Advances on Matroid Secretary Problem: Free Order and Laminar Case

Patrick Jaillet - **José A. Soto** - Rico Zenklusen

MIT | U.Chile & TU-Berlin | Johns Hopkins University

May 16th, 2013

Matroid Secretary Problem: Outline

Matroid Secretary Problem: Outline

1. Introduction
   - Classic Secretary Problem
   - Generalized Secretary Problem
   - Matroid Secretary Problem

2. Laminar matroids

3. Free Order Model Variant

Secretary Problem

Classical Problem: Select top element of an n-stream.

- Hire one person from n candidates arriving in unif. random order.
- Each person reveals a hidden weight during interview.
- Rule: Must decide during the interview.

Secretary Problem

Classical Problem: Select top element of an n-stream.

- Hire one person from n candidates arriving in unif. random order.
- Each person reveals a hidden weight during interview.
- Rule: Must decide during the interview.

Best algorithm (variant of Lindley / Dynkin 60's)

1. Wait until $\mathrm{Bin}(n, 1/e)$ elements have revealed its weight.
2. Select the first record among remaining ones.

This return the top candidate with probability $1/e$.

Generalizations

Generalized secretary problems (random order).

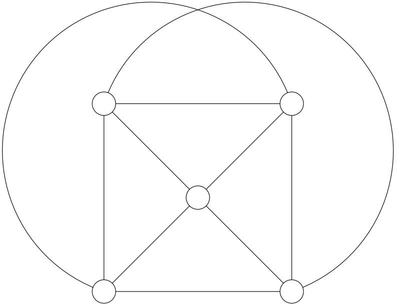
[Babaioff, Immorlica, Kleinberg 2007]

- Select a subset of elements of a stream (one by one).
- Each element reveals a hidden weight during interview.
- Rule: Must decide during the interview.
- The selected subset must belong to a fixed family of feasible sets (closed for inclusion).

Example 1: Select at most r candidates.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

