Unweighted Interval Scheduling

Specification

Input: intervals $I_i = [s_i, f_i)$ and values $v_i \in \mathbb{R}$~~and values $v_i \in \mathbb{R}$~~ for $i \in [n]$.

Output: $S \subseteq [n]$ such that no intervals I_i and I_j for distinct $i, j \in S$ overlap and $\sum_{i \in S} v_i |S|$ is maximized.

Greedy algorithm

- Local criterion
- Order

Earliest Finish Time First

Algorithm (assuming $f_i \leq f_{i+1}$ for $i \in [n-1]$)

$G \leftarrow \emptyset$
$f \leftarrow -\infty$
for $i = 1$ to n **do**
  if $s_i \geq f$ **then**
    $G \leftarrow G \cup \{i\}$
    $f \leftarrow f_i$
return G

Complexity analysis

- $O(n)$ time and $O(1)$ space for finding maximum value and producing schedule on-line.
- $O(n \log n)$ time due to sorting.

Correctness – Greed Stays Ahead

Strategy

Design a quality measure for partial solutions such that:

- For every valid solution S and every point in time t, the quality measure of the greedy solution G up to t is at least as good as S up to t.
- For a full solution, optimal quality measure implies optimal objective value.

Quality measures for earliest finish time first

Assume intervals numbered in greedy order.

- input components: cardinality $|S \cap [t]|$
- output components: finish time of the t-th interval in S

Cardinality as Quality Measure

Claim

$(\forall t \in \mathbb{N})\, |G \cap [t]| \geq |S \cap [t]|$

Proof: Induction on t

- Base case: $t = 0$
- Inductive step $t \to t+1$ for $t+1 \notin S$
- Inductive step $t \to t+1$ for $t+1 \in S$
  Let k be meeting in S right before $t+1$ ($k = 0$ if there is none).

$$\begin{aligned} |G \cap [t+1]| &\geq 1 + |G \cap [k]| && \text{[greedy criterion]} \\ &\geq 1 + |S \cap [k]| && \text{[induction hypothesis]} \\ &= |S \cap [t+1]| && \text{[definition of } k\text{]} \end{aligned}$$

Finish Time as Quality Measure

Definition

$$f_S(t) = \begin{cases} \text{finish time of } t\text{-th interval in } S & \text{if it exists} \\ \infty & \text{if } t > |S| \\ -\infty & \text{for } t = 0 \end{cases}$$

Claim

$(\forall t \in \mathbb{N})\, f_G(t) \leq f_S(t)$

Proof: Induction on t

- Base case: $t = 0$
- Inductive step $t \to t+1$

Corollary

$|G| \geq |S|$

From DP to Greed

- Consider meetings ordered earliest start time first.
- $\mathrm{OPT}(k) \doteq \mathrm{OPT}(\{k, k+1, \ldots, n\})$ for $1 \leq k \leq n+1$
- $\mathrm{OPT}(k) = \max(1 + \mathrm{OPT}(\mathrm{next}(k)), \mathrm{OPT}(k+1))$
  where $\mathrm{next}(k) \doteq \min\{\ell : k < \ell \leq n+1 \text{ and } s_\ell \geq f_k\}$
- $$\begin{aligned} \mathrm{OPT}(k) = \max(&1 + \mathrm{OPT}(\mathrm{next}(k)), \\ &1 + \mathrm{OPT}(\mathrm{next}(k+1)), \\ &1 + \mathrm{OPT}(\mathrm{next}(k+2)), \\ &\ldots) \\ = 1 + &\mathrm{OPT}(\mathrm{next}(k^*)) \\ &\text{where } k^* = \arg\min_{k \leq i \leq n} \mathrm{next}(i) \end{aligned}$$
- k^* is interval with earliest finish time among $\{k, \ldots, n\}$.
- Include k^*.
- Continue process with $k \leftarrow \mathrm{next}(k^*)$.

