Secretary Problem

Applications and Generalizations

Negar Sakhaei

February 8, 2020

Outline

Online Algorithms

Definition

An online algorithm is one that can process its input piece-by-piece in a serial fashion, without having the entire input available from the start.

- Need instant response, may later turn out not to be optimal
- How to analyze them?

Analysis

Competitive Analysis:

- Input sequence: σ
- Full knowledge optimum: OPT_σ
  Best result of offline algorithm.
- Online Algorithm cost: ALG_σ
- The online algorithm is k-competitive if:

$$\forall\sigma \quad ALG_\sigma \leq k\, OPT_\sigma$$

Probabilistic Analysis:

- Maximize probability of finding optimum solution

Analysis

Competitive Analysis:

- Input sequence: σ
- Full knowledge optimum: OPT_σ
  Best result of offline algorithm.
- Online Algorithm cost: ALG_σ
- The online algorithm is k-competitive if:

$$\forall\sigma \quad ALG_\sigma \leq k\, OPT_\sigma$$

Probabilistic Analysis:

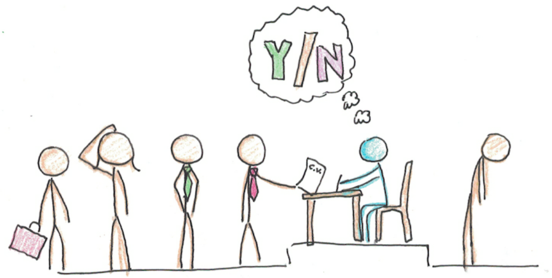
- Maximize probability of finding optimum solution

Definition

Want to hire THE best secretary!

- Single position to fill.
- n **rankable** candidates, one at a time, random order.
- No look-ahead, can't predict the future.
- No undo, can't call back candidates.

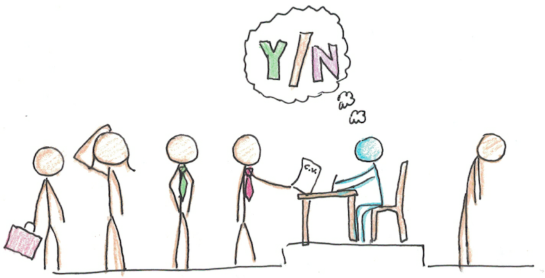
Definition



Want to hire THE best secretary!

- Single position to fill.
- n **rankable** candidates, one at a time, random order.
- No look-ahead, can't predict the future.
- No undo, can't call back candidates.

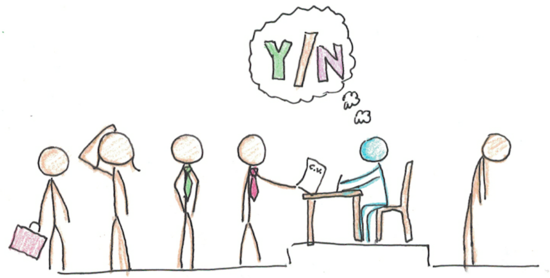
Definition



Want to hire THE best secretary!

- Single position to fill.
- n **rankable** candidates, one at a time, random order.
- No look-ahead, can't predict the future.
- No undo, can't call back candidates.

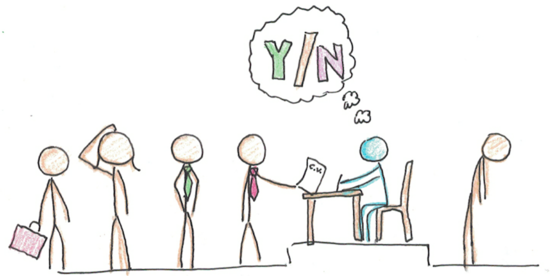
Definition



Want to hire THE best secretary!

- Single position to fill.
- n **rankable** candidates, one at a time, random order.
- No look-ahead, can't predict the future.
- No undo, can't call back candidates.

Definition



Want to hire THE best secretary!

- Single position to fill.
- n **rankable** candidates, one at a time, random order.
- No look-ahead, can't predict the future.
- No undo, can't call back candidates.

Policies

1. Choose the rth candidate.
   Prob. of success: $\frac{1}{n}$. tends to zero as n tends to infinity.
2. **Stopping Rule**: Observe until the rth candidate, accept best candidate afterwards.

