Algorithm: The method is to always arrange jobs providing most profit first. Meanwhile, make sure the arrangement left as much time space as possible.

Step 1: Sort jobs by profit to an array A, that is the first element in array A is the job providing most profit, which means $A[0]$ has greatest g. And the last element in array A is the job providing least profit, which means $A[n-1]$ has the smallest g.

Step 2: Schedule the first job in A one unit of time before its deadline t. That is, suppose job i is in $A[0]$, schedule job i to start at $t_i - 1$.

Step 3: Schedule next job in A one unit of time before its deadline t. That is, suppose job j is in $A[1]$, schedule job j to start at $t_j - 1$. If there is already another job k been scheduled at $t_j - 1$, keep tracing back one unit of time from $t_j - 1$ until a free time interval. Schedule job j in that time interval. For example, if there are jobs scheduled at $t_j - 1$ and $t_j - 2$, but $t_j - 3$ is free, schedule job j at $t_j - 3$. If such a time interval cannot be found, stop the procedure.

Step 4: Repeat step 3 until there is no job haven't been scheduled left in array A or the procedure stop in step 3.

Step 5: Check whether there are jobs haven't been scheduled left in array A. If there are, then our current schedule is optimal. If there isn't, then we successfully scheduled every job before its corresponding deadline. However, there might be free time intervals in our schedule. Because we do not want to do nothing in those time intervals, remove all intervals and start the first job in the schedule at time 0. Now this schedule is optimal.