CSE 421: Introduction to Algorithms

Greedy: Interval Scheduling / Partitioning
Shayan Oveis Gharan
Interval Scheduling
Interval Scheduling

- Job j starts at $s(j)$ and finishes at $f(j)$.
- Two jobs compatible if they don’t overlap.
- Goal: find maximum subset of mutually compatible jobs.
Greedy Strategy

Sort the jobs in **some** order. Go over the jobs and take as much as possible provided it is compatible with the jobs already taken.

Main question:

- What order?
- Does it give the Optimum answer?
- Why?
Possible Approaches for Inter Sched

Sort the jobs in some order. Go over the jobs and take as much as possible provided it is compatible with the jobs already taken.

[Earliest start time] Consider jobs in ascending order of start time $s_j$.

[Earliest finish time] Consider jobs in ascending order of finish time $f_j$.

[Shortest interval] Consider jobs in ascending order of interval length $f_j - s_j$.

[Fewest conflicts] For each job, count the number of conflicting jobs $c_j$. Schedule in ascending order of conflicts $c_j$. 
Greedy Alg: Earliest Finish Time

Consider jobs in increasing order of finish time. Take each job provided it’s compatible with the ones already taken.

Sort jobs by finish times so that $f(1) \leq f(2) \leq \ldots \leq f(n)$.  
$A \leftarrow \emptyset$

for $j = 1$ to $n$ {
  if (job $j$ compatible with $A$)
    $A \leftarrow A \cup \{j\}$
}

return $A$

Implementation. O(n log n).
• Remember job $j^*$ that was added last to $A$.
• Job $j$ is compatible with $A$ if $s(j) \geq f(j^*)$. 
Greedy Alg: Example
**Correctness**

**Theorem:** Greedy algorithm is optimal.

**Pf:** (technique: “Greedy stays ahead”)

Let $i_1, i_2, \ldots, i_k$ be jobs picked by greedy, $j_1, j_2, \ldots, j_m$ those in some optimal solution in order.

We show $f(i_r) \leq f(j_r)$ for all $r$, by induction on $r$.

**Base Case:** $i_1$ chosen to have min finish time, so $f(i_1) \leq f(j_1)$.

**IH:** $f(i_r) \leq f(j_r)$ for some $r$

**IS:** Since $f(i_r) \leq f(j_r) \leq s(j_{r+1})$, $j_{r+1}$ is among the candidates considered by greedy when it picked $i_{r+1}$, & it picks min finish, so $f(i_{r+1}) \leq f(j_{r+1})$

Observe that we must have $k \geq m$, else $j_{k+1}$ is among (nonempty) set of candidates for $i_{k+1}$
Interval Partitioning Technique: Structural
Interval Partitioning

Lecture $j$ starts at $s(j)$ and finishes at $f(j)$.

**Goal**: find minimum number of classrooms to schedule all lectures so that no two occur at the same time in the same room.
Interval Partitioning

Note: graph coloring is very hard in general, but graphs corresponding to interval intersections are simpler.
A Better Schedule

This one uses only 3 classrooms
A Structural Lower-Bound on OPT

Def. The depth of a set of open intervals is the maximum number that contain any given time.
A Structural Lower-Bound on OPT

**Def.** The depth of a set of open intervals is the maximum number that contain any given time.

**Key observation.** Number of classrooms needed \( \geq \) depth.

**Ex:** Depth of schedule below = 3 \( \Rightarrow \) schedule below is optimal.

**Q.** Does there always exist a schedule equal to depth of intervals?
Greedy algorithm: Consider lectures in increasing order of start time: assign lecture to any compatible classroom.

