CMSC5706 Topics in Theoretical Computer Science

Week 9: Online Algorithms

Instructor: Shengyu Zhang

Secretary hiring problem

A motivating problem

- *Secretary problem*:
  - We want to hire a new office assistant.
  - There are a number of candidates.
  - We can interview one candidate each day, but we have to decide the acceptance/rejection immediately.

One possible strategy

- On each day, if candidate A is better than the current secretary B, then fire B and hire A.
  - Each has a score. Assume no tie.
- Firing and hiring always have overhead.
  - Say: cost c.
- We'd like to pay this but it'll be good if we could have an estimate first.
- *Question: Assuming that the candidates come in a random order, what's the expected total cost?*

Probability…

- Define a random variable X

  X = # of times we hire a new secretary
- Our question is just to compute

$$\mathbf{E}[cX] = c \cdot \mathbf{E}[X].$$

- By definition,

$$\mathbf{E}[X] = \sum_{x=1}^{n} x \cdot \mathbf{Pr}[X = x].$$

- But this seems complicated to compute.

Indicator variables

- Now we see how to compute it easily, by introducing some new random variables.
- Define $X_i = \begin{cases} 1 & \text{if candidate } i \text{ has been hired} \\ 0 & \text{otherwise} \end{cases}$.
- Then $X = \sum_{i=1}^{n} X_i$.
- Recall the linearity of expectation:

$$\mathbf{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbf{E}[X_i]$$

- We thus have $\mathbf{E}[X] = \sum_{i=1}^{n} \mathbf{E}[X_i]$.

Analysis continued

- What is $\mathbf{E}[X_i]$?
- Recall $X_i = \begin{cases} 1 & \text{if candidate } i \text{ has been hired} \\ 0 & \text{otherwise} \end{cases}$.
- Thus $\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/i$.
  - Candidate i was hired iff she is the best among the first i candidates.
- So $\mathbf{E}[X] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \approx \ln(n)$.
- The average cost is $\ln(n) \cdot c$.

Another strategy

- A more natural scenario is that we only hire once.
- And of course, we hope to hire the best one.
- But the candidates on the market also get other offers. So we need to issue offer fast.
- Interview one candidate each day, and decide acceptance/rejection immediately.
- The candidates come in a random order.

Strategy

- Reject the first k candidates no matter how good they are.
  - Because there may be better ones later.
- After this, hire the first one who is better than all the first k candidates.
- If all the rest $n - k$ are worse than the best one among the first k, then hire the last one.

Pseudo-code

- $best\_score = 0$
- **for** $i = 1$ **to** k

    if $score(i) > best\_score$

        $best\_score = score(i)$

  for $i = k + 1$ **to** n

    if $score(i) > best\_score$

        return(i)

  return n

Next

- We want to determine, for each k, the probability that we hire the best one.
- And then maximize this probability over all k.
- Suppose we hire candidate i.
  - $i > k$ in the strategy (since we choose to reject the first k candidates).
- S: event that we hire the best one.
- S_i: event that we hire the best one, which is candidate i.
- $\mathbf{Pr}[S] = \sum_{i=k+1}^{n} \mathbf{Pr}[S_i]$.

- S_i: candidate i is the best among the n candidates, …
  - probability: $1/n$.
- and candidates $k+1, \ldots, i-1$ are all worse than the best one among $1, \ldots, k$.
  - so that candidates $k+1, \ldots, i-1$ are not hired.
  - probability: $k/(i-1)$. (*The best one among the first* $i-1$ *appears in the first* k.)

Putting together

- $\mathbf{Pr}[S_i] = \frac{1}{n} \cdot \frac{k}{i-1} = \frac{k}{n(i-1)}$.
- So $\mathbf{Pr}[S] = \sum_{i=k+1}^{n} \mathbf{Pr}[S_i]$

$$= \sum_{i=k+1}^{n} \frac{k}{n(i-1)}$$

$$= (k/n) \sum_{i=k}^{n-1} (1/i)$$

$$\approx (k/n)(\ln(n-1) - \ln(k)).$$

- Maximize this over all $k \in \{1, \dots, n\}$ we get

$$k = n/e \approx 0.368 \cdot n$$

  - take derivative with respect to k, and set it equal to 0.
- And the success probability is $1/e \approx 0.368$.

Summary for the Secretary problem

- In the first strategy (*always hire a better one*) we hire around $\ln(n)$ times (in expectation).

- In the second strategy (*hire only once*) we hire the best with probability ≈ 0.368.
  - Reject the first $k = 0.368 \cdot n$ candidates
  - And in the rest hire the first one who beats all the first k ones.

