Random Order Models

Pre-print Beyond the Worst-Case Analysis of Algorithms
Chapter 11: Random-Order Models - Anupam Gupta, Sahil Singla (2020)

Arthur Monteiro Ferraz - INF2982 2020.2

Agenda

- Secretary Problem
- Estratégia Wait-and-Pick
- Multiple-Secretary Problem
- Modelos de Ordem Aleatória e Programação Inteira

Secretary Problem

- Após adquirir valiosos conhecimentos sobre Algoritmos e Incerteza, os alunos da disciplina receberam diversas propostas de emprego e devem decidir qual proposta aceitar.

- As propostas serão apresentadas em ordem aleatória e, após ser revelada, o aluno deve imediatamente aceitá-la ou recusá-la, não podendo futuramente aceitar uma proposta previamente recusada.

Secretary Problem

- Para cada proposta revelada, é possível saber o seu “valor”, não sendo possível utilizar informações externas

- O número total de propostas é previamente conhecido

- As propostas são pré-definidas e ordenadas aleatoriamente com distribuição uniforme.

Secretary Problem

- Como funciona o jogo:
  - Sabemos previamente o número *n* de propostas que chegarão
  - Em cada tempo *t* é revelada uma proposta
  - Após ver o valor *v* devemos decidir se aceitamos ou recusamos
  - Se recusada, não é possível aceitar a proposta posteriormente
  - O jogo acaba quando aceitarmos uma proposta (ou quando elas acabam)

Queremos maximizar o retorno

Secretary Problem

- Como funciona o jogo:
  - Sabemos previamente o número *n* de propostas que chegarão
  - Em cada tempo *t* é revelada uma proposta
  - Após ver o valor *v* devemos decidir se aceitamos ou recusamos
  - Se recusada, não é possível aceitar a proposta posteriormente
  - O jogo acaba quando aceitarmos uma proposta (ou quando elas acabam)

$n = 4$

Valor = 10

Secretary Problem

- Como funciona o jogo:
  - Sabemos previamente o número *n* de propostas que chegarão
  - Em cada tempo *t* é revelada uma proposta
  - Após ver o valor *v* devemos decidir se aceitamos ou recusamos
  - Se recusada, não é possível aceitar a proposta posteriormente
  - O jogo acaba quando aceitarmos uma proposta (ou quando elas acabam)

n = 4

Valor = 10

Recusou

Secretary Problem

- Como funciona o jogo:
  - Sabemos previamente o número *n* de propostas que chegarão
  - Em cada tempo *t* é revelada uma proposta
  - Após ver o valor *v* devemos decidir se aceitamos ou recusamos
  - Se recusada, não é possível aceitar a proposta posteriormente
  - O jogo acaba quando aceitarmos uma proposta (ou quando elas acabam)

$n = 4$

~~Valor = 10~~

Valor = 15

Secretary Problem

- Como funciona o jogo:
  - Sabemos previamente o número n de propostas que chegarão
  - Em cada tempo t é revelada uma proposta
  - Após ver o valor v devemos decidir se aceitamos ou recusamos
  - Se recusada, não é possível aceitar a proposta posteriormente
  - O jogo acaba quando aceitarmos uma proposta (ou quando elas acabam)

$n = 4$

Valor = 10

Recusou

Valor = 15

Secretary Problem

- Como funciona o jogo:
  - Sabemos previamente o número *n* de propostas que chegarão
  - Em cada tempo *t* é revelada uma proposta
  - Após ver o valor *v* devemos decidir se aceitamos ou recusamos
  - Se recusada, não é possível aceitar a proposta posteriormente
  - O jogo acaba quando aceitarmos uma proposta (ou quando elas acabam)

n = 4

~~Valor = 10~~

~~Valor = 15~~

Valor = 13

Secretary Problem

- Como funciona o jogo:
  - Sabemos previamente o número *n* de propostas que chegarão
  - Em cada tempo *t* é revelada uma proposta
  - Após ver o valor *v* devemos decidir se aceitamos ou recusamos
  - Se recusada, não é possível aceitar a proposta posteriormente
  - O jogo acaba quando aceitarmos uma proposta (ou quando elas acabam)