Sort intervals by starting time so that \( s_1 \leq s_2 \leq \ldots \leq s_n \).

\[ d \leftarrow 0 \]

\[
\text{for } j = 1 \text{ to } n \{ \\
    \text{if (lect } j \text{ is compatible with some classroom } k, 1 \leq k \leq d) } \\
    \text{schedule lecture } j \text{ in classroom } k \\
    \text{else} \\
    \text{allocate a new classroom } d + 1 \\
    \text{schedule lecture } j \text{ in classroom } d + 1 \\
    d \leftarrow d + 1 \\
\}
\]

Implementation: Exercise!
Correctness

**Observation**: Greedy algorithm never schedules two incompatible lectures in the same classroom.

**Theorem**: Greedy algorithm is optimal.

**Pf (exploit structural property)**.

Let $d =$ number of classrooms that the greedy algorithm allocates.

Classroom $d$ is opened because we needed to schedule a job, say $j$, that is incompatible with all $d-1$ previously used classrooms.

Since we sorted by start time, all these incompatibilities are caused by lectures that start no later than $s(j)$.

Thus, we have $d$ lectures overlapping at time $s(j) + \epsilon$, i.e. $\text{depth} \geq d$

"OPT Observation" $\Rightarrow$ all schedules use $\geq \text{depth}$ classrooms, so $d = \text{depth}$ and greedy is optimal $\cdot$
Minimum Spanning Tree Problem
Minimum Spanning Tree (MST)

Given a connected graph $G = (V, E)$ with real-valued edge weights $c_e$, an MST is a subset of the edges $T \subseteq E$ such that $T$ is a spanning tree whose sum of edge weights is minimized.

$G = (V, E)$

$T$, $\sum_{e \in T} c_e = 50$
Applications

Network design:
• telephone, electrical, hydraulic, TV cable, computer, road

Approximation algorithms for NP-hard problems:
• traveling salesperson problem, Steiner tree

Indirect applications:
• Graph clustering
• max bottleneck paths
• LDPC codes for error correction
• image registration with Renyi entropy
• learning salient features for real-time face verification
• reducing data storage in sequencing amino acids in a protein
• model locality of particle interactions in turbulent fluid flows
• autoconfig protocol for Ethernet bridging to avoid cycles in a network
Properties of the OPT

Simplifying assumption: All edge costs $c_e$ are distinct.

Cut property: Let $S$ be any subset of nodes (called a cut), and let $e$ be the \textit{min} cost edge with exactly one endpoint in $S$. Then every MST contains $e$.

Cycle property. Let $C$ be any cycle, and let $f$ be the \textit{max} cost edge belonging to $C$. Then no MST contains $f$. 

\[
\begin{array}{c}
\text{red edge is in the MST} \\
\text{Green edge is not in the MST}
\end{array}
\]
Cycles and Cuts

Claim. A cycle crosses a cut (from S to V-S) an even number of times.

Pf. (by picture)
**Cut Property: Proof**

**Simplifying assumption:** All edge costs $c_e$ are distinct.

**Cut property.** Let $S$ be any subset of nodes, and let $e$ be the min cost edge with exactly one endpoint in $S$. Then the $T^*$ contains $e$.

**Pf.** By contradiction

Suppose $e = \{u,v\}$ does not belong to $T^*$.

Adding $e$ to $T^*$ creates a cycle $C$ in $T^*$.

There is a path from $u$ to $v$ in $T^* \Rightarrow$ there exists another edge, say $f$, that leaves $S$.

$$T = T^* \cup \{e\} - \{f\}$$

is also a spanning tree.

Since $c_e < c_f$, $\text{cost}(T) < \text{cost}(T^*)$.

This is a contradiction.
Cycle Property: Proof

**Simplifying assumption:** All edge costs $c_e$ are distinct.

**Cycle property:** Let $C$ be any cycle in $G$, and let $f$ be the \text{max} cost edge belonging to $C$. Then the MST $T^*$ does not contain $f$.

**Pf.** (By contradiction)

Suppose $f$ belongs to $T^*$.

Deleting $f$ from $T^*$ cuts $T^*$ into two connected components. There exists another edge, say $e$, that is in the cycle and connects the components.

\[ T = T^* \cup \{e\} - \{f\} \] is also a spanning tree.

Since $c_e < c_f$, $\text{cost}(T) < \text{cost}(T^*)$.

This is a contradiction.