Online vs. Offline

- Almost all algorithms we encountered in this course assume that the entire input is given all at once.
- These are called offline algorithms.
- In Secretary problem.
  - The input is given gradually.
  - We need to respond to each candidate in time.
  - We care about our performance compared to the best one in hindsight.
    - Namely the best one by an offline algorithm.

Online algorithms

- The input is revealed in parts.
- An online algorithm needs to respond to each part (of the input) upon its arrival.
- The responding actions cannot be canceled/revoked later.
- We care about the competitive ratio, which compares the performance of an online algorithm to that of the best offline algorithm.
  - Offline: the entire input is given beforehand.

Ski rental problem

Ski rental

- A person goes to a ski resort for a long vacation.
- Two choices everyday:
  - Rent a ski: $1 per day.
  - Buy a ski: \$$B$ once.
- An unknown factor: the number k of remaining days for ski in this season.
  - When snow melts, the ski resort closes.

Offline algorithm

- If we had known k, then it's easy.
  - If $k < B$, then we should rent everyday. The total cost is k.
  - If $k \geq B$, then we should buy on day 1. The total cost is B.
- In any case, the cost is $\min\{k, B\}$.
- *Question*: *Without knowing k, how to make decision every day?*

Deterministic algorithm

- There is a simple deterministic algorithm s.t. our cost is at most $2 \cdot \min\{k, B\}$.
  - We then say that the algorithm has a competitive ratio of 2.
- Algorithm:
  On each day $j < B$, rent.
  On day B, buy.
- If $k < B$, then our cost is k, which is optimal.
- If $k \geq B$, then our cost is
  $$B - 1 + B = 2B - 1 < 2B = 2 \cdot \min\{k, B\}$$

Randomized algorithm

- It turns out to exist a randomized algorithm with a competitive ratio of $\frac{e}{e-1} \approx 1.58$

- The algorithm uses integer programming and linear programming.

Integer programming

- There is an integer programming to solve the offline version of the ski-rental problem.
- We introduce variables $x, z_1, z_2, \ldots, z_k \in \{0,1\}$.
  - x: indicate whether we eventually buy it.
  - z_i: indicate whether we rent on day i.
  - k: the unknown number of remaining days for ski.
- IP:

$$\begin{aligned} \min \quad & B \cdot x + \sum_{j=1}^{k} z_j \\ s.t. \quad & x + z_j \geq 1, \qquad \forall j \in [k] \\ & x, z_j \in \{0,1\} \qquad \forall j \in [k] \end{aligned}$$

Solution

- It's not hard to see that the optimal solution to the IP is

$$\begin{cases} x = 0, z_j = 1, & \text{if } k < B \\ x = 1, z_j = 0, & \text{if } k \geq B \end{cases}$$

  - same as the previous optimal solution for the offline problem.
- So the IP does solve the offline problem.

Relaxation

- Relax it to LP.
- IP:

$$\begin{aligned} \min \quad & B \cdot x + \textstyle\sum_{j=1}^{k} z_j \\ s.t. \quad & x + z_j \geq 1, && \forall j \in [k] \\ & x, z_j \in \{0,1\} && \forall j \in [k] \end{aligned}$$

- LP:

$$\begin{aligned} \min \quad & B \cdot x + \textstyle\sum_{j=1}^{k} z_j \\ s.t. \quad & x + z_j \geq 1, && \forall j \in [k] \\ & x \geq 0, z_j \geq 0, && \forall j \in [k] \end{aligned}$$

The relaxation doesn't lose anything

- It is easily observed that the LP has the following optimal solution

$$\begin{cases} x = 0, z_j = 1, & \text{if } k < B \\ x = 1, z_j = 0, & \text{if } k \geq B \end{cases}$$

- This is the same as the optimal solution to the IP.
- So the LP relaxation doesn't lose anything.

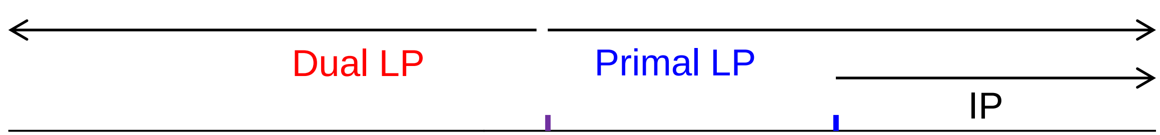
Dual LP

Primal

$$\begin{aligned} \min \quad & Bx + \textstyle\sum_{j=1}^{k} z_j & \\ s.t. \quad & x + z_j \geq 1, & \forall j \\ & x \geq 0, z_j \geq 0, & \forall j \end{aligned}$$

Dual

$$\begin{aligned} \max \quad & \textstyle\sum_{j=1}^{k} y_j & \\ s.t. \quad & \textstyle\sum_{j=1}^{k} y_j \leq B & \forall j \\ & y_j \in [0,1] & \forall j \end{aligned}$$



$OPT_{Dual\ LP} = OPT_{Primal\ LP}$ OPT_{IP}

- Consider the following algorithm, which defines variables x, y_j, z_j.
- $x = 0, y = 0, z = 0$.

  for each new $j = 1,2, \dots, k$

  if $x < 1$

  $x \leftarrow x + \frac{x}{B} + \frac{1}{cB}$, where $c = \left(1 + \frac{1}{B}\right)^B - 1$

  $z_j = 1 - x$

  $y_j = 1$
- Output $x, y_1, \dots, y_k, z_1, \dots, z_k$.

