3: Scheduling (Greedy)

Wednesday, March 3, 2021 8:54 PM

Greedy: does the best thing now (doesn't worry about future)
simple heuristic to make next choice

Problem: Scheduling

t	
t_1	1
t_2	4
t_3	3

w	
w_1	5
w_2	1
w_3	2

Input: n tasks,
- time for each task
- importance (weight) for each
(larger weight = more important)

Output: Ordering of jobs $\sigma = (\sigma_1, \sigma_2, \sigma_3, \dots \sigma_n)$ "sigma"
important jobs get done earlier

σ_3 is 3rd job performed in order σ

Ethical Concern? How to choose weights! What tasks are important
Sometimes, the ethical concern is not the algorithm so much as the inputs to the alg. and how those weights are chosen. [explore in pset → reflect → discuss]
[not just an issue for M.L.]

Applications: Jobs on computer CPU

Important Concept: Completion time

ex:

job	time
1	5
2	3

Order: $(2, 1)$ (σ_1, σ_2)

(only one job at a time)

Time →
0, 3, 8
Job 2 completed: $C_2 = 3$
Job 1 completed: $C_1 = 8$

$C_i(\sigma)$ = completion time of job i under ordering σ

Order Should Minimize:

$$A(\sigma) = \sum_{i=1}^{n} w_i C_i(\sigma)$$

σ affects completion time

job	time	weight
1	5	2
2	3	1

What is $A(1,2)$?
A) 3
B) 8
C) 13
D) 18

$(1,2)$
time: 5, 8
job 1, job 2
$C_1 = 5 \quad C_2 = 8$

$A = w_1C_1 + w_2C_2$
$= 1\cdot 8 + 2\cdot 5 = 18$

$(2,1)$
time: 3, 8
job 2, job 1
$C_1 = 3 \quad C_2 = 8$

$A = w_1C_1 + w_2C_2$
$= 1\cdot 3 + 2\cdot 8$
$= 19$

How to Create a Greedy Algorithm

1. Choose a function that assigns a score to each job
   ex: $f(i) = w_i + t_i$ "heuristic"

job	time	weight	f-score
1	5	2	7
2	3	1	4
3	9	4	13

2. Sort jobs by f-score
3. Choose σ = sorted order (increasing or decreasing)

Pros
- Very easy to create
- Fast to run

Cons
- Usually not optimal (unless f is carefully chosen)
- Hard to prove when optimal
  We'll see how!

How to Create a hopefully optimal Greedy Algorithm

1. Choose several f-functions that seem reasonable, given desired output
2. Test on several inputs to see whether give correct ordering → try to find examples where is incorrect to rule out
3. If an f-function always seems to give correct ordering, then try to prove correct.

In Groups

- Come up with at least 2 reasonable f-functions to minimize A.
  (use $+, -, \div$, or $*$ of w_i, t_i)
- Test on

job	time	weight
1	5	2
2	3	1

(To be consistent, will order large f-value jobs first)

A: Good jobs have large weight, short time
- $f_1 = \frac{w_i}{t_i}$
- $f_2 = w_i - t_i$

job	time	weight	f_1	f_2
1	5	2	2/5	-3
2	3	1	$1/3 = 2/6$	-2

If follow f_1 → order $(1,2)$
If follow f_2 → order $(2,1)$

Order	A
1 2	$2\cdot5 + 1\cdot8 = 18$
2 1	$1\cdot3 + 2\cdot8 = 19$

$\boxed{f_1 \text{ is better!}}$

Maybe you do a few more examples and f_1 always gives you the correct ordering. Then you are ready to try to prove it always gives an optimal ordering.

Thm: Greedy algorithm with $f = w_i/t_i$ is optimal for objective function $\sum_{i=1}^{n} w_iC_i$.

Pf: EXCHANGE argument (Proof by Contradiction)

Assume w_i/t_i are distinct $\forall\, i \in \{1, 2, \dots, n\}$

WLOG, relabel so $w_1/t_1 > w_2/t_2 > \dots > w_n/t_n$.