Knapsack Problem with Unit Values

Specification

Input: items $i \in [n]$ specified by weight $w_i \in \mathbb{R}$ ~~and value~~ ~~$v_i \in \mathbb{R}$~~; weight limit $W \in \mathbb{Z}^+$

Ouput: $S \subseteq [n]$ such that $\sum_{i \in S} w_i \leq W$ and $\cancel{\sum_{i \in S} v_i}$ $|S|$ is maximized.

Greedy algorithm

- Local criterion
- Order: lightest first

Shortest Paths – nonnegative weights

Specification

Input: (di)graph $G = (V, E)$; lengths $\ell : E \to [0, \infty)$
$s, t \in V$

Ouput: path P from s to t with minimum length
$\ell(P) \doteq \sum_{e \in P} \ell(e)$

Variants

- Single pair
- Single source

Distance $d(s, t)$

$= \min\{\ell(P) \mid P \text{ path from } s \text{ to } t\}$
$= \infty$ if there is no path from s to t

Greedy Algorithm

Approach

Grow set S of $v \in V$ for which we know $d(s, v)$.

Initialization

$S = \{s\}$ as $d(s, s) = 0$.

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s, u) + \ell(u, v))$$

Proof

- P has to include an edge $(u, v) \in E \cap S \times \overline{S}$.
- $\ell(P) = \ell(P|_{s \rightsquigarrow u}) + \ell(u, v) + \ell(P|_{v \rightsquigarrow}) \geq d(s, u) + \ell(u, v) + 0$

Greed Stays Ahead

Claim

Every path P from s to some vertex in $\overline{S}$ satisfies

$$\ell(P) \geq \min_{(u,v) \in E \cap S \times \overline{S}} (d(s, u) + \ell(u, v))$$

Extending S

- Let $(u^*, v^*) = \arg\min_{(u,v) \in E \cap S \times \overline{S}} (d(s, u) + \ell(u, v))$
- Shortest path $s \rightsquigarrow u^*$ followed by (u^*, v^*) is shortest path $s \rightsquigarrow v^*$.
- $d(s, v^*) = d(s, u^*) + \ell(u^*, v^*)$
- $S \leftarrow S \cup \{v^*\}$

Implementation

Priority queue

Key for $v \in \overline{S}$: $\lambda(v) \doteq \min_{u \in S : (u,v) \in E} (d(s, u) + \ell(u, v))$

Running time with binary heap

- Initialization: $O(n)$
- n min extractions: $O(n \log n)$
- $m = \sum_{v \in V} \text{outdeg}(v)$ key updates: $O(m \log n)$
- Total: $O((n + m) \log n)$

Better algorithms

- Improved data structures (Fibonacci heaps): $O(m + n \log n)$
- Other approaches:
  $O(n + m)$ undirected, $O(m + n \log \log n)$ directed

From DP to Greed

- $\text{OPT}(k, v)$ = length shortest path $s \rightsquigarrow v$ using $\leq k$ edges
- $\text{OPT}(v) = d(s, v) = \lim_{k \to \infty} \text{OPT}(k, v)$
- $\text{OPT}(s) = 0$
- $\text{OPT}(v) = \min_{(u,v) \in E} (\text{OPT}(u) + \ell(u, v))$ for $v \neq s$ (*)
- Let S be set of $u \in V$ for which we know $\text{OPT}(u)$.
- Rewrite $\text{OPT}(v)$ for $v \notin S$ using (*).
- After n levels resulting expression for $\text{OPT}(v)$ is minimum of:
  - (a) $\text{OPT}(u) + \ell(u, v)$ for all $u \in S$ with $(u, v) \in E$
  - (b) $\text{OPT}(u) + \ell(u, v') + \ell(P)$ for some $(u, v') \in E$ with $v \neq v' \notin S$ and $P : v' \rightsquigarrow v$
  - (c) $\text{OPT}(u) + \ell(P)$ for some $u \notin S$ and P containing n edges.
- Minimum of terms over all $v \in \overline{S}$ achieved by term of type (a).
- Finding $(u^*, v^*) = \arg\min_{(u,v) \in E \cap S \times \overline{S}} (\text{OPT}(u) + \ell(u, v))$ tells us how to extend S.
- Time complexity: $O((n + m)n) \to O((n + m) \log n)$.