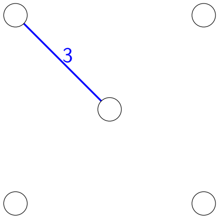


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

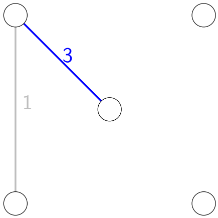


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

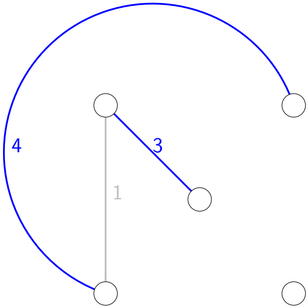


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

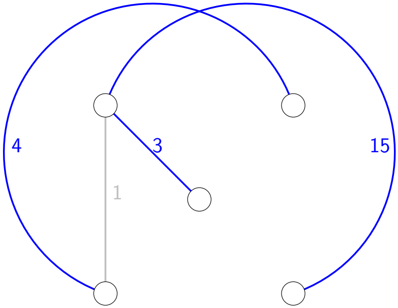


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

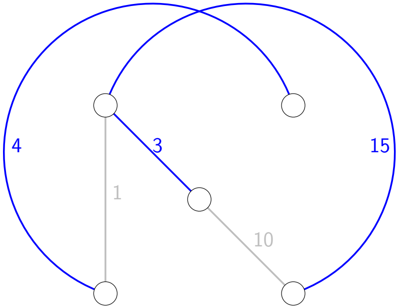


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

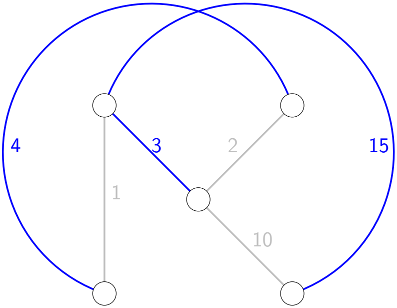


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

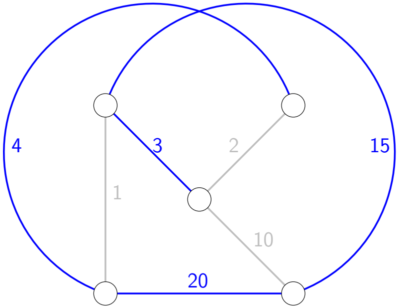


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

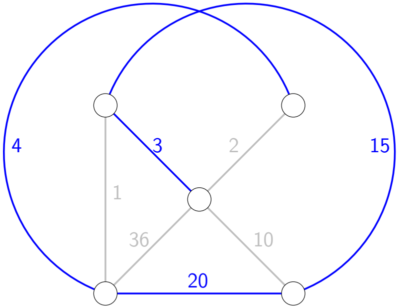


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

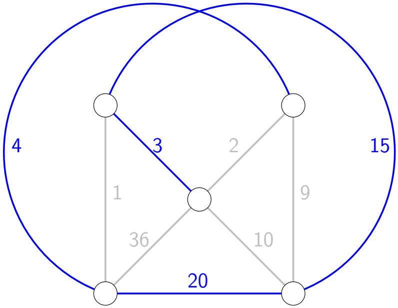


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

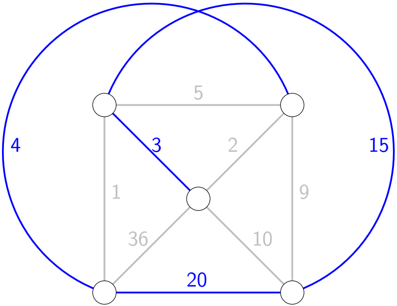


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.

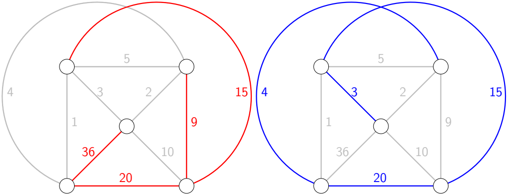


Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

Example 2: Acyclic subgraphs.

- Want: High weight forest.
- Hidden weights are revealed in uniform random order.



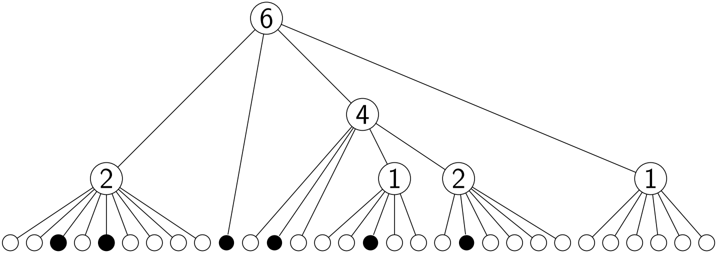
$w(\text{OPT}) = 80$ $w(\text{ALG}) = 42$

Competitive ratio: $\frac{w(\text{OPT})}{\mathbb{E}[w(\text{ALG})]}$.

Rules

- Accept or reject an element **when its weight is revealed.**
- Accepted elements must form a forest at every moment.

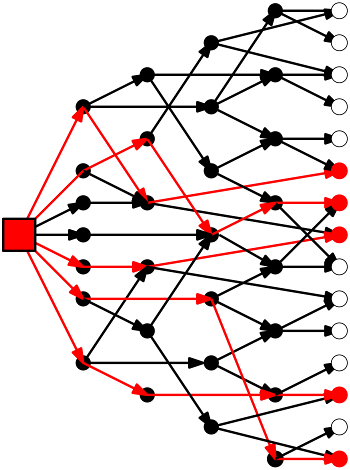
Example 3: Job offerings with quotas.



Want to hire 6 new professors with some quotas:

- Computer Science department can hire at most 2 new professors.
- Physics department can hire at most 1 new position.
- Math department can hire at most 4 new positions.
  - At most 1 of them can be a logician.
  - At most 2 of them can be probabilists.

Example 4: Communication Network



Can only serve clients via disjoint paths.

Matroid Secretary Problem

Generalized secretary problems in which the feasible sets are the independent sets of a matroid.

Matroid Secretary Problem

Generalized secretary problems in which the feasible sets are the independent sets of a matroid.

Reminder: Matroid $M = (E, \mathcal{I})$.

E: ground set of elements.

$\mathcal{I}$: independent sets satisfying:

- $\emptyset \in \mathcal{I}$.
- If $A \in \mathcal{I}$ then every subset $A' \subseteq A$ is in $\mathcal{I}$.
- If $A, B \in \mathcal{I}$ and $|A| < |B|$ then $\exists y \in B \colon A \cup \{y\} \in \mathcal{I}$.

Matroid Secretary Problem

Generalized secretary problems in which the feasible sets are the independent sets of a matroid.

Reminder: Matroid $M = (E, \mathcal{I})$.

E: ground set of elements.

$\mathcal{I}$: independent sets satisfying:

- $\emptyset \in \mathcal{I}$.
- If $A \in \mathcal{I}$ then every subset $A' \subseteq A$ is in $\mathcal{I}$.
- If $A, B \in \mathcal{I}$ and $|A| < |B|$ then $\exists y \in B$: $A \cup \{y\} \in \mathcal{I}$.