Property 1

$$\begin{aligned} \min \quad & Bx + \sum_{j=1}^{k} z_j & \\ s.t. \quad & x + z_j \geq 1, & \forall j \\ & x \geq 0, z_j \geq 0, & \forall j \end{aligned}$$

- Theorem. The above algorithm produces a feasible solution (x, z_j) to Primal LP and a feasible solution y_j to Dual LP.
- Proof. Feasible to Primal LP:
  - $x \geq 0$ always holds.
    - Starting from 0, x always increases until $x \geq 1$.
  - Before $x \geq 1$: $z_j = 1 - x > 0$, $x + z_j = 1$.
  - After $x \geq 1$: $z_j = 0$, $x + z_j = x \geq 1$.

Property 1

$$\max \quad \sum_{j=1}^{k} y_j$$
$$s.t. \quad \sum_{j=1}^{k} y_j \leq B \quad \forall j$$
$$y_j \in [0,1] \quad \forall j$$

- Theorem. The above algorithm produces a feasible solution (x, z_j) to Primal LP and a feasible solution y_j to Dual LP.
- Proof. Feasible to Dual LP:
  - $y_j \in \{0,1\} \subseteq [0,1]$.
  - To show $\sum_j y_j \leq B$, we need to show that the algorithm stops after $\leq B$ iterations.

- Consider $x_j \stackrel{\text{def}}{=}$ the increment of x in iteration j.
  - Recall: In the algorithm $x \leftarrow x + \frac{x}{B} + \frac{1}{cB}$
- $x_1 = \frac{0}{B} + \frac{1}{cB} = \frac{1}{cB}$,
- $x_2 = \frac{x_1}{B} + \frac{1}{cB} = \frac{1}{cB}\left(1 + \frac{1}{B}\right)$.
- $x_3 = \frac{x_1 + x_2}{B} + \frac{1}{cB} = \frac{1}{cB}\left(\frac{1}{B} + \frac{1+\frac{1}{B}}{B} + 1\right) = \frac{1}{cB}\left(1 + \frac{1}{B}\right)^2$.
- In general, it's not hard to prove that

$$x_j = \frac{1}{cB}\left(1 + \frac{1}{B}\right)^{j-1}$$

- So after B iterations, x increases to

$$\sum_{j=1}^{B} \frac{1}{cB}\left(1+\frac{1}{B}\right)^{j-1} = \frac{\left(1+\frac{1}{B}\right)^{B}-1}{c} = 1.$$

  - since we defined $c = \left(1+\frac{1}{B}\right)^{B} - 1$

- So only the first B dual variables $y_j = 1$, resulting in $\sum_j y_j = B$. Thus y is dual feasible.

Case 1: $k \leq B$

- Primal variables are $x_1, x_2, \ldots, x_k$
  - There is no variable $x_{k+1}, \ldots, x_B$.
  - $x_1 + x_2 + \cdots + x_k \leq 1$.
  - The final $x = x_1 + x_2 + \cdots + x_k \leq 1$.

- Dual variables are $y_1, y_2, \ldots, y_k$
  - There is no variable $y_{k+1}, \ldots, y_B$.
  - $y_1 = y_2 = \cdots = y_k = 1$.

Case 2: $k > B$

- Primal variables are $x_1, x_2, \dots, x_B, x_{B+1}, \dots, x_k$.
  - $x_1 + x_2 + \cdots + x_B = 1$.
  - $x_{B+1} = \cdots = x_k = 0$.
  - The final $x = x_1 + x_2 + \cdots + x_k = 1$.
- Dual variables are $y_1, y_2, \dots, y_B, y_{B+1}, \dots, y_k$.
  - $y_1 = y_2 = \cdots = y_B = 1$.
  - $y_{B+1} = \cdots = y_k = 0$.

Property 2

- The outputted variables x, y_j, z_j satisfy

$$\overbrace{Bx + \sum_j z_j}^{\text{primal obj value}} \leq \left(1 + \frac{1}{c}\right) \overbrace{\sum_j y_j}^{\text{dual obj value}}$$

- Actually, we will show something stronger: In every iteration, the increment of primal obj value is $\leq (1 + 1/c) \cdot$ that of dual.
- The increment of dual is always $y_j = 1$ before x reaches 1.