DP / Greed for DAGs

- Evaluate $\text{OPT}(v) = \min_{(u,v)\in E}(\text{OPT}(u) + \ell(u,v))$ for $v \neq s$ in topological order.
- Running time: $O(n+m)$
- Replacing min by max yields solution to longest path problem.

Greed Stays Ahead vs Exchange Argument

Greed stays ahead

Design a quality measure for partial solutions such that:

- For every valid solution S and every point in time t, the quality measure of the greedy solution G up to t is at least as good as S up to t.
- For a full solution, optimal quality measure implies optimal objective value.

Exchange argument

Consider an optimal solution S. Establish a sequence of local transformations (exchanges) such that:

- The sequence ends in the greedy solution G.
- Each transformation maintains validity and does not deteriorate the objective value.

Unweighted Interval Scheduling

Specification

Input: intervals $I_i = [s_i, f_i)$ for $i \in [n]$.

Output: $S \subseteq [n]$ such that no intervals I_i and I_j for distinct $i, j \in S$ overlap and $|S|$ is maximized.

Greedy algorithm

- Order: earliest finish time first
- Local criterion

Exchange Argument

- Consider an optimal solution S that differs from G.
- There exists a first interval i in the greedy order on which S differs from G.
- It has to be the case that $i \in G$ and $i \notin S$.
- There exists an interval $j > i$ such that $j \in S$.
- $S' \doteq S \setminus \{j\} \cup \{i\}$ is an optimal solution.
- The first interval i' on which S' differs from G (if any) satisfies $i' > i$.
- As there are only a finite number of intervals, the process has to end in G.

Minimizing Maximum Tardiness

Problem

Input: tasks $i \in [n]$ specified by duration $t_i \in \mathbb{R}$ and deadline $d_i \in \mathbb{R}$.

Output: for each $i \in [n]$: $s_i \in [0, \infty)$ and $f_i \doteq s_i + t_i$ s.t.

- no intervals $[s_i, f_i)$ for distinct $i \in [n]$ overlap
- $\max_{i\in[n]}(\text{tardiness}(i))$ is minimized, where $\text{tardiness}(i) \doteq \max(0, f_i - d_i)$.

Example

$n = 2$, $(t_1, d_1) = (10, 14)$, $(t_2, d_2) = (5, 11)$

Approach

- Schedule all tasks back to back starting from 0.
- Remains to find optimal ordering S of $[n]$.

Natural Task Orders

- Smallest duration first
- Smallest slack first
- Earliest deadline first
  - Correctness: exchange argument
  - Running time: $O(n \log n)$

Exchange Argument

- Consider an optimal ordering S and suppose $S \neq G$.
- There have to be two tasks $i, j \in [n]$ such that S schedules i right before j but $d_i \geq d_j$.
- Consider S' obtained by swapping i and j in S.
  - $\text{tardiness}_{S'}(k) = \text{tardiness}_S(k)$ for all $k \in [n] \setminus \{i, j\}$
  - $\text{tardiness}_{S'}(j) \leq \text{tardiness}_S(j)$
  - $\text{tardiness}_{S'}(i) \leq \text{tardiness}_S(j)$

  $\therefore$ Max tardiness does not increase from S to S'.
- Number of inversions of S' with respect to G is one less than of S with respect to G.
  $\therefore$ We end up in G eventually.

Optimal Binary Codes

Want to send messages over alphabet $S = \{\text{A,B,C,D,E}\}$ through binary channel.