Generalize linear independence. For $X \subseteq E$:

- $\mathrm{rk}(X)$ is the size of largest independent set in X.
- $\mathrm{span}(X)$ is the largest set containing X with $\mathrm{rk}(X) = \mathrm{rk}(\mathrm{span}(X))$.

Examples

Linear matroids.

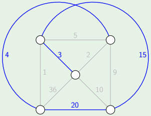
E: Finite family of vectors.
$\mathcal{I}$: Linearly independent set.

Partition matroids.

E: $E_1 \cup \cdots \cup E_k$.
$\mathcal{I}$: $I \subseteq E$ with $|E_i \cap I| \leq b_i$.

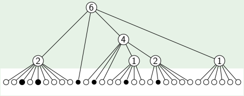
Graphic matroids.

E: Edges of a graph.
$\mathcal{I}$: Forests.



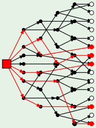
Laminar.

E: Leaves of a tree.
$\mathcal{I}$: Sets satisfying internal node capacities



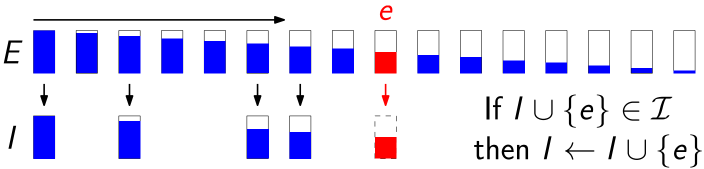
Gammoids.

E: Clients in a directed network.
$\mathcal{I}$: Sets that can be connected to a given server on disjoint paths.



Offline Greedy algorithms

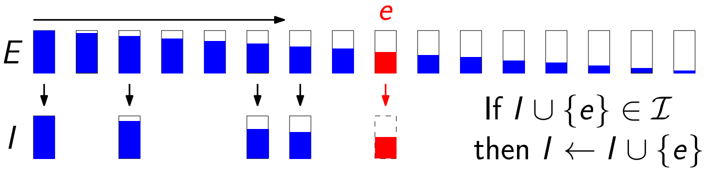
Sorted Greedy (incremental greedy)



If $I \cup \{e\} \in \mathcal{I}$
then $I \leftarrow I \cup \{e\}$.

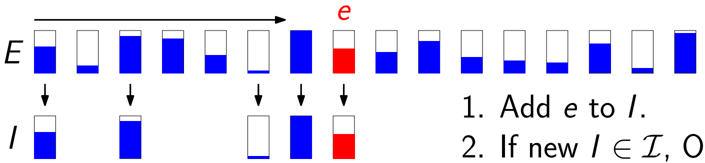
Offline Greedy algorithms

Sorted Greedy (incremental greedy)



If $I \cup \{e\} \in \mathcal{I}$
then $I \leftarrow I \cup \{e\}$.

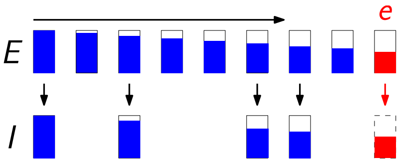
Unsorted Greedy (swap greedy)



1. Add e to I.
2. If new $I \in \mathcal{I}$, OK.

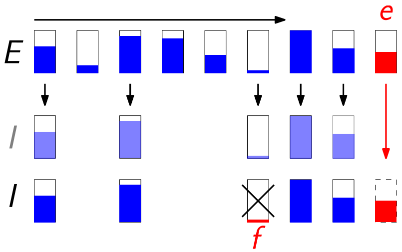
Offline Greedy algorithms

Sorted Greedy (incremental greedy)



If $I \cup \{e\} \in \mathcal{I}$
then $I \leftarrow I \cup \{e\}$.

Unsorted Greedy (swap greedy)



1. Add e to I.
2. If new $I \in \mathcal{I}$, OK.
3. Otherwise remove smallest f such that $I \setminus \{f\} \in \mathcal{I}$

Previous work on Matroid Secretary Problem

- Conjecture [BIK07]: There is an $O(1)$-competitive algorithm for random order of MSP.

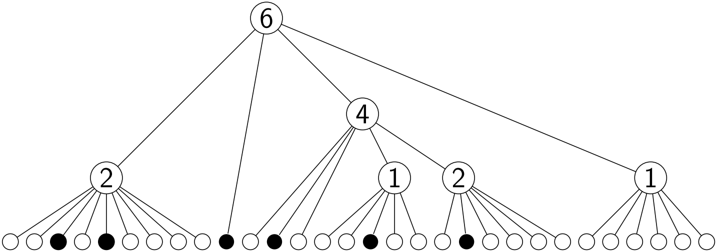
Previous work on Matroid Secretary Problem

- Conjecture [BIK07]: There is an $O(1)$-competitive algorithm for random order of MSP.

- [BIK07] $O(\log \mathrm{rk}(M))$ for general matroids.
- [CL12] $O(\sqrt{\log \mathrm{rk}(M)})$ for general matroids.
- $O(1)$ for:
  - [K05] Partition.
  - [BIK07,KP09] Graphic.
  - [BIK07,DP08,KP09] Transversal.
  - [S11] Cographic.
  - [IW11, JSZ] Laminar.
  - [DK12] Regular.
  - Other cases (low density, sparse linear, truncations, parallel extensions).

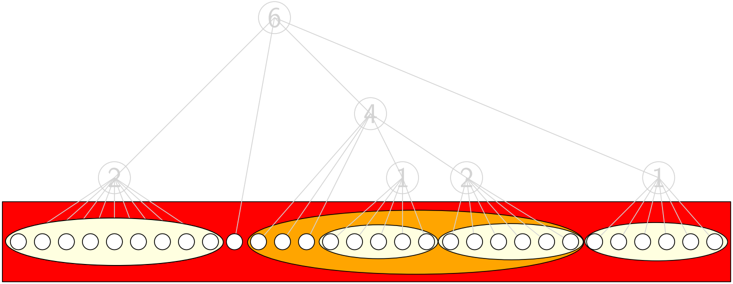
Matroid Secretary Problem: Outline

Laminar Matroids



T: Rooted tree with positive capacities $b(v)$ on internal nodes.
E: Leaves.
$I \subseteq E$ is independent iff $|I \cap L(v)| \leq b(v)$, for every internal v.

Laminar Matroids



T: Rooted tree with positive capacities $b(v)$ on internal nodes.
E: Leaves.
$I \subseteq E$ is independent iff $|I \cap L(v)| \leq b(v)$, for every internal v.

Important: Each v correspond to a consecutive interval of E.
These intervals form a laminar family.

Results.

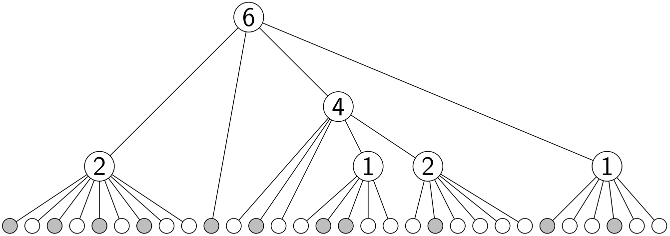
There is a very involved algorithm by Im and Wu (2011)
Large constant $16000/3$-competitive.

Theorem [JSZ12]

There is a simple $3\sqrt{3}e \approx 14.12$-competitive algorithm.

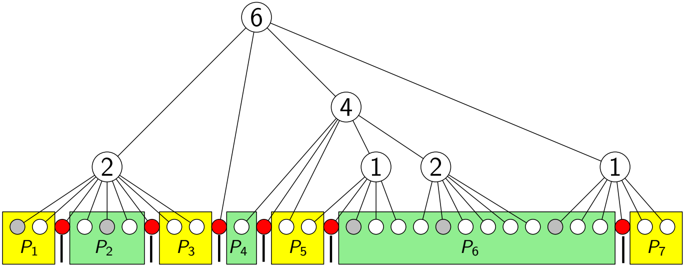
Here: I will show a $16e$-competitive algorithm.

Laminar Matroid algorithm:



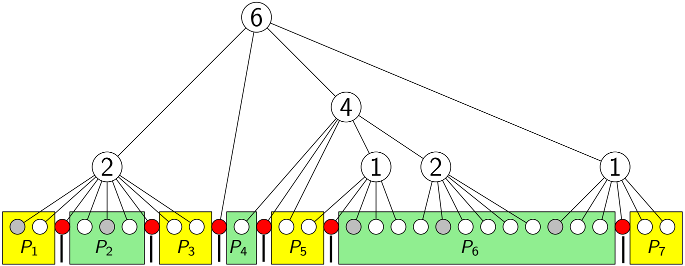
1. $A \leftarrow$ first $\mathrm{Bin}(n, 1/2)$ elements revealed. ● $\leftarrow A$.
2. Use $\mathrm{OPT}(A)$ to divide $E \setminus A$ into intervals $P_1, P_2, \ldots, P_k$.
3. $\mathcal{S} = \begin{cases} \text{Even intervals,} & \text{with prob. 1/2.} \\ \text{Odd intervals,} & \text{with prob. 1/2.} \end{cases}$
4. Run e-competitive alg. to select top element of each interval in $\mathcal{S}$.

Laminar Matroid algorithm:



