CSC226, Summer 2023
Lab 2: Greedy Scheduling

Ali Mortazavi

June 21, 2023

1 Greedy Algorithm

In this lab session, we want to learn:

"how to prove a greedy algorithm can(/cannot) always find the optimal solution."

We want to consider the problem of Interval Scheduling.

2 Interval Scheduling

- We have a set of n jobs $j_1, \ldots, j_n$. (Notice that jobs are not ordered.)
- Each job j starts at s_j and finishes at f_j (Obviously $s_j < f_j$)
- Two jobs are *compatible* if they don't overlap
- The goal is to find the **maximum subset of mutually compatible jobs**.

Question 1. *Try to formally define compatibility for jobs using math notations: consider job i with starting time s_i and finish time f_i, and job j with s_j and f_j respectively.*
Job i and j are compatible if and only if ______________________________

3 Trivial Algorithm

One way to solve this problem is to go over all possible subsets of jobs and check if they're all mutually compatible, and among those compatible sets return the largest subset.

The trivial algorithm has a bad time complexity. In fact:

Question 2. *If we have a set A of size n, then the number of subsets of A is:* ____________________

As you see, when we have n jobs, we need to go over 2^n subsets and check whether they're compatible or not. For slightly large n (like $n \geq 100$), it's impossible to implement it in practice.

4 Greedy Algorithm

Now we want to see if there is a greedy algorithm to solve this problem both "optimally" and efficiently.

- Optimally: the algorithm finds the optimal solution which is a subset of jobs with the maximum size, where all jobs are mutually compatible.
- Efficiently: we would like to find an algorithm with a polynomial running time.

The idea of greedy: Pick jobs in each iteration based on some local criteria.

But which local criteria?

4.1 Finding the right greedy approach

Here are some greedy criteria:

1. In each iteration, pick the job j that has the earliest start time s_j
2. In each iteration, pick the job j that has the shortest time interval $f_j - s_j$
3. In each iteration, pick the job j that has the earliest finish time f_j

5 Proving(/Disproving) the optimality of the solution

Here we will show that using the earliest-start-time-first greedy algorithm and shortest-time-interval algorithm would **not** lead to the optimal solution.

Usually, when we want to show an algorithm doesn't work, we use a counter-example to do so.

Question 3. *Find a counter-example where "earliest-start-time-first" greedy does not lead to the optimal solution.*

Hint: You need to find a set of jobs where if we use the "earliest-start-time-first" algorithm, then we get a sub-optimal solution.

Question 4. *Find a counter-example where "shortest-time-interval" greedy does not lead to the optimal solution.*

However, we will show that the "earliest-finish-time-first" Algorithm is optimal and also can be implemented efficiently.

Question 5. *Prove that the earliest-finish-time-first algorithm is optimal.*

Solution: Let's say we have n jobs. Let $i_1, i_2, \ldots, i_k$ denote the set of jobs selected by the greedy algorithm. Without loss of generality, assume that jobs are ordered based on their finish time. I.e. $f_{i_1} < f_{i_2} < \ldots < f_{i_k}$.

Now, consider an arbitrary optimal solution with jobs $j_1, \ldots, j_m$ where $f_{j_1} < f_{j_2} < \ldots < f_{i_k}$.

Observation 1. *For any optimal solution OPT, there exists some $r \in \{0, 1, 2 \ldots, k\}$ such that the first r jobs selected by greedy are the same as OPT. i.e. $i_1 = j_1, i_2 = j_2, \ldots, i_r = j_r$.*

Now, we prove the following lemma.

Lemma 1. *Among the set of optimal solutions, if there is an optimal solution $(j_1, \ldots, j_m)$ where the first r jobs are the same as the first r jobs in the earliest-finish-time-first greedy solution $(i_1, \ldots, i_k)$, where $r < k$, then there exists an optimal solution $(j'_1, \ldots, j'_m)$ with $r + 1$ first jobs equal to the greedy solution.*

Question 6. *Prove Lemma 1.*

Now, starting with any arbitrary optimal solution OPT, by Observation 1, we know the first $r \leq k$ jobs in OPT is the same as GREEDY. Now, if $r = k$, we are done.

If $r < k$, by applying Lemma 1, we can find another optimal solution OPT' with $r + 1$ jobs the same as GREEDY.

If we apply Lemma 1 enough times, we get an optimal solution OPT* where the first k jobs in OPT* are the same as the jobs in the GREEDY.

Lemma 2. *No job is compatible with set of jobs $i_1, \ldots, i_k$*

Question 7. *Prove Lemma 2.*

Question 8. *Finish the proof.*