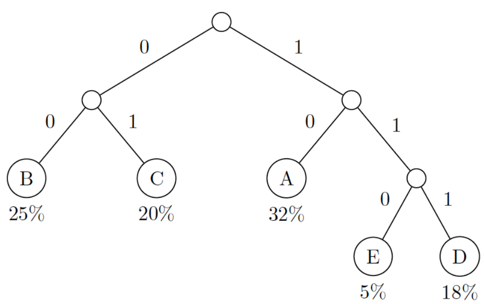
Fixed-length encoding

- 3 bits per symbol

Variable-length encoding

- Frequencies:
  A: 32%, B: 25%, C: 20%, D: 18%, E: 5%
- Goal is to minimize average encoding length.
- No encoding can be a prefix of another encoding.
  $\equiv$ Encodings are paths in binary tree with symbols as leaves.

Optimal Binary Tree



- Encoding: A: 10, B: 00, C: 01, D: 111, E: 110
- Average encoding length:
  $32\% \cdot 2 + 25\% \cdot 2 + 20\% \cdot 2 + 18\% \cdot 3 + 5\% \cdot 3 = 2 + .23 = 2.23$

Key Observations

Observation 1

Optimal tree needs to be full (each node either has two children or none).

Observation 2 - Exchange

- Frequencies of leaves at higher levels need to be $\geq$ frequencies of leaves at lower levels.
- Leaves at a given level are interchangeable.

Corollary

There is an optimal tree in which the two symbols with the lowest frequencies form a cherry (leaves with the same parent).

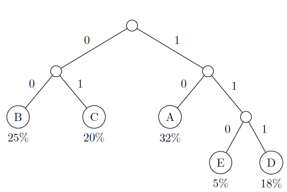
Principle of Optimality

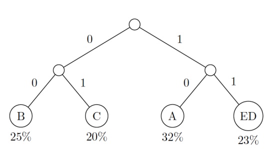
Problem instance I

Input: frequencies $f_s \geq 0$ for $s \in S$ with $\sum_{s \in S} f_s = 1$

Output: binary tree T with leave set S such that $\text{cost}(T) \doteq \sum_{s \in S} f_s \cdot \text{depth}_T(s)$ is minimized.

T for I with cherry $\{\text{E,D}\}$ $\rightleftharpoons$ T' for I' over $\{\text{A,B,C,ED}\}$





$$\text{cost}_I(T) = \text{cost}_{I'}(T') + f_E + f_D$$

T optimal for I w/ cherry $\{\text{E,D}\}$ $\Leftrightarrow$ T' optimal for I'

Algorithm

Recursive case: $|S| \geq 2$

- Find lowest frequency symbols E and D.
- Create instance I' over $S' \doteq S \setminus \{\text{E,D}\} \cup \{\text{ED}\}$ with $f'_{\text{ED}} = f_{\text{E}} + f_{\text{D}}$ and $f'_s = f_s$ for $s \in S \setminus \{\text{ED}\}$.
- Find optimal solution T' for I'.
- Expand leaf ED in T' to cherry with leaves E and D to obtain T.

Implementation

- Binary heap
- Iterative version

Running time: $O(n \log n)$

Outline

Common framework

- System consisting of n components.
- Each component can be in any of a finite number of states.
- Want to set the states of the components so as to optimize a certain objective under certain constraints.

Greedy paradigm

- Consider components in some order.
- Locally optimize setting of component based on prior components and settings only.

Correctness argument

- Greed stays ahead: interval scheduling, shortest paths
- Exchanges: interval scheduling, minimizing maximum tardiness, optimal binary codes, minimum spanning tree

Exchange Argument

Structure

Consider an optimal solution S. Establish a sequence of local transformations (exchanges) such that:

- Each transformation maintains validity and does not deteriorate the objective value.
- The sequence ends in the greedy solution G.

Remarks

- Minimizing maximum tardiness:
  Case $n = 2$ plays central role.
- Optimal binary codes:
  From DP to greed.

Minimum Spanning Tree

Problem

Input: connected graph $G = (V, E)$ and $w : E \to \mathbb{R}$

Output: tree $T = (V, F)$ with $F \subseteq E$ such that $w(T) \doteq \sum_{e \in F} w(e)$ is minimized

Greedy algorithm

Try to make progress using edges of smallest weight w first.