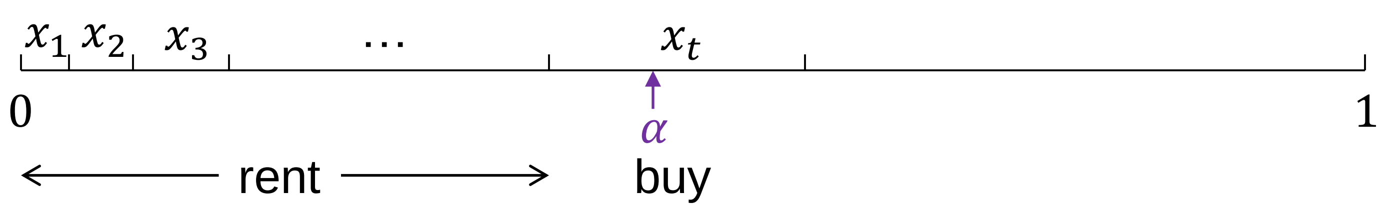
- The increment of primal is

$$Bx_j + z_j = x_{<j} + \frac{1}{c} + 1 - x_{\le j} \le 1 + 1/c.$$

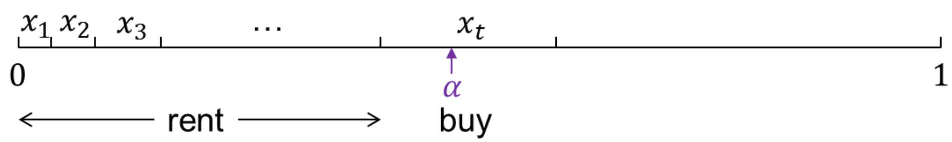
  - $x_{<j} = \sum_{i=1}^{j-1} x_i$ and $x_{\le j} = \sum_{i=1}^{j} x_i$ are the x before and after iteration j, respectively.
  - Recall update: $x \leftarrow x + \frac{x}{B} + \frac{1}{cB}$. So $Bx_j = x_{<j} + \frac{1}{c}$.
  - Recall update: $z_j = 1 - x$. So $z_j = 1 - x_{\le j}$.
- So the increment of primal obj value is at most $(1 + 1/c) \times$ that of dual.

Turning into an online algorithm

- The above algorithm just gives (x, z_j, y_j).
- Now we give an online algorithm based on it.
- Pick $\alpha \in [0,1]$ uniformly at random.
- Suppose t is the first day that $\sum_{j=1}^{t} x_j \geq \alpha$, then rent in all days before t and buy on day t.



Expected cost



- Theorem. $\mathbf{E}[cost] \le \left(1 + \frac{1}{c}\right)\mathrm{OPT}$.
- There are two costs. One is buying cost, and the other is renting cost.
- Obs. $\mathbf{Pr}[\text{buy in day } \boldsymbol{i}] = x_i$.
- So in either case ($k \le B$ or $k > B$),

  $$\mathbf{E}[buying\ cost] = B \sum_{j=1}^{k} x_i = Bx$$

  the first term of the obj function of Primal.
- $\mathbf{Pr}[\text{rent in day } j] = \mathbf{Pr}[\text{no buy in days } 1, \ldots, j]$

  $= 1 - \sum_{i=1}^{j} x_i \le 1 - \sum_{i=1}^{j-1} x_i = z_j$.

- So $\mathbf{E}[renting\ cost] = \sum_{j=1}^{k} z_j$, the second term of the obj function of Primal.
- $\mathbf{E}[cost] = \mathbf{E}[buying\ cost] + \mathbf{E}[renting\ cost]$ $= Bx + \sum_{j=1}^{k} z_j$, the Primal objective value.
- So $\mathbf{E}[cost]$
  $= Primal\ obj$ // above
  $\leq \left(1 + \frac{1}{c}\right) dual\ obj$ // Property 2
  $\leq \left(1 + \frac{1}{c}\right) OPT$. // dual feasible $\leq$ OPT.

- So the online algorithm achieves a competitive ratio of $\left(1 + \frac{1}{c}\right)$.
- Recall that $c = (1 + 1/B)^B - 1$, which is close to $e - 1$ for large B.
- Thus the competitive ratio is $1 + \frac{1}{c} = \frac{e}{e-1} \approx 1.58$, as claimed.

- Optimality: Both deterministic and randomized algorithms are optimal.
  - No better competitive ratio is possible.
- Reference: **The design of competitive online algorithms via a primal dual approach**, Niv Buchbinder and Joseph Naor, *Foundations and Trends in Theoretical Computer Science*, Vol. 3, pp. 93-263, 2007.