Policies

1. Choose the rth candidate.
   Prob. of success: $\frac{1}{n}$. tends to zero as n tends to infinity.
2. **Stopping Rule**: Observe until the rth candidate, accept best candidate afterwards.

Policies

- **Stopping Rule**: Observe until the rth candidate (reject all of them), accept best candidate afterwards.
  - Candidates 1 to $r-1$ are rejected. Set M the best candidate in $[1, r)$
  - Pick first candidate after $r-1$ that is better than M.

It can be shown that the optimal strategy lies in this class of strategies.[4]

Policies

- **Stopping Rule**: Observe until the rth candidate (reject all of them), accept best candidate afterwards.
  - Candidates 1 to $r-1$ are rejected.
    Set M the best candidate in $[1, r)$
  - Pick first candidate after $r-1$ that is better than M.

It can be shown that the optimal strategy lies in this class of strategies.[4]

Policies

- **Stopping Rule**: Observe until the rth candidate (reject all of them), accept best candidate afterwards.
  - Candidates 1 to $r-1$ are rejected.
    Set M the best candidate in $[1, r)$
  - Pick first candidate after $r-1$ that is better than M.

It can be shown that the optimal strategy lies in this class of strategies.[4]

Deriving the Optimal Strategy

$$Pr(r) = \sum_{i=1}^{n} Pr(i \text{ is selected} \cap Pr(i \text{ is the best})$$

$$= \sum_{i=1}^{n} Pr(i \text{ is selected} \mid i \text{ is the best}) \cdot Pr(i \text{ is the best})$$

$$= \sum_{i=1}^{n} Pr(i \text{ is selected} \mid i \text{ is the best}) \cdot \frac{1}{n}$$

$$= \sum_{i=r}^{n} Pr(\max_{j \in [1,r)} x_j = \max_{j \in [1,i)} x_j \mid i \text{ is the best}) \cdot \frac{1}{n}$$

Deriving the Optimal Strategy

$$Pr(r) = \sum_{i=1}^{n} Pr(i \text{ is selected} \cap Pr(i \text{ is the best})$$

$$= \sum_{i=1}^{n} Pr(i \text{ is selected} \mid i \text{ is the best}) \cdot Pr(i \text{ is the best})$$

$$= \sum_{i=1}^{n} Pr(i \text{ is selected} \mid i \text{ is the best}) \cdot \frac{1}{n}$$

$$= \sum_{i=r}^{n} Pr(\max_{j\in[1,r)} x_j = \max_{j\in[1,i)} x_j \mid i \text{ is the best}) \cdot \frac{1}{n}$$

Deriving the Optimal Strategy

$$Pr(r) = \sum_{i=1}^{n} Pr(i \text{ is selected} \cap Pr(i \text{ is the best})$$

$$= \sum_{i=1}^{n} Pr(i \text{ is selected} \mid i \text{ is the best}) \cdot Pr(i \text{ is the best})$$

$$= \sum_{i=1}^{n} Pr(i \text{ is selected} \mid i \text{ is the best}) \cdot \frac{1}{n}$$

$$= \sum_{i=r}^{n} Pr(\max_{j \in [1,r)} x_j = \max_{j \in [1,i)} x_j \mid i \text{ is the best}) \cdot \frac{1}{n}$$

Deriving the Optimal Strategy

$$Pr(r) = \sum_{i=1}^{n} Pr(i \text{ is selected} \cap Pr(i \text{ is the best})$$

$$= \sum_{i=1}^{n} Pr(i \text{ is selected} \mid i \text{ is the best}) \cdot Pr(i \text{ is the best})$$

$$= \sum_{i=1}^{n} Pr(i \text{ is selected} \mid i \text{ is the best}) \cdot \frac{1}{n}$$

$$= \sum_{i=r}^{n} Pr(\max_{j \in [1,r)} x_j = \max_{j \in [1,i)} x_j \mid i \text{ is the best}) \cdot \frac{1}{n}$$

