Lecture 13 – Interval Scheduling
Reading: KT Section 4.1 and 4.2

Partial content of these slides have been obtained from the official lecture slides that accompany the textbook. A complete set of slides can be found at: http://www.cs.princeton.edu/~wayne/kleinberg-tardos/

Greedy algorithm

Greedy(C)  {C is the set of choice candidate}
    S = Ø    {S the solution set}
    while not solution(S) and C ≠ Ø
        x = greedy-choice(C)
        C = C \ {x}
        if feasible(S ∪ {x})
            S = S ∪ {x}
    return S
Greedy analysis strategies

- Greedy algorithm stays ahead.
  - Show that after each step of the greedy algorithm, its solution is at least as good as any optimal algorithm's.

- Exchange argument.
  - Gradually transform any solution to the one found by the greedy algorithm without hurting its quality.

- Structural.
  - Discover a simple "structural" bound asserting that every possible solution must have a certain value. Then show that your algorithm always achieves this bound.

Scheduling to minimize lateness

The exchange argument approach
Scheduling to minimize lateness

• Setting:
  • A single resource
  • Each job $j$ requires $t_j$ time units, and a deadline of $d_j$
  • $j$ starts at $s_j$, and is done at $f_j = s_j + t_j$
  • The lateness of a job $j$ is $l_j = f_j - d_j$

• Goal:
  • Decide on the start time of each job to minimize the maximum lateness ($L$)
  • $L = \max_j l_j$

Greedy algorithms to minimize lateness

• How would you order the jobs in the schedule?
Earliest deadline first

**Algorithm for Earliest Deadline First**

```
EARLIEST-DEADLINE-FIRST (n, l_1, l_2, ..., l_n, d_1, d_2, ..., d_n)

SORT n jobs so that d_1 ≤ d_2 ≤ ... ≤ d_n.

\[ t \leftarrow 0 \]

FOR \( j = 1 \) TO \( n \)

Assign job \( j \) to interval \([t, t + l_j]\).

\[ s_j \leftarrow t : f_j \leftarrow t + l_j \]

\[ t \leftarrow t + l_j \]

RETURN intervals \([s_1, f_1], [s_2, f_2], ..., [s_n, f_n]\).
```

Correctness — Exchange approach

**What is it?**
- We start with an optimal solution \( O \), and turn it into the greedy solution \( A \)

**For the earliest deadline first algorithm...**
- There exists an optimal schedule with no idle time
- The earliest-deadline-first schedule has no idle time
- The earliest-deadline-first schedule has no inversions
- All schedules with no inversions and no idle time have the same maximum lateness
- If a schedule (with no idle time) has an inversion, it has one with a pair of inverted jobs scheduled consecutively
- Swapping two adjacent, inverted jobs reduces the number of inversions by one and does not increase the max lateness

**There is an optimal schedule that has no inversions and no idle time.**

**The schedule \( A \) produced by the greedy algorithm has optimal maximum lateness \( L \).**