How does the problem of "Scheduling to Minimize Lateness" exhibit optimal substructure?

The problem of "Scheduling to Minimize Lateness" is as follows (see Section 4.2 of the book "Algorithm Design" by Jon Kleinberg and Eva Tardos):

> ***Input:*** A finite set $J = J_1, J_2, \ldots, J_n$ of n jobs, their processing time $t_1, t_2, \ldots, t_n$ and deadlines $d_1, d_2, \ldots, d_n$. We assume that all jobs were available to start at the common start time 0.
>
> ***Output:*** A schedule of these jobs to minimize their maximum lateness.
> Suppose a job J_i is scheduled at time s_i, then its lateness is defined to be $l_i = s_i + t_i - d_i$ (it is 0 if $s_i + t_i \leq d_i$.)

It is known that this problem can be solved by a greedy algorithm which schedules the jobs in the non-decreasing order of their deadlines.

I am confused about its optimal substructure property (not explicitly described in the book).

Let $L(s, S)$ be the optimal lateness obtainable from scheduling the jobs in S which are available to start at the common start time s.

On the one hand, by enumerating over the possible first jobs to schedule, we get the following recurrence:

$$L(s, S) = \min_{1 \leq i \leq n} \Big(\max \big(L(0, \{J_i\}), L(t_i, J \setminus \{J_i\}) \big) \Big).$$

Intuitively, this problem exhibits optimal substructure.

On the other hand, it seems that the problem does *not* exhibit optimal substructure with respect to the form of subproblems defined above.
To illustrate this, consider three jobs $J = J_1, J_2, J_3$ with processing time $t_1 = 6, t_2 = 2, t_3 = 2$ and deadlines $d_1 = 2, d_2 = 7, d_3 = 8$. An optimal solution is to schedule J_1, J_3, J_2 in order, producing a maximum lateness of 4. However, this optimal solution does *not* contain the optimal solution to the subproblem of scheduling $\{J_2, J_3\}$ given the common start time $t_1 = 6$, which is $L(6, \{J_2, J_3\}) = 2$ by scheduling J_2, J_3 in order.

This contradicts the "definition" of the optimal substructure property given in CLRS (Sections 15.3 and 16.2; 3rd edition):

> A problem exhibits optimal substructure if an optimal solution to the problem contains within it optimal solutions to subproblems.

Problem: How does the problem exhibit optimal substructure?

algorithms optimization dynamic-programming scheduling

edited 11 hours ago

asked 18 hours ago

hengxin
6,895 ● 15 ● 46

question eligible for bounty tomorrow

1 Answer

Let us refine the "definition" of the optimal substructure property given in CLRS into two definitions.

> A problem exhibits **strongly** optimal substructure if **every** optimal solution to the problem contains optimal solutions to subproblems.
>
> A problem exhibits **weakly** optimal substructure if **there is at least one** optimal solution to the problem contains optimal solutions to subproblems.

What you see in the problem of "Interval Scheduling to Minimize Lateness" is the weakly optimal substructure. Take your example of three jobs $J = \{J_1, J_2, J_3\}$ with processing time $t_1 = 6, t_2 = 2, t_3 = 2$ and deadlines $d_1 = 2, d_2 = 7, d_3 = 8$. One optimal solution is to schedule $J1, J2, J3$ in that order, which contains optimal solutions to all subproblems.

Weakly optimal substructure happens often when the goal of an optimization problem is not restricting enough to guarantee every optimal solution must contain optimal solutions to subproblems.

Here is a version of the interval scheduling problem that exhibits strongly optimal substructure. There are two changes in the problem statement compared to the previous one. What is to be minimized is their total lateness instead of their maximum latess. Also the lateness of a job is allowed to be negative. While this version of the problem is rather trivial to solve, it does serve to illustrate the concept of strongly optimal substructure.

> ***Input:*** A finite set $J = \{J_1, J_2, \ldots, J_n\}$ of n jobs, their processing time $t_1, t_2, \ldots, t_n$ and deadlines $d_1, d_2, \ldots, d_n$. We assume that all jobs were available to start at the common start time 0.

Output: A schedule of these jobs to minimize their total lateness. Each job must continue processing once started. At any time, there can be at most one job that is being processed. Suppose a job J_i is scheduled at time s_i, then its lateness is defined to be $l_i = s_i + t_i - d_i$. The total lateness is the sum of all l_i.

edited 10 hours ago

answered 10 hours ago

Apass.Jack
2,122 2 21

Thanks. I agree with you (+1 vote). That is also my guess that there is a difference between "for every" and "there exists". (However, I did not realize that before I tried to solve this problem) Let's wait for more reviews. – hengxin 50 secs ago edit

Answer Your Question