Valor = 10

Valor = 15

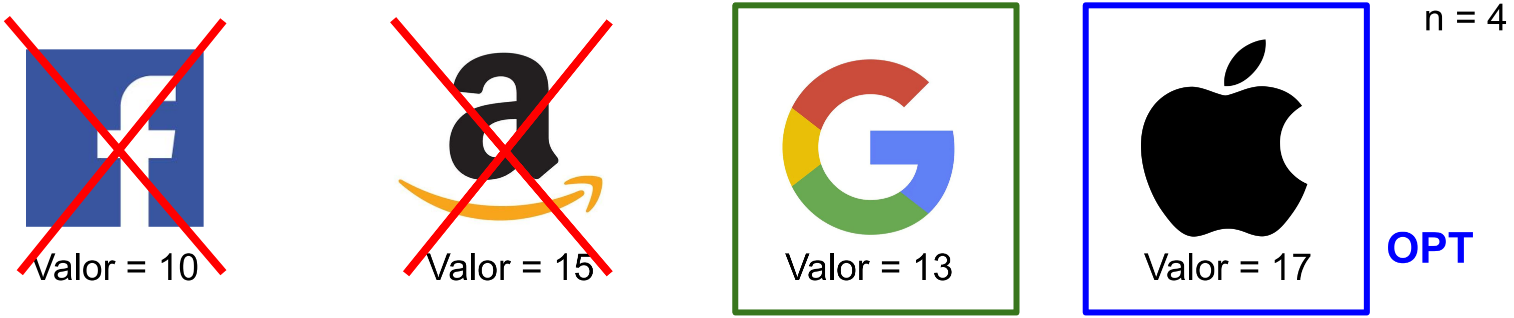
Valor = 13

Aceitou

n = 4

Secretary Problem

- Como funciona o jogo:
  - Sabemos previamente o número *n* de propostas que chegarão
  - Em cada tempo *t* é revelada uma proposta
  - Após ver o valor *v* devemos decidir se aceitamos ou recusamos
  - Se recusada, não é possível aceitar a proposta posteriormente
  - O jogo acaba quando aceitarmos uma proposta (ou quando elas acabam)



Estratégia Wait-and-Pick

“ Se preferir o Sr. Martin a qualquer outra pessoa, se acha que ele é o melhor homem em cuja companhia já esteve, porque haverá de exitar?”

Jane Austen, Emma

Estratégia Wait-and-Pick

- Estratégia que separa a execução do algoritmo em duas fases:
  - Wait - recusar todas as opções apresentadas
  - Pick - aceitar a primeira proposta que é a melhor do que as já vistas

Estratégia Wait-and-Pick

- Algoritmo 50% - vamos recusar metade das opções, então aceitaremos a primeira proposta da segunda metade que for melhor do que as já apresentadas, se houver.

Estratégia Wait-and-Pick

- Algoritmo 50% - vamos recusar metade das opções, então aceitaremos a primeira proposta da segunda metade que for melhor do que as já apresentadas, se houver.

n = 4

Valor = 10	Valor = 15	Valor = 13	Valor = 17

Estratégia Wait-and-Pick

- Algoritmo 50% - vamos recusar metade das opções, então aceitaremos a primeira proposta da segunda metade que for melhor do que as já apresentadas, se houver.

n = 4

Valor = 10

Valor = 15

Valor = 13

Valor = 17

Wait

Pick

Estratégia Wait-and-Pick

- Algoritmo 50% - vamos recusar metade das opções, então aceitaremos a primeira proposta da segunda metade que for melhor do que as já apresentadas, se houver.

n = 4

Valor = 10

Valor = 17

Valor = 13

Valor = 15

Estratégia Wait-and-Pick

- Algoritmo 50% - vamos recusar metade das opções, então aceitaremos a primeira proposta da segunda metade que for melhor do que as já apresentadas, se houver.

n = 4

Valor = 10

Valor = 17

Valor = 13

Valor = 15

Wait

Pick

Nenhuma proposta aceita - valor zero

Estratégia Wait-and-Pick

- Estratégia que separa a execução do algoritmo em duas fases:
  - Wait - recusar todas as opções apresentadas
  - Pick - aceitar a primeira proposta que é a melhor do que as já vistas
- Precisa buscar equilíbrio - se a proporção de Wait for muito grande, podemos descartar a melhor opção. Se for muito pequena, podemos aceitar uma proposta ruim.