(reorder)

	job	time	weight	f-score
(2)	1	5	2	2/5
(3)	2	3	1	1/3
(1)	3	9	4	4/9

$\sigma = (1,2,3)$ = greedy
$\sigma^* = (3,1,2)$
$\sigma^{*\prime} = (1,3,2)$

Let σ be ordering using greedy, so $\sigma = (1, 2, 3, \dots, n)$
Let σ^* be optimal ordering
For contradiction, assume $\sigma \neq \sigma^*$
Then $\exists$ jobs j, k that are out of numeric order in σ^*

$\sigma^* = (\dots\, k, j \dots)$ where $j < k$
sequential in σ^*

Let $\sigma^{*\prime}$ be σ^* but with j, k switched

$\sigma^* = (\dots\, k, j, \dots)$
$\sigma^{*\prime} = (\dots\, j, k, \dots)$

Q: What is $A(\sigma^{*\prime}) - A(\sigma^*)$

σ^*: time → | .. | k | j | .. |
$\sigma^{*\prime}$: time → | .. | j | k | .. |

T: time to complete first set of jobs

σ^*: | jobs | h | t_k | t_j | jobs |
$\sigma^{*\prime}$: | jobs | h | t_j | t_k | jobs |

$$A(\sigma^*) = \sum w_iC_i + w_k(T + t_k) + w_j(T + t_k + t_j) + \sum w_rC_r$$
$$A(\sigma^{*\prime}) = \sum w_iC_i + w_j(T + t_j) + w_k(T + t_j + t_k) + \sum w_rC_r$$
$$\boxed{A(\sigma^*) - A(\sigma^{*\prime}) = w_jt_k - w_kt_j}$$

$\boxed{\frac{w_j}{t_j} > \frac{w_k}{t_k}} \Rightarrow w_jt_k > w_kt_j$

$A(\sigma^{*\prime}) < A(\sigma^*) \Rightarrow$ Contradiction, b/c we assumed σ^* was optimal, so it should have smallest A-value.

Thus our assumption that σ was not optimal was incorrect and σ is optimal schedule.

Structure of Exchange Proof

1. Assume for contradiction σ^* is optimal, $\sigma^* \neq \sigma$.
   greedy
2. Modify (exchange elements of) σ^* to make it more like σ. (Create $\sigma^{*\prime}$).
3. Show $\sigma^{*\prime}$ is better than σ^*.
4. Thus σ^* is not optimal.
5. Thus σ is optimal

Can apply to every non-greedy strategy
⇓
This shows every strategy except greedy is not optimal

Q: What is the runtime of the greedy scheduling alg.?
A) $O(1)$ B) $O(n)$ C) $O(n\log n)$ D) $O(n^2)$

need to sort by f-value

- Greedy approach: lots of solutions/strategies some are better than others.
- Choice of objective function $\xrightarrow{\text{affects}}$ choice of f

ex: $A = \sum \underbrace{w_iC_i}_{\text{weights weight + time equally}} \rightarrow f = w/t$

ex $A = \sum \underbrace{w_i^2C_i}_{\text{weight more important than time}} \xrightarrow{?} f = w^2/t$

Ethics Conversation based on Pset

- What surprised you about these examples (3/5 compromise, college rankings?
- Which groups are harmed or benefit from the weighting of each of these algorithms, and how does that harm/benefit reinforce systemic inequities?
- Milner says that data can "become a tool of profound social change." For each of these algorithms, how could weights have instead be chosen to work against inequitable power structures? Explain.
- Can you think of another real-life "simple" algorithm (like an averaging algorithm, or a voting algorithm) that is problematic because of its choice of weights?

Dropping w_i/t_i are unique assumption?

where did it show up?

Here → $\frac{w_j}{t_j} \geq \frac{w_k}{t_k} \Rightarrow w_jt_k \geq w_kt_j \Rightarrow A(\sigma^*) \geq A(\sigma^{*\prime})$
contradiction
Not a contradiction!
Proof structure ✗

New Idea: Keep exchanging

Pf Sketch: Choose some labeling s.t.
$w_1/t_1 \geq w_2/t_2 \geq w_3/t_3 \dots$
Let $\sigma = (1, 2, 3, \dots, n)$ (Greedy)
Let σ^* be any other strategy
Will show $A(\sigma^*) \geq A(\sigma) \Rightarrow \sigma$ is optimal

$\sigma^* = (\dots\, k, j \dots)$, $w_j/t_j \geq w_k/t_j$ $\xrightarrow{\text{exchange } j,k}$ $\sigma^{*\prime} = (\dots\, k', j' \dots)$, $w_{j'}/t_{j'} \geq w_{k'}/t_{j'}$ $\xrightarrow{\text{exchange } j',k'}$ … σ

Bubble Sort!

$A(\sigma^*) \geq A(\sigma^{*\prime}) \geq \dots \geq A(\sigma)$

What is the runtime of this greedy alg. (no assumption of unique f-values)?

A) $O(1)$ B) $O(n)$ C) $O(n\log n)$ D) $O(n^2)$

Alg doesn't use Bubblesort ⇒ Alg uses MergeSort.
Proof "uses" Bubblesort