, say at time u.
$Opt'(I)$ clearly agrees with $A(I)$ for one more step, as it now also schedules J_A at time t. The problem addressed in part b of this problem can also be disproven here. We know that J_A has a shorter time left until completion than J_O, because of the definition of A. Thus, if J_O were to complete between times t and u, we could simply swap the entirety of J_A into the spots that J_O is ran, and have it complete even earlier, lowering the total completion time. The increased completion time added by having to shift J_O down would be less than the decreased time from completing J_A earlier, because J_O could also be scheduled into the additional slots that J_A does not need.

17

Consider the following algorithm A:
Given rooted tree T:

  Sort the leaves by value in descending order.
  For each leaf l (from greatest value to least):
    If any ancestor's current capacity is less than 0, do not include l in the output
    Otherwise, include l, and for each ancestor of l set its new capacity to be its current capacity - 1

Proof: Assume to reach a contradiction that our algorithm, hereafter referred to as A, is incorrect. Then there exists an input I on which A produces a suboptimal output. Let $OPT(I)$ be the optimal solution to the problem that agrees with $A(I)$ for the max number of steps i.e. up to leaf n $OPT(I)$ and $A(I)$ have included and excluded the same leaves. Because each step is either including or excluding a leaf, the disagreement can be one of two cases:

1. $A(I)$ excluded leaf n and $OPT(I)$ included leaf n:
   Because $A(I)$ and $OPT(I)$ agreed on every leaf up to this step and A always considers the next highest possible value and only excludes leaves when a leaf's ancestors' capacity does not accommodate it, it cannot be the case that $OPT(I)$ includes n and $A(I)$ does not.

2. $A(I)$ included leaf n and $OPT(I)$ excluded leaf n:
   $OPT(I)$ excludes a leaf that $A(I)$ includes either because the parent nodes are at capacity or because there is a better leaf it will select later on. The first case cannot occur, as $OPT(I)$ and $A(I)$ have the same capacity because they have agreed on every leaf up to this point. In the second case, because A always selects the next highest possible value and $OPT(I)$ has agreed with $A(I)$ up to this point, it is impossible for there to be a leaf with greater value than n available for $OPT(I)$ to select later on. Therefore the leaf $OPT(I)$ would select instead of n has to be of equal or lesser value than n and $OPT(I)$'s solution is equivalent to or lesser than $A(I)$'s solution.

Therefore $OPT(I)$ can be modified into $OPT'(I)$ where $OPT'(I)$ agrees with $A(I)$ for one more step (including or excluding n), contradicting the statement that $OPT(I)$ is the optimal solution that agrees with $A(I)$ for the most number of steps.
Thus, by contradiction, $A(I)$ is correct.