CSC 373: Algorithm Design and Analysis
Lecture 2

Allan Borodin

January 9, 2013

Some materials are from Kevin Wayne's slides.

Lecture 2: Announcements and Outline

Announcements

- As now changed on the web page, last assignment due April 1 (start of lecture) and the last term test will be on April 3.
- Most of Problem set 1 is now posted.
- Pragmatic definition for avoiding plagarism.

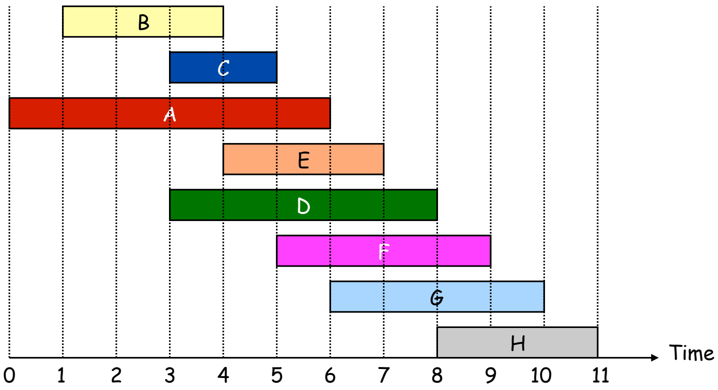
Todays topics

- Greedy algorithms for interval scheduling continued
- Two proof techniques for proving optimality of earliest finish time (EFT) greedy algorithm for interval scheduling
  1. The inductive promising partial solution argument (as in KT-chapter 4)
  2. The charging argument (Wikipedia points to a handout in a previous CSC 373)
- The JISP problem (EFT) greedy algorithm for interval scheduling

Greedy algorithms for interval scheduling

Interval Scheduling Problem

- Job j starts at s_j and finishes at f_j.
- Two jobs are compatible if they don't overlap. (Allow $f_i = s_j$)
- Goal: find maximum subset of mutually compatible jobs.

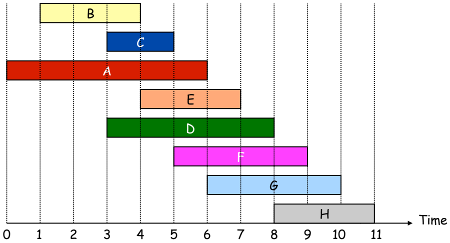


Interval Scheduling: Greedy Algorithm

Greedy template

- Consider jobs in some natural order.
- Take each job provided it's compatible with the ones already taken.

1. Earliest start time: Consider jobs in ascending order of s_j.
2. Earliest finish time: Consider jobs in ascending order of f_j.
3. Shortest interval: Consider jobs in ascending order of $f_j - s_j$.
4. Fewest conflicts: For each job j, count the remaining number of conflicting jobs c_j. Schedule in ascending order of c_j.



Interval Scheduling: Greedy Algorithm

Greedy template

- Consider jobs in some natural order.
- Take each job provided it's compatible with the ones already taken.

counterexample for earliest start time

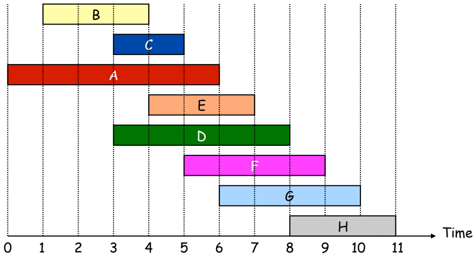
counterexample for shortest interval

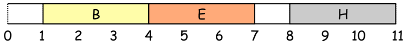
counterexample for fewest conflicts

Optimality of EFT Greedy algorithm

Earliest Finish Time (EFT) Algorithm

- Consider jobs in ascending order of finishing time f_j.
- Take each job provided it's compatible with the ones already taken.





Given the fact that some other reasonable greedy algorithms for the interval scheduling problem do not yield optimal solutions, how can we be convinced that EFT is optimal?

Comments on the optimality of EFT

- The proof outline shows that

The partial solution $S(i)$ at the end of the ith iteration is promising in that it can be extended to an optimal solution (using intervals not yet considered).

- This is not the only possible proof of this result. But before giving another type of proof (a charging argument), you might rightfully ask

"why bother proving this?"

Why prove facts about a particular algorithm?

1. As we have seen, other reasonable (greedy) algorithms for ISP fail to obtain an optimal solution (for all input instances).
   - So while in hindsight we can motivate and convince ourselves that EFT is optimal, we need a convincing argument (i.e. a proof at some level of being convincing) that EFT is indeed optimal.
2. Proofs give us insight into the limitations of an algorithm and also what is and isn't necessary to establish the desired properties.
   - For example, the proof does not rely on the exact manner in which we break "ties" (between intervals with the same finishing time).
   - Hence while an algorithm needs exact specification, any tie breaking rule will work!
3. Proofs also can yield additional facts as we will see in the case of interval colouring and MST problems

The charging argument for EFT optimality

- The previous proof for the optmality of EFT is a proof technique that seems mostly applicable to one pass algorithms, for example as in the greedy (myopic) template.
- The next proof technique (a charging argument) is more widely applicable and easily adapts to approximation bounds for both maximization and minimization problems. Here is the argument for EFT maximization problem.

Charging argument for optimality of EFT

- Let $OPT(\mathcal{I})$ be any solution and in particular an optimal solution on an (arbitrary) input instance $\mathcal{I}$ and let $EFT(\mathcal{I})$ be the output of the EFT algorithm.
- Then we wish to show that we can construct a 1-1 function $h : OPT(\mathcal{I}) \rightarrow EFT(\mathcal{I})$.

The EFT algorithm for the JISP problem

- We consider an NP-hard variant of the interval scheduling problem (ISP) called the job interval scheduling problem (JISP). An instance $\mathcal{I} = \{I_1, \ldots, I_n\}$ has intervals $I_j = (s_j, f_j, \kappa_j)$ where now κ_j is the job class to which interval I_j belongs.
- Jobs are compatible if they do not intersect (as before) and do not belong to the same job class. That is, we can take at most one interval from each job class.

Theorem (EFT applied to JISP)

The EFT algorithm when applied to the JISP problem produces a solution which satisifies $|OPT(\mathcal{I})| \leq 2 \cdot |EFT(\mathcal{I})|$ for every input instance $\mathcal{I}$. That is, EFT is a 2-approximation algorithm for JISP.

- This 2-approximation bound can be proven by extending the EFT optimality proof for the ISP problem to show that there is a 2-1 function $h : OPT(\mathcal{I}) \rightarrow EFT(\mathcal{I})$.