- Locally: Tree growing (Prim)
- Globally: Tree joining (Kruskal)

Tree Growing

$S \leftarrow \{s\}$; $F \leftarrow \emptyset$
while $S \neq V$ **do**
  $(u^*, v^*) \leftarrow \arg\min_{(u,v) \in E \cap S \times \overline{S}} (w(u, v))$
  $S \leftarrow S \cup \{v^*\}$; $F \leftarrow F \cup \{(u^*, v^*)\}$
return $T \doteq (V, F)$

Invariant

$T \doteq (S, F)$ is MST for subgraph of $G = (V, E)$ induced by S.

Implementation

Priority queue for $v \in \overline{S}$ with key $\lambda(v) \doteq \min_{u \in S : (u,v) \in E} (w(u, v))$

Running time with binary heap

- n min extractions: $O(n \log n)$
- $m = \frac{1}{2} \sum_{v \in V} \deg(v)$ key updates: $O(m \log n)$
- Total: $O((n + m) \log n) = O(m \log n)$ as $m \geq n - 1$

Tree Joining

$F \leftarrow \emptyset$
while (V, F) disconnected **do**
  $(u^*, v^*) \leftarrow \arg\min_{(u,v) \in E : u \not\leadsto v \text{ in } (V,F)} (w(u, v))$
  $F \leftarrow F \cup \{(u^*, v^*)\}$
return $T \doteq (V, F)$

Invariant

Connected components of (V, F) are MSTs for subgraphs of $G = (V, E)$ that they induce.

Implementation

- Consider edges $(u, v) \in E$ in order of nondecreasing weight.
- Add (u, v) to T if $u \not\leadsto v$ in (V, F).

Maintaining Connected Components

Tables

- Table with (v,cc(v)) pairs: $O(1)$ time to find cc(v)
- Table with (C,v) pairs: $O(|C|)$ time to relabel C

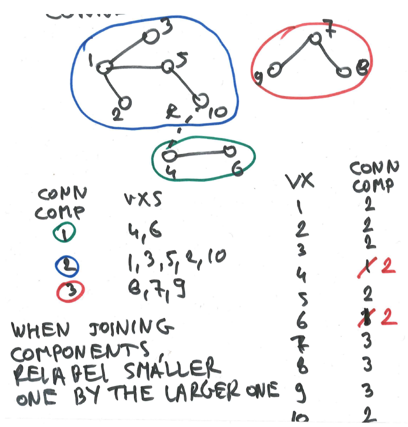
Lazy relabeling

- When merging C_1 and C_2, relabel smaller one.
- Each time v gets relabeled, $|\text{cc}(v)|$ at least doubles.
- Number of times v gets relabeled is at most $\log n$.

Running time

- Sorting the edges: $O(m \log m)$
- Testing edges: $O(m)$
- Maintaining connected components: $O(n \log n)$
- Total: $O(m \log(m) + n \log(n)) = O(m \log n)$ as $m \geq n - 1$

Lazy Relabeling – example



Maintaining Connected Components

Rooted trees

- Store each connected component as a rooted tree, with edges pointing in direction of root.
- $O(\text{depth}(C))$ time to find root of $C = \text{cc}(v)$ given v.

Lazy relabeling

- When merging C_1 and C_2, make root of smaller one point to root of larger one.
- Keeps depth upper bounded by $\log n$.

