Greedy Algorithms

Interval Scheduling: Section 4.1

Greedy Algorithms

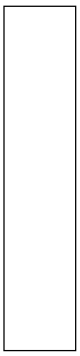
- **Optimization problems involve profit maximization (or cost minimization) under resource constraints**
- **Greedy Algorithms build solutions step-by-step: each step uses "locally optimal" (short-sighted?) rules to optimize the objective function**

Greed

- ***"Greed ... is good. Greed is right. Greed works."***

  -- Gordon Gekko (Michael Douglas) in the movie *Wall Street*

Knapsack Problem



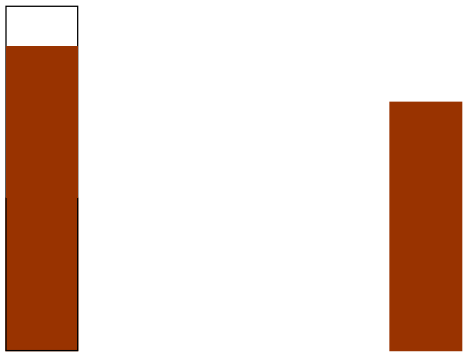
Bag of fixed size

Chocolates of different sizes

How to maximize the amount of chocolate packed?

Knapsack Problem: greedy

Knapsack Problem: optimal



Interval scheduling

- **Set of jobs {*1, 2, …, n*}**
- **Job *i* starts at time s_i and ends at time f_i**
- **Two jobs *compatible* if they do not overlap**
- **Goal: find the maximum subset of mutually compatible jobs**

Interval scheduling

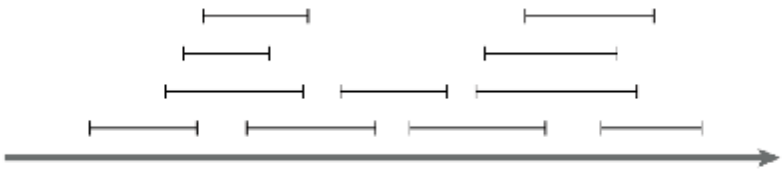


Figure 1.4 An instance of the Interval Scheduling Problem.

Interval scheduling: greed algorithms

- **Greedy approach**
  - **Consider jobs in a specific order**
  - **Choose the "current best" job**
  - **Delete all other jobs that are not compatible with this job**

Interval scheduling: greedy algorithms

- **[Earliest start time] Consider jobs in the ascending order of their start times s_i**
- **[Earliest finish time] Consider jobs in the ascending order of their finish times f_i**
- **[Shortest interval] Consider jobs in the ascending order of their lengths $f_i - s_i$**
- **[Fewest conflicts] For each job i, count the number of conflicting jobs c_i. Consider jobs in the ascending order of c_i**

Interval scheduling: greedy algorithms

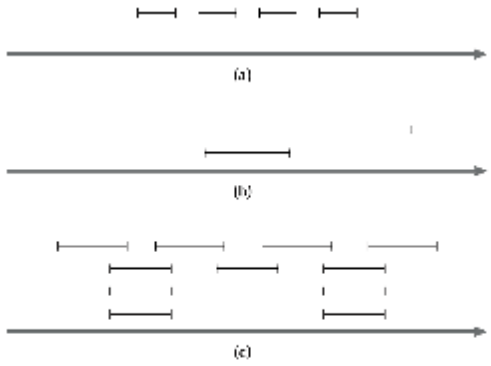


Figure 4.1 Some instances of the Interval Scheduling Problem on which natural greedy algorithms fail to find the optimal solution. In (a), it does not work to select the interval that starts earliest; in (b), it does not work to select the shortest interval; and in (c), it does not work to select the interval with the fewest conflicts.