Deriving the Optimal Strategy

$$= \sum_{i=r}^{n} Pr(\max_{j \in [1,r)} x_j = \max_{j \in [1,i)} x_j \mid i \text{ is the best}) \cdot \frac{1}{n}$$

$$= \sum_{i=r}^{n} \frac{r-1}{i-1} \cdot \frac{1}{n}$$

$$= \frac{r-1}{n} \cdot \sum_{i=r}^{n} \frac{1}{i-1}$$

Deriving the Optimal Strategy

$$= \sum_{i=r}^{n} Pr(\max_{j \in [1,r)} x_j = \max_{j \in [1,i)} x_j \mid i \text{ is the best}) \cdot \frac{1}{n}$$

$$= \sum_{i=r}^{n} \frac{r-1}{i-1} \cdot \frac{1}{n}$$

$$= \frac{r-1}{n} \cdot \sum_{i=r}^{n} \frac{1}{i-1}$$

Deriving the Optimal Strategy

$$= \sum_{i=r}^{n} Pr(\max_{j \in [1,r)} x_j = \max_{j \in [1,i)} x_j \mid i \text{ is the best}) \cdot \frac{1}{n}$$

$$= \sum_{i=r}^{n} \frac{r-1}{i-1} \cdot \frac{1}{n}$$

$$= \frac{r-1}{n} \cdot \sum_{i=r}^{n} \frac{1}{i-1}$$

Deriving the Optimal Strategy

Theorem

Optimal cutoff in the stopping rule tends to $\frac{n}{e}$ as n increases.

Proof.

For cutoff value r, probability of success is:

$$P(r) = \frac{r-1}{n} \cdot \sum_{i=r}^{n} \frac{1}{i-1} \tag{1}$$

For large values of n, $x = \lim_{n\to\infty} \frac{r}{n}$ and $t = \lim_{n\to\infty} \frac{i}{n}$, 1 is a Riemann approximation of an integral:

□

Deriving the Optimal Strategy

Theorem

Optimal cutoff in the stopping rule tends to $\frac{n}{e}$ *as* n *increases.*

Proof.

For cutoff value r, probability of success is:

$$P(r) = \frac{r-1}{n} \cdot \sum_{i=r}^{n} \frac{1}{i-1} \tag{1}$$

For large values of n, $x = \lim_{n\to\infty} \frac{r}{n}$ and $t = \lim_{n\to\infty} \frac{i}{n}$, 1 is a Riemann approximation of an integral:

□

Deriving the Optimal Strategy

Theorem

Optimal cutoff in the stopping rule tends to $\frac{n}{e}$ *as* n *increases.*

Proof.

For large values of n, $x = \lim_{n\to\infty} \frac{r}{n}$ and $t = \lim_{n\to\infty} \frac{i}{n}$, 1 is a Riemann approximation of an integral:

$$P(x) = x \int_x^1 \frac{1}{t}\, \mathrm{d}t = -x \ln x \tag{2}$$

Now to find the optimal r:

$$P'(x) = -\ln x - 1 = 0 \Rightarrow x = \frac{1}{e}$$

□

Deriving the Optimal Strategy

Theorem

Optimal cutoff in the stopping rule tends to $\frac{n}{e}$ as n increases.

Proof.

For large values of n, $x = \lim_{n\to\infty} \frac{r}{n}$ and $t = \lim_{n\to\infty} \frac{i}{n}$, 1 is a Riemann approximation of an integral:

$$P(x) = x \int_x^1 \frac{1}{t}\,\mathrm{d}t = -x \ln x \tag{2}$$

Now to find the optimal r:

$$P'(x) = -\ln x - 1 = 0 \Rightarrow x = \frac{1}{e}$$

□

Deriving the Optimal Strategy

Theorem

Optimal cutoff in the stopping rule tends to $\frac{n}{e}$ as n increases.

Proof.

For large values of n, $x = \lim_{n\to\infty} \frac{r}{n}$ and $t = \lim_{n\to\infty} \frac{i}{n}$, 1 is a Riemann approximation of an integral:

$$P(x) = x \int_x^1 \frac{1}{t}\,\mathrm{d}t = -x \ln x \tag{2}$$

Now to find the optimal r:

$$P'(x) = -\ln x - 1 = 0 \Rightarrow x = \frac{1}{e}$$

$\square$

1 Introduction

2 Secretary Problem
- Applications
- Variations

3 Matroid Secretary Problem

Real-Life Applications

- Apartment hunting! [3] [7]
  Estimated n, used secretary problem to limit the hunting time.
  Optimal result.
- **Kepler's Problem**(1611): Wanted to find a new wife!
  11 candidates, married the 5th one. Not sure about the objective or the algorithm.[4]
  Apparently optimal result!