Running time

- Sorting the edges: $O(m \log m)$
- Testing edges: $O(m \log n)$
- Maintaining connected components: $O(n)$
- Total: $O(m \log(m) + m \log(n) + n) = O(m \log n)$ as $m \geq n - 1$

Better Algorithms

Based on tree growing

- Binary heap: $O(m \log n)$
- Improved data structures (Fibonacci heaps): $O(m + n \log n)$

Based on tree joining

- $O(m \log m)$ due to sorting edges
- Lazy relabeling: $O(m \log n)$ given sorted edges
- Improved data structures (Union-Find): $O(m \cdot \alpha(n, m))$ given sorted edges, where α is inverse Ackermann

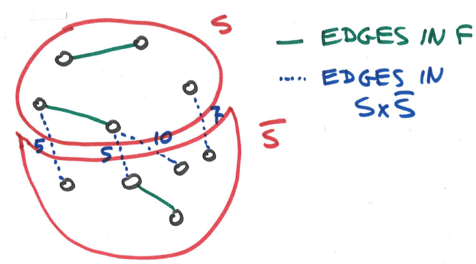
Other approaches

$O(m \cdot \alpha(n, m))$ where α is inverse Ackermann

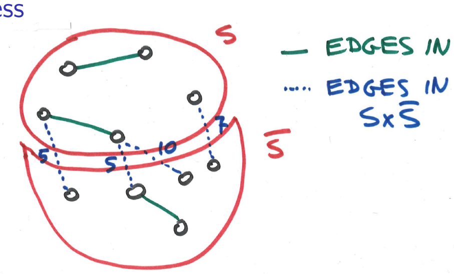
Correctness

Setting

- Suppose we know a subset $F \subseteq E$ such that there exists an MST T of G that contains F.
- Consider a subset $S \subseteq V$ such that no edge in F crosses the cut $(S, \overline{S})$, i.e., $F \cap S \times \overline{S} = \emptyset$.



Correctness



Observations

- T has to contain an edge in $E \cap S \times \overline{S}$.
- This edge contributes at least $\min_{e \in E \cap S \times \overline{S}}(w(e))$ to $w(T)$.

Cut property

- Let $e^* = \arg\min_{e \in E \cap S \times \overline{S}}(w(e))$.
- There exists an MST T' of G that contains $F \cup \{e^*\}$.

Instantiations

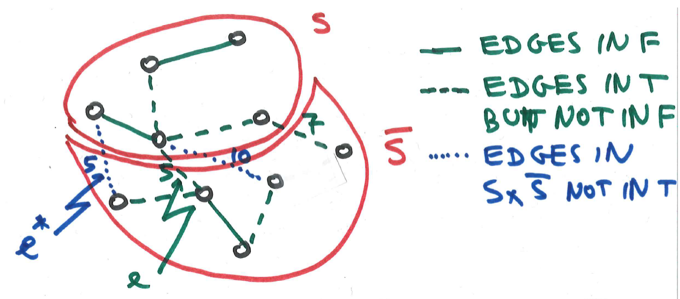
Cut property

- Let $F \subseteq E$ such that there exists an MST T of G that contains F.
- Let $S \subseteq V$ such that no edge in F crosses the cut $(S, \overline{S})$, i.e., $F \cap S \times \overline{S} = \emptyset$.
- Let $e^* = \arg\min_{e \in E \cap S \times \overline{S}}(w(e))$.
- There exists an MST T' of G that contains $F \cup \{e^*\}$.

Correctness implications

- Apply cut property with F the set of edges included thus far.
- Tree growing: S is set of vertices in current tree.
- Tree joining: S is set of vertices connected to u^* in current forest, where (u^*, v^*) is edge under consideration.

Exchange Argument for Cut Property



Exchange Argument for Cut Property

Cut property

Let $e^* = \arg\min_{e \in E \cap S \times \overline{S}}(w(e))$.
There exists an MST T' of G that contains $F \cup \{e^*\}$.

Proof

- Suppose e^* not in T; otherwise done.
- Consider adding e^* to T. This induces cycle that crosses $(S, \overline{S})$ somewhere else, say at $e \in E \cap S \times \overline{S}$.
- Replacing e by e^* in T yields spanning tree T' of G.
- Since $w(e^*) \leq w(e)$,

$$w(T') = w(T) + w(e^*) - w(e) \leq w(T).$$

- $\therefore$ T' is an MST of G containing $F \cup \{e^*\}$