Department of
Computer Science and Engineering

Design and Analysis of Algorithms

Job Sequencing with Deadlines.

Problem Description

We are given a set of n jobs. Associated with job i is an integer deadline $d_i \geq 0$ and a profit $p_i > 0$. For any job i the profit p_i is earned iff the job is completed by its deadline. To complete a job, one has to process the job on a machine for one unit of time. Only one machine is available for processing jobs. A feasible solution for this problem is a subset J of jobs such that each job in this subset can be completed by its deadline. The value of a feasible solution J is the sum of the profits of the jobs in J, or $\sum_{i \in J} p_i$. An optimal solution is a feasible solution with maximum value. Here again, since the problem involves the identification of a subset, it fits the subset paradigm.

Example 4.2 Let $n = 4$, $(p_1, p_2, p_3, p_4) = (100, 10, 15, 27)$ and $(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$. The feasible solutions and their values are:

Example 4.2 Let $n = 4$, $(p_1, p_2, p_3, p_4) = (100, 10, 15, 27)$ and $(d_1, d_2, d_3, d_4) = (2, 1, 2, 1)$. The feasible solutions and their values are:

	feasible solution	processing sequence	value
1.	(1, 2)	2, 1	110
2.	(1, 3)	1, 3 or 3, 1	115
3.	(1, 4)	4, 1	127
4.	(2, 3)	2, 3	25
5.	(3, 4)	4, 3	42
6.	(1)	1	100
7.	(2)	2	10
8.	(3)	3	15
9.	(4)	4	27

Theorem 4.3 Let J be a set of k jobs and $\sigma = i_1, i_2, \ldots, i_k$ a permutation of jobs in J such that $d_{i_1} \le d_{i_2} \le \cdots \le d_{i_k}$. Then J is a feasible solution iff the jobs in J can be processed in the order σ without violating any deadline.

Theorem 4.3 Let J be a set of k jobs and $\sigma = i_1, i_2, \ldots, i_k$ a permutation of jobs in J such that $d_{i_1} \le d_{i_2} \le \cdots \le d_{i_k}$. Then J is a feasible solution iff the jobs in J can be processed in the order σ without violating any deadline.

Proof: Clearly, if the jobs in J can be processed in the order σ without violating any deadline, then J is a feasible solution. So, we have only to show that if J is feasible, then σ represents a possible order in which the jobs can be processed. If J is feasible, then there exists $\sigma' = r_1, r_2, \ldots, r_k$ such that $d_{r_q} \ge q$, $1 \le q \le k$. Assume $\sigma' \ne \sigma$. Then let a be the least index such that $r_a \ne i_a$. Let $r_b = i_a$. Clearly, $b > a$. In σ' we can interchange r_a and r_b. Since $d_{r_a} \ge d_{r_b}$, the resulting permutation $\sigma'' = s_1, s_2, \ldots, s_k$ represents an order in which the jobs can be processed without violating a deadline. Continuing in this way, σ' can be transformed into σ without violating any deadline. Hence, the theorem is proved. $\square$

High-level description of job sequencing algorithm

```
Algorithm GreedyJob(d, J, n)
// J is a set of jobs that can be completed by their deadlines.
{
    J := {1};
    for i := 2 to n do
    {
        if (all jobs in J ∪ {i} can be completed
            by their deadlines) then J := J ∪ {i};
    }
}
```

High-level description of job sequencing algorithm

Greedy Algorithm for sequencing unit time jobs with dead-lines and profits

```
Algorithm JS(d, j, n)
// d[i] ≥ 1, 1 ≤ i ≤ n are the deadlines, n ≥ 1. The jobs
// are ordered such that p[1] ≥ p[2] ≥ ··· ≥ p[n]. J[i]
// is the ith job in the optimal solution, 1 ≤ i ≤ k.
// Also, at termination d[J[i]] ≤ d[J[i + 1]], 1 ≤ i < k.
{
    d[0] := J[0] := 0; // Initialize.
    J[1] := 1; // Include job 1.
    k := 1;
    for i := 2 to n do
    {
        // Consider jobs in nonincreasing order of p[i]. Find
        // position for i and check feasibility of insertion.
        r := k;
        while ((d[J[r]] > d[i]) and (d[J[r]] ≠ r)) do r := r − 1;
        if ((d[J[r]] ≤ d[i]) and (d[i] > r)) then
        {
            // Insert i into J[ ].
            for q := k to (r + 1)  step −1 do J[q + 1] := J[q];
            J[r + 1] := i; k := k + 1;
        }
    }
    return k;
}
```

Example 4.3 Let $n = 5, (p_1, \ldots, p_5) = (20, 15, 10, 5, 1)$ and $(d_1, \ldots, d_5) = (2, 2, 1, 3, 3)$. Using the above feasibility rule, we have

J	assigned slots	job considered	action	profit
$\emptyset$	none	1	assign to [1, 2]	0
{1}	[1, 2]	2	assign to [0, 1]	20
{1, 2}	[0, 1], [1, 2]	3	cannot fit; reject	35
{1, 2}	[0, 1], [1, 2]	4	assign to [2, 3]	35
{1, 2, 4}	[0, 1], [1, 2], [2, 3]	5	reject	40

The optimal solution is $J = \{1, 2, 4\}$ with a profit of 40. $\square$