CSE 331: Introduction to Algorithm Analysis and Design

Greedy Algorithms

1 Interval Scheduling: Minimize Maximum Lateness

1.1 Problem

Input: A set of n intervals $I = \{i_0, i_1, \ldots, i_{n-1}\}$ such that each i_j is defined by its deadline $d(i_j)$ and duration $t(i_j)$. The input also includes a global start time s which we will assume is 0. If $s \neq 0$ we can shift s and all the deadlines to make $s = 0$ so we will use this assumption to make things simpler.

Output: A schedule of all n tasks with no conflicts that minimizes the maximum lateness. For the schedule, each task is given a start time $s(i_j)$ and we define a finish time $f(i_j) = s(i_j) + d(i_j)$. The lateness of a task is defined as $l(i_j) = max(f(i_j) - d(i_j), 0)$ and the maximum lateness of a schedule is the lateness of the latest task $L = \max_{0 \leq j \leq n-1} l(i_j)$. The algorithm should output a schedule that minimizes L.

1.2 Algorithm

1. Sort the tasks in I by increasing deadline
2. Initialize the current finish time to the global start time $f = s = 0$
3. Iterate through the tasks in I. Call the current task i_j
   (a) Set the start time of i_j to the current finish time $s(i_j) = f$
   (b) Compute the finish time of i_j to $f(i_j) = s(i_j) + t(i_j)$
   (c) Update the finish time to include the current task $f = f + t(i_j)$
4. return the resulting schedule

1.3 Correctness

Exchange argument: We will prove the correctness of the algorithm by proving the following three properties. We define an inversion as two tasks i_j and $i_{j'}$ in a schedule such that i_j is scheduled before $i_{j'}$ and $d(i_{j'}) < d(i_j)$:

- Any two schedules with 0 idle time and 0 inversions have the same max lateness.
- Greedy schedule has 0 idle time and 0 inversions.
- There is an optimal schedule with 0 idle time and 0 inversions.

If all three of these properties are true, we know there is an optimal solution with 0 idle time and 0 inversions and that the greedy solution has 0 idle time and 0 inversions. Since any two solution that both have 0 idle time and 0 inversions have the same maximum lateness, the greedy solution must be optimal.

Lemma 1. *Any two schedules with 0 idle time and 0 inversions have the same max lateness.*

Proof. Let S_0 and S_1 be two arbitrary schedules which both have 0 idle time and 0 inversions. From this, we know that both schedules performs tasks consecutively and that for any two tasks such that $d(i_j) < d(i_{j'})$, that task i_j must be scheduled before $i_{j'}$ in both S_0 and S_1.

This implies that the schedules must be identical except for tasks such that $d(i_k) = d(i_{k'})$. If the deadlines are identical, the tasks are scheduled consecutively but in arbitrary order. We will show that this order does not affect the maximum lateness of the schedule. For completeness, our argument can be applied to an arbitrary number of tasks all with the same deadline, but for clarity we present the argument with just two tasks.

For two tasks i_k and $i_{k'}$ such that $d(i_k) = d(i_{k'}) = d'$ switching the tasks doesn't change the maximum lateness of the two tasks. No matter which task is scheduled first, the end time of the later task will be $t_f = f' + t(i_k) + t(i_{k'})$ where f' is the finish time of the previously scheduled task. Given this the maximum lateness of these tasks is $max(t_f - d', 0)$ which is the same regardless of the order of i_k and $i_{k'}$. To expand this proof to an arbitrary number of tasks with the same deadline, the argument is the same except $t_f = f' + \sum t(i_j)$ for all j in the set of tasks with the same deadline.

□

Lemma 2. *The greedy schedule has 0 idle time and 0 inversions.*

Proof. This follows directly from the definition of the algorithm. Since the algorithm schedules each task to start at the end of the previously scheduled task, the resulting schedule will have 0 idle time. The algorithm schedules the tasks in order of increasing deadline, so there are no inversions. □

Lemma 3. *There is an optimal schedule with 0 idle time and 0 inversions.*

Proof. Here we will apply the core of the exchange argument. Having 0 idle time and 0 inversions is the fundamental property that a schedule from the greedy algorithm will have. For this proof we will assume there is an optimal solution that does not have 0 idle time and 0 inversions and prove that it's maximum lateness L can't increase by adding this property. We will show this in two part, one for removing idle time and one for removing inversions.

Removing idle time from the optimal schedule can not increase the maximum lateness of the schedule. Removing idle time before a task will decrease the finish time for that task. By the equation for the lateness of a task $l(i_j) = max(f(i_j) - d(i_j), 0)$, we can see that decreasing $f(i_j)$ can't increase $l(i_j)$.

Removing an inversion from the optimal solution can not increase the maximum lateness of the schedule. If a schedule has any inversions, there must exist two consecutive tasks that are inverted that we'll call i_p and i_{p+1}. We will switch the order of these tasks and show that it can't increase the maximum lateness of the schedule. We know that $d(i_{p+1}) < d(i_p)$ and that switching the tasks does not change the finish time of the later scheduled task which we'll call f. Before the switch, the lateness of the tasks are $l(i_p) = max(f - t(i_{p+1}) - d(i_p), 0)$ and $l(i_{p+1}) = max(f - d(i_{p+1}), 0)$ making the maximum lateness between the two task $l = max(f - t(i_{p+1}) - d(i_p), f - d(i_{p+1}), 0)$.

After switching the tasks, the maximum lateness is $l' = max(f - d(i_p), f - t(i_p) - d(i_{p+1}), 0)$ and we need to show that $l' \leq l$. We know that $f - t(i_p) - d(i_{p+1}) < f - d(i_{p+1})$ so we only need to show that $f - d(i_p) \leq l$ which we will do by showing that $f - d(i_p) \leq f - d(i_{p+1})$. We can rewrite this inequality as follows:

$$f - d(i_p) \leq f - d(i_{p+1})$$
$$-d(i_p) \leq -d(i_{p+1})$$
$$d(i_p) > d(i_{p+1})$$

Which is true from the definition of an inversion.

By fixing all such inversions, we will arrive at a schedule with 0 inversions that could only be more optimal. Since the solution was already assumed to be optimal, it must still be optimal now that it has 0 idle time and 0 inversions which completes the proof.

□

1.4 Runtime

1. Sort the tasks in I by increasing deadline: Sorting n elements in $O(n \cdot log(n))$ time.
2. Initialize the current finish time to the global start time $f = s = 0$: $O(1)$
3. Iterate through the tasks in I. Call the current task i_j: Iterate through all n tasks for $O(n)$ total iterations.
   (a) Set the start time of i_j to the current finish time $s(i_j) = f$: $O(1)$
   (b) Compute the finish time of i_j to $f(i_j) = s(i_j) + t(i_j)$: $O(1)$
   (c) Update the finish time to include the current task $f = f + t(i_j)$: $O(1)$
4. return the resulting schedule: $O(1)$

The total runtime is $O(n \cdot log(n)) + O(1) + O(n) \cdot (O(1) + O(1) + O(1)) + O(1)$ which is $O(n \cdot log(n))$.