Earliest finish time

```
Initially let R be the set of all requests, and let A be empty
While R is not yet empty
  Choose a request i ∈ R that has the smallest finishing time
  Add request i to A
  Delete all requests from R that are not compatible with request i
EndWhile
Return the set A as the set of accepted requests
```

Earliest finish time

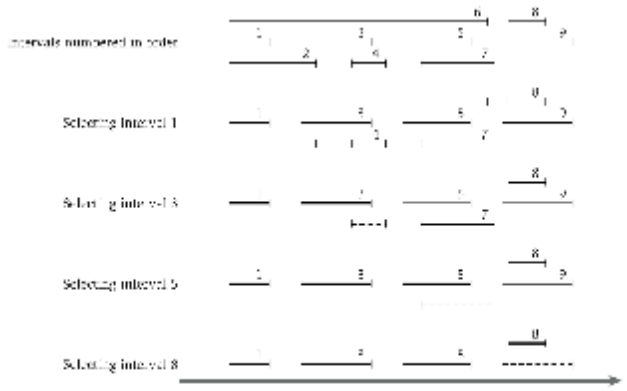


Figure 4.2 Sample run of the Interval Scheduling Algorithm. At each step the selected intervals are darker lines, and the intervals deleted at the corresponding step are indicated with dashed lines.

Analysis

- **Running time = O(n log n)**
- **Let A be the solution of the greedy algorithm**
  - **A is a compatible set of requests**

Analysis

- **Running time = O(n log n)**
- **Let A be the solution of the greedy algorithm (earliest finish time algo.)**
  - **A is a compatible set of requests**
  - **A is optimal**

Greedy stays ahead

- **Let A = i_1, i_2 , …, i_k**
- **Let OPT be the optimal solution**
  - **OPT = $j_1, j_2, \ldots, j_m$**
- **Both A and OPT are arranged in the natural interval order**
- **Need to show that $k \geq m$**

Greedy stays ahead

- **[4.2] For all indices $r \leq k$ we have $f(i_r) \leq f(j_r)$**

Greedy stays ahead

- **[4.2] For all indices $r \leq k$ we have $f(i_r) \leq f(j_r)$**
- **Proof by induction**
  - **Base case: r = 1. Greedy picks the earliest finish time**
  - **Assume $f(i_{r-1}) \leq f(j_{r-1})$**
  - **Can the greedy algorithm's r^{th} job really finish later than OPT's?**

Greedy stays ahead

- **[4.2] For all indices r ≤ k we have f(i_r) ≤ f(j_r)**
- **Proof by induction**
  - **Base case: r = 1. Greedy picks the earliest finish time**
  - **Assume f(i_{r-1}) ≤ f(j_{r-1})**

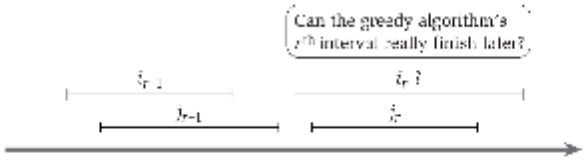


Figure 4.3 The inductive step in the proof that the greedy algorithm stays ahead.

Greedy stays ahead

- **[4.2] For all indices r ≤ k we have f(i_r) ≤ f(j_r)**
- **Proof by induction**
  - **Base case: r = 1. Greedy picks the earliest finish time**
  - **Hypothesis: Let r > 1; assume f(i_{r-1}) ≤ f(j_{r-1})**
  - **Induction step**
    - **f(i_{r-1}) ≤ f(j_{r-1}) ≤ s(j_r)**
    - **Job j_r is available after j_{r-1} finishes for greedy algo; greedy goes for one with least finish time**
    - **Hence f(i_r) ≤ f(j_r)**

Greedy is optimal

- **[4.3] The greedy algorithm returns an optimal set A**
  - **Proof by contradiction: Assume A is not optimal; lets see what happens**
  - **Let |A| = k and |OPT| = m > k**
  - **$f(i_k) \leq f(j_k) \leq s(j_{k+1})$**
  - **After finishing job j_k, job j_{k+1} is available for greedy – contradicts termination condition for greedy**

Scheduling all intervals: Interval partitioning problem

- **Set of lectures {1, 2, …, n}**
- **Lecture i starts at time si and ends at fi**
- **We need to schedule *all* lectures**
  - **What is the minimum n**