Proof for Optimality of Earliest Finish Time Algorithm for Interval Scheduling

T. M. Murali

Suppose that A is the set of jobs computed by the Earliest First Time (EFT) algorithm and that A has k jobs. We can sort the jobs in non-decreasing order of finish time.[1] Let $i_i, i_2, i_3, \dots i_{k-1}, i_k$ be the jobs in this order. Because of the way we sorted them, we know that for every $1 \leq t \leq k-1$,[2] $f(i_t) \leq f(i_{t+1})$.

Now suppose that the algorithm has not produced an optimal solution. Then there must some other set O of jobs with $m > k$ jobs. Since O is a solution to the problem, the jobs in it are mutually compatible. We can sort the jobs in O by finish time as well.[3] Let $j_i, j_2, j_3, \dots j_{m-1}, j_m$ be the jobs in these order. Because of the way we sorted them, we know that for every $1 \leq t \leq m-1$, $f(i_t) \leq f(i_{t+1})$.

The key idea now is to compare the jobs at the same index in A and O. They must have different jobs[4] at *some* index; otherwise, both A and O would be the same, meaning the algorithm is optimal! Let p be the first index at which they are different, i.e., for every index $q < p$, $i_q = j_q$ but $i_p \neq j_p$. Note that it is possible that $p = 1$. What we will do is to replace j_p with i_p in O and show that O still contains a compatible set of jobs. Thus, the smallest index at which A and O differ "bubbles" up by at least one index. There are three cases to consider.

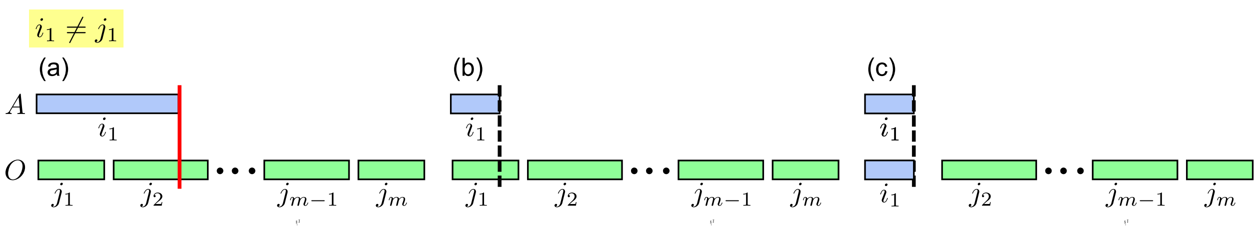

Figure 1: The case when $i_1 \neq j_1$. We show only the job i_1 in A. Black dots indicate intermediate jobs. (a) Can the finish time of i_1 be larger than the finish time of j_1 (potentially causing i_1 to conflict with j_2)? (b) No! The reason is that i_1 is the first job selected by the EFT algorithm. Hence, its finish time must be less than or equal to the finish time of j_1. (c) Therefore, if we replace j_1 with i_1 in O, all the jobs in O continue to be mutually compatible.

Case 1: $i_1 \neq j_1$. As a warm-up, let us consider an easy case first. Suppose $i_1 \neq j_1$ (Figure 1). Then we can start making some interesting observations. Since i_1 was the first job selected by the EFT algorithm, its finish time must be the smallest among all the jobs in the input. Therefore, we can be sure that

$$f(i_1) \leq f(j_1),$$

i.e., the situation illustrated in Figure 1(a) is not possible. Moreover, since the jobs in O are mutually compatible, we have

$$f(j_1) \leq s(j_2)$$

[1] This idea comes from the fact that about the only thing we know regarding the algorithm is that it outputs jobs in non-decreasing order of finish time.

[2] We don't allow $t = k$, since there is no job i_{k+1} in A.

[3] Let us get the jobs in O to also have the only property that we know of the jobs in A so far.

[4] Two jobs are different if have unequal starting times and/or unequal ending times.

Chaining these inequalities together, we have that

$$f(i_1) \leq s(j_2), \text{ (Figure 1(b))}$$

Therefore, if we replace j_1 with i_1 in O, then the jobs in O remain mutually compatible (Figure 1(c))!