Spherical Cow

- Needs knowing the exact value of n in advance.
- Assumes no other information about the candidates.
- No callbacks at all.
- Ranks sometimes not easily determined.
- Hiring the second-best is as bad as hiring the worst.

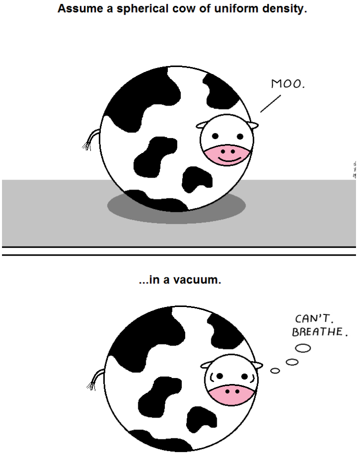


Figure: Spherical Cow

How to Solve Them?

- Needs knowing the exact value of n in advance:
  Suppose n is a random variable with a known distribution.
- No callbacks at all:
  Set a cost for each callback and minimize the overall cost.
- Hiring the second-best is as bad as hiring the worst.
  - Minimize rank: Average rank $O(1)$.[6]
  - Maximize payoff: cutoff at $\sqrt[2]{n}$, result tends to maximum at infinity.[2]

How to Solve Them?

- Needs knowing the exact value of n in advance:
  Suppose n is a random variable with a known distribution.
- No callbacks at all:
  Set a cost for each callback and minimize the overall cost.
- Hiring the second-best is as bad as hiring the worst.
  - Minimize rank: Average rank $O(1)$.[6]
  - Maximize payoff: cutoff at $\sqrt[2]{n}$, result tends to maximum at infinity.[2]

How to Solve Them?

- Needs knowing the exact value of n in advance:
  Suppose n is a random variable with a known distribution.
- No callbacks at all:
  Set a cost for each callback and minimize the overall cost.
- Hiring the second-best is as bad as hiring the worst.
  - Minimize rank: Average rank $O(1)$.[6]
  - Maximize payoff: cutoff at $\sqrt[2]{n}$, result tends to maximum at infinity.[2]

How to Solve Them?

- Needs knowing the exact value of n in advance:
  Suppose n is a random variable with a known distribution.
- No callbacks at all:
  Set a cost for each callback and minimize the overall cost.
- Hiring the second-best is as bad as hiring the worst.
  - Minimize rank: Average rank $O(1)$.[6]
  - Maximize payoff: cutoff at $\sqrt[2]{n}$, result tends to maximum at infinity.[2]

Googol Game

Definition

Ask someone to take as many slips of paper as she pleases, and on each slip write a different positive number. These slips are turned face down and shuffled. One at a time you turn the slips. The aim is to stop turning when you come to the number you guess is the largest of the series.

- Two-person zero-sum game!
- Depends on how Alice chooses the numbers.

Googol Game

Definition

Ask someone to take as many slips of paper as she pleases, and on each slip write a different positive number. These slips are turned face down and shuffled. One at a time you turn the slips. The aim is to stop turning when you come to the number you guess is the largest of the series.

- Two-person zero-sum game!
- Depends on how Alice chooses the numbers.

Googol Game

Definition

Ask someone to take as many slips of paper as she pleases, and on each slip write a different positive number. These slips are turned face down and shuffled. One at a time you turn the slips. The aim is to stop turning when you come to the number you guess is the largest of the series.

- Two-person zero-sum game!
- Depends on how Alice chooses the numbers.

Models

The difficulty of the problem changes depending on the information we know beforehand about the weights.[5]

- **Full Information model**: Chosen i.i.d. from a known distribution.
- **Partial Information model**: Chosen i.i.d. from an unknown distribution.
- **Random Assignment model**: Adversary chooses weights, assigned using a uniform random one-to-one correspondence.
- **Zero information model**: Adversary assigns.