Estratégia Wait-and-Pick

- **Teorema:** Estratégia $1/e \approx 37\%$ possui valor esperado pelo menos $1/e$ do OPT e é a estratégia wait-and-pick ótima.

Estratégia Wait-and-Pick

- **Teorema:** Estratégia 1/e ≈ 37% possui valor esperado pelo menos 1/e do OPT e é a estratégia wait-and-pick ótima.
    - **Prova:** vamos partir de um problema com *n* propostas e utilizando a estratégia Wait-and-Pick, sendo *m* o valor a partir do qual começamos a fase de Pick

$$\sum_{t=m+1}^{n} \Pr[v_t \text{ is max}] \cdot \Pr[\text{max among first } t-1 \text{ numbers falls in first } m \text{ positions}]$$

Estratégia Wait-and-Pick

$$\sum_{t=m+1}^{n} \Pr[v_t \text{ is max}] \cdot \Pr[\text{max among first } t-1 \text{ numbers falls in first } m \text{ positions}]$$

$$\overset{(\star)}{=} \sum_{t=m+1}^{n} \frac{1}{n} \cdot \frac{m}{t-1}$$

Estratégia Wait-and-Pick

$$\sum_{t=m+1}^{n} \Pr[v_t \text{ is max}] \cdot \Pr[\text{max among first } t-1 \text{ numbers falls in first } m \text{ positions}]$$

$$\overset{(\star)}{=} \sum_{t=m+1}^{n} \frac{1}{n} \cdot \frac{m}{t-1}$$

$$= \frac{m}{n}\left(H_{n-1} - H_{m-1}\right)$$

$$H_k = 1 + \tfrac{1}{2} + \tfrac{1}{3} + \ldots + \tfrac{1}{k}$$

Estratégia Wait-and-Pick

$$\sum_{t=m+1}^{n} \Pr[v_t \text{ is max}] \cdot \Pr[\text{max among first } t-1 \text{ numbers falls in first } m \text{ positions}]$$

$$\stackrel{(\star)}{=} \sum_{t=m+1}^{n} \frac{1}{n} \cdot \frac{m}{t-1}$$

$$= \frac{m}{n}\left(H_{n-1} - H_{m-1}\right)$$

$$= \frac{m}{n} \ln \frac{n-1}{m-1}$$

$$H_k = 1 + \tfrac{1}{2} + \tfrac{1}{3} + \ldots + \tfrac{1}{k}$$

$$H_k \approx \ln k + 0.57 \text{ for large } k$$

Estratégia Wait-and-Pick

$$\max\left(\frac{m}{n}\ln\frac{n-1}{m-1}\right)=\frac{1}{e} \qquad\qquad m=\frac{n}{e}$$

$$\mathbf{E}(A)\geq\frac{OPT}{e}$$

$\blacksquare$

Estratégia Wait-and-Pick

- **Teorema:** 1/e Wait-and-Pick é melhor algoritmo para o problema da secretária

Estratégia Wait-and-Pick

- **Teorema:** 1/e Wait-and-Pick é melhor algoritmo para o problema da secretária

- Ideia da prova é mostrar que qualquer estratégia que busca maximizar a probabilidade de obter melhor valor pode ser assumida como algum *Wait-and-Pick*

Multiple Secretary Problem

- Vamos supor, que ao invés de aceitar apenas uma proposta de emprego, nós temos a capacidade de ter *k* empregos. Nosso objetivo continua sendo maximizar o valor, só que agora da soma.

Multiple Secretary Problem

- Vamos supor, que ao invés de aceitar apenas uma proposta de emprego, nós temos a capacidade de ter *k* empregos. Nosso objetivo continua sendo maximizar o valor, só qu

Multiple Secretary Problem

- Estratégia **Order-Oblivous**: utiliza basicamente a mesma ideia do *Wait-and-Pick*, mas ao invés de sempre comparar com o melhor já encontrado, compara com o tau-ésimo melhor valor encontrado.

Multiple Secretary Problem

- Estratégia **Order-Oblivous**: utiliza basicamente a mesma ideia do *Wait-and-Pick*, mas ao invés de sempre comparar com o melhor já encontrado, compara com o tau-ésimo melhor valor encontrado.

1. Set $\varepsilon = \delta = O\left(\frac{\log k}{k^{1/3}}\right)$.
2. *First phase:* ignore the first δn items.
   Threshold $\tau \leftarrow$ value of the $(1-\varepsilon)\delta k^{th}$-highest valued item in this ignored set.
3. *Second phase:* pick the first k items seen that have value greater than τ.

Multiple Secretary Problem

- **Teorema:** o algoritmo Order-Oblivious possui valor esperado maior ou igual à

$$V^\star(1 - O(\delta)),\ \ where\ \delta = O\Big(\frac{\log k}{k^{1/3}}\Big)$$

Multiple Secretary Problem

Prova Informal Bound Order-Oblivious:

Vamos chamar de S* o conjunto de valores que pertencem a solução OPT, logo

|S*| = k. Como ignoramos os primeiros δ*n, a quantidade esperada de itens de S* ignorados é δ*k.

Multiple Secretary Problem

Prova Informal Bound Order-Oblivious:

Vamos chamar de v’ o menor valor pertencente a S*. O algoritmo pode falhar por dois motivos:

(i) limite < v’: selecionamos elementos de qualidade inferior à v’. Isso ocorre quando menos de (1-ε)*δ*k elementos estão entre os δ*n primeiros.

Multiple Secretary Problem

Prova Informal Bound Order-Oblivious:

Usando a forma multiplicativa do Chernoff Bound:

$$Pr(Z \leq (1-\rho)\mu) \leq e^{\frac{-\rho^2 \mu}{2}}$$

Temos:

$$Pr(Z \leq (1-\varepsilon)\delta k) \leq e^{\frac{-\varepsilon^2 \delta k}{2}}$$

Multiple Secretary Problem

Prova Informal Bound Order-Oblivious:

(ii) limite > v': selecionamos elementos de qualidade estritamente superior à v'. Ideia é similar ao caso (i) - mesma probabilidade de acontecer

Queremos mostrar uma perda da O(δV*), portanto basta que a probabilidade de falha seja limitada por O(1/polinomio(k)).

$$\delta = \varepsilon = \sqrt[3]{2} \cdot \frac{\log k}{\sqrt[3]{k}}$$

$$e^{\frac{-\varepsilon^2 \delta k}{2}} = \frac{1}{3k}$$

Multiple Secretary Problem

- Estratégia **Order-Adaptive**: se baseia na Order-Oblivious, porém utiliza o fato de que a distribuição é aleatória em todo o intervalo. Dessa forma podemos ajustar o valor do limite mesmo após a fase de ignorar.

- Possui garantia melhor:

$$V^\star\left(1 - O\left(\sqrt{\frac{\log k}{k}}\right)\right)$$

Modelos de Ordem Aleatória e Programação Inteira

- **Problema da Mochila:** devemos escolher dentre um conjunto de d itens para maximizar o valor obtido com uma capacidade máxima k.

- Vamos supor que, ao invés de conhecermos todos os valores previamente, é apresentado o valor de um item, e devemos aceitá-lo ou descartá-lo, assim como no problema da Secretária

Modelos de Ordem Aleatória e Programação Inteira

- Com pequenas adaptações podemos utilizar o algoritmo Order-Adaptive e obter o seguinte bound:

$$(1 - O(\sqrt{(\log d)/k}))$$

Modelos de Ordem Aleatória e Programação Inteira

- Com pequenas adaptações podemos utilizar o algoritmo Order-Adaptive e obter o seguinte bound:

$$(1 - O(\sqrt{(\log d)/k}))$$

- Estimar o limite de aceitação de um Modelo de Ordem Aleatória é análogo a encontrar a variável dual do Problema

Dúvidas?

Referências

- Pre-print Beyond the Worst-Case Analysis of Algorithms Chapter 11: Random-Order Models - Anupam Gupta, Sahil Singla (2020) - [2002.12159.pdf (arxiv.org)]

- Algoritmos para Viver - A Ciência Exata Das Decisões Humanas - Brian Christian & Tom Griffiths