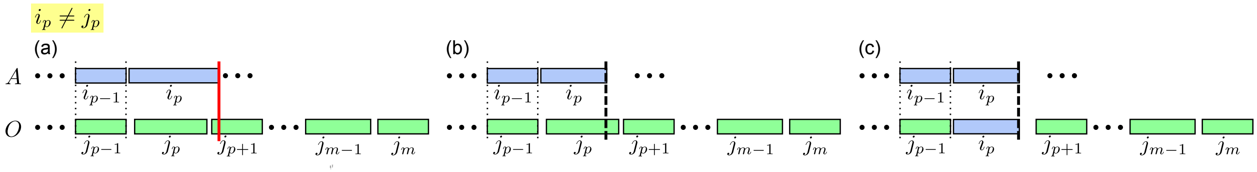

Figure 2: The case when $i_p \neq j_p$, for some $p > 1$. We show only the jobs i_{p-1} and i_p in A. Black dots indicate earlier, intermediate, or later jobs. (a) Can the finish time of i_p be larger than the finish time of j_p (potentially causing i_p and j_{p+1} to conflict)? (b) No! The reason is that both i_p and j_p start after i_{p-1} finishes. Therefore, after the EFT algorithm has selected i_{p-1} and included it in A, both i_p and j_p (which are compatible with i_{p-1}) were available for being chosen as the next job in A, However, the EFT algorithm selected the job i_p. Hence, its finish time must be less than or equal to the finish time of j_p. (c) Therefore, if we replace j_p with i_p in O, all the jobs in O continue to be mutually compatible.

Case 2: $i_p \neq j_p$, for some $1 < p \leq k$. Now suppose that the smallest index at which A and O differ is some $p > 1$; p must also be at most k. Recall that this statement means that for every index $q < p$, $i_q = j_q$ but $i_p \neq j_p$. We can make virtually a similar argument as before but do it in two parts:[5]

$$f(i_{p-1}) = f(j_{p-1}), \text{ since } i_{p-1} \text{ and } j_{p-1} \text{ are the same job}$$

Moreover, since the jobs in O are mutually compatible, we have

$$f(j_{p-1}) \leq s(j_p)$$

Chaining these inequalities together, we have that

$$f(i_{p-1}) \leq s(j_p)$$

Therefore, j_p is compatible with i_{p-1} and would have been in the list of jobs available to the EFT algorithm when it selected i_p. Since the algorithm selects the available job with the smallest finishing time, we can conclude that

$$f(i_p) \leq f(j_p)$$

All jobs with index $> p$ in O are compatible with j_p. Since we have just shown that $f(i_p) \leq f(j_p)$, we can conclude that i_p is also compatible with all jobs with index $> p$ in O. In other words, if we replace j_p with i_p in O, the set of jobs in O continue to be mutually compatible!

We can iterate this "exchange argument" for every index at which A and O have different jobs. It is crucial that we make this argument index by index, starting at the smallest index at which A and O differ. That is the only way we can guarantee the equality $f(i_{p-1}) = f(j_{p-1})$ above. It is important to note that while the proof appears to be iterative, we are not describing an algorithm. All we are doing is mentally processing A and O and removing their differences one job at a time.

[5]The argument for i_1 was simpler because we had no earlier jobs to worry about. Here, we have to start the proof with i_{p-1} in mind.

Case 3: $i_p = j_p$ for all $1 \leq p \leq k$ but $m > k$. Are we done? Well, no! The reason is that this process proves the following: as long as the index of the differing job is less than or equal to k, we can exchange the job in O with the job in A. Therefore, we can ensure that the sequence of jobs (notice the change at index $k+1$) $i_1, i_2, \ldots, i_{k-1}, i_k, j_{k+1}, j_{k+2}, \ldots j_{m-1}, j_m$ is mutually compatible. We have still not precluded the possibility that O contains more jobs than A.

Fortunately, it is easy to deal with this possibility. If O indeed has the structure above, then j_{k+1} is compatible with i_k. Therefore, after the EFT algorithm selected i_k, it would not have processed all the jobs, meaning that the while loop would not have ended. This fact contradicts our assumption that the algorithm output A when it concluded. Therefore, O must also have k jobs.