Models

The difficulty of the problem changes depending on the information we know beforehand about the weights.[5]

- **Full Information model**: Chosen i.i.d. from a known distribution.
- **Partial Information model**: Chosen i.i.d. from an unknown distribution.
- **Random Assignment model**: Adversary chooses weights, assigned using a uniform random one-to-one correspondence.
- **Zero information model**: Adversary assigns.

Models

The difficulty of the problem changes depending on the information we know beforehand about the weights.[5]

- **Full Information model**: Chosen i.i.d. from a known distribution.
- **Partial Information model**: Chosen i.i.d. from an unknown distribution.
- **Random Assignment model**: Adversary chooses weights, assigned using a uniform random one-to-one correspondence.
- **Zero information model**: Adversary assigns.

Models

The difficulty of the problem changes depending on the information we know beforehand about the weights.[5]

- **Full Information model**: Chosen i.i.d. from a known distribution.
- **Partial Information model**: Chosen i.i.d. from an unknown distribution.
- **Random Assignment model**: Adversary chooses weights, assigned using a uniform random one-to-one correspondence.
- **Zero information model**: Adversary assigns.

Definition

A matroid $M = (S, \mathcal{I})$ is a finite ground set S together with a collection of sets $\mathcal{I} \subseteq 2^S$, known as the independent sets, satisfying the following axioms:

- If $I \in \mathcal{I}$ and $J \subseteq I$ then $J \in \mathcal{I}$.
- If $I, J \in \mathcal{I}$ and $|J| > |I|$, then there exists an element $z \in J \backslash I$ such that $I \cup \{z\} \in \mathcal{I}$.

The rank of a matroid is the maximum size of an independent set in the matroid.

Definition

A matroid $M = (S, \mathcal{I})$ is a finite ground set S together with a collection of sets $\mathcal{I} \subseteq 2^S$, known as the independent sets, satisfying the following axioms:

- If $I \in \mathcal{I}$ and $J \subseteq I$ then $J \in \mathcal{I}$.
- If $I, J \in \mathcal{I}$ and $|J| > |I|$, then there exists an element $z \in J \backslash I$ such that $I \cup \{z\} \in \mathcal{I}$.

The rank of a matroid is the maximum size of an independent set in the matroid.

Definition

A matroid $M = (S, \mathcal{I})$ is a finite ground set S together with a collection of sets $\mathcal{I} \subseteq 2^S$, known as the independent sets, satisfying the following axioms:

- If $I \in \mathcal{I}$ and $J \subseteq I$ then $J \in \mathcal{I}$.
- If $I, J \in \mathcal{I}$ and $|J| > |I|$, then there exists an element $z \in J \backslash I$ such that $I \cup \{z\} \in \mathcal{I}$.

The rank of a matroid is the maximum size of an independent set in the matroid.

Definition

A matroid $M = (S, \mathcal{I})$ is a finite ground set S together with a collection of sets $\mathcal{I} \subseteq 2^S$, known as the independent sets, satisfying the following axioms:

- If $I \in \mathcal{I}$ and $J \subseteq I$ then $J \in \mathcal{I}$.
- If $I, J \in \mathcal{I}$ and $|J| > |I|$, then there exists an element $z \in J \backslash I$ such that $I \cup \{z\} \in \mathcal{I}$.

The rank of a matroid is the maximum size of an independent set in the matroid.

Examples

- **Uniform Matroids:** For any ground set S and a specific k, let $I \in \mathcal{I}$ if $|I| \leq k$. Denote this matroid U_S^k
- **Graphic Matroids:** For an undirected graph $G = (V, E)$, let the ground set S be the set E of edges of the graph. The matroid $M(G)$, sometimes called the cycle matroid of G, is defined as $M(G) = (E, \mathcal{I})$ where $\mathcal{I} = \{F \subseteq E \mid F \text{ is acyclic}\}$.

Examples

- **Uniform Matroids:** For any ground set S and a specific k, let $I \in \mathcal{I}$ if $|I| \leq k$. Denote this matroid U_S^k
- **Graphic Matroids:** For an undirected graph $G = (V, E)$, let the ground set S be the set E of edges of the graph. The matroid $M(G)$, sometimes called the cycle matroid of G, is defined as $M(G) = (E, \mathcal{I})$ where $\mathcal{I} = \{F \subseteq E \mid F \text{ is acyclic}\}$.

Problem Formulation

Definition

There is a matroid $(S; \mathcal{I})$, and a weight function assigning $w(i)$ to each element $i \in S$. We wish to design an algorithm which given the matroid structure (but not the weights $w(i)$) selects online an independent set of maximal weight.

- The ground set of the matroid is presented in random order.
- When an element i arrives, we learn the weight $w(i)$.

Problem Formulation

Definition

There is a matroid $(S; \mathcal{I})$, and a weight function assigning $w(i)$ to each element $i \in S$. We wish to design an algorithm which given the matroid structure (but not the weights $w(i)$) selects online an independent set of maximal weight.

- The ground set of the matroid is presented in random order.
- When an element i arrives, we learn the weight $w(i)$.

Problem Formulation

Definition

There is a matroid $(S;\mathcal{I})$, and a weight function assigning $w(i)$ to each element $i \in S$. We wish to design an algorithm which given the matroid structure (but not the weights $w(i)$) selects online an independent set of maximal weight.

- The ground set of the matroid is presented in random order.
- When an element i arrives, we learn the weight $w(i)$.

Models

- **Full Information model**: Chosen i.i.d. from a known distribution.
- **Partial Information model**: Chosen i.i.d. from an unknown distribution.
- **Random Assignment model**: Adversary chooses weights, assigned using a uniform random one-to-one correspondence.
  For general matroids, a $c-$competitive algorithm exists.
- **Zero information model**: Adversary assigns.
  For general matroids, a $(log\ r)-$competitive algorithm exists. [1]

Models

- **Full Information model**: Chosen i.i.d. from a known distribution.
- **Partial Information model**: Chosen i.i.d. from an unknown distribution.
- **Random Assignment model**: Adversary chooses weights, assigned using a uniform random one-to-one correspondence. For general matroids, a $c-$competitive algorithm exists.
- **Zero information model**: Adversary assigns.
  For general matroids, a $(log\ r)-$competitive algorithm exists. [1]

Models

- **Full Information model**: Chosen i.i.d. from a known distribution.
- **Partial Information model**: Chosen i.i.d. from an unknown distribution.
- **Random Assignment model**: Adversary chooses weights, assigned using a uniform random one-to-one correspondence. For general matroids, a $c-$competitive algorithm exists.
- **Zero information model**: Adversary assigns. For general matroids, a $(log\ r)-$competitive algorithm exists. [1]

General Framework

- **Standard Version**: Set U to the set of all candidates, $\mathcal{I}$ to all the singleton sets and the empty set.
- **Multiple Choice Secretary Problem version**: Set the matroid to U_S^k for at most k choices.

General Framework

- **Standard Version**: Set U to the set of all candidates, $\mathcal{I}$ to all the singleton sets and the empty set.
- **Multiple Choice Secretary Problem version**: Set the matroid to U_S^k for at most k choices.

Domains

- **Unit-Demand Domain**
- **Graphical Matroids**
- **Truncated Partition Matroids**

Domains

- **Unit-Demand Domain**
- **Graphical Matroids**
- **Truncated Partition Matroids**

Domains

- **Unit-Demand Domain**
- **Graphical Matroids**
- **Truncated Partition Matroids**

Refrences I

Babaioff, M., Immorlica, N. & Kleinberg, R. *Matroids, Secretary Problems, and Online Mechanisms.* in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Society for Industrial and Applied Mathematics, New Orleans, Louisiana, 2007), 434–443. ISBN: 9780898716245.

Bearden, N. Skip the Square Root of n: A New Secretary Problem. (Jan. 2005).

Blitzstein, J. *Is the solution to the secretary problem ever applied to real life situations?.* https://qr.ae/TlzK7R.

Ferguson, T. S. Who Solved the Secretary Problem? *Statist. Sci.* **4,** 282–289. https://doi.org/10.1214/ss/1177012493 (Aug. 1989).

Refrences II

Soto, J. A. Matroid Secretary Problem in the Random Assignment Model. *CoRR* **abs/1007.2152.** arXiv: 1007.2152. http://arxiv.org/abs/1007.2152 (2010).

Trevisan, L. *Lecture notes in Optimization.* Mar. 2011.

Wees, D. *How I used mathematics to choose my next apartment.* https://davidwees.com/content/how-i-used-mathematics-choose-my-next-apartment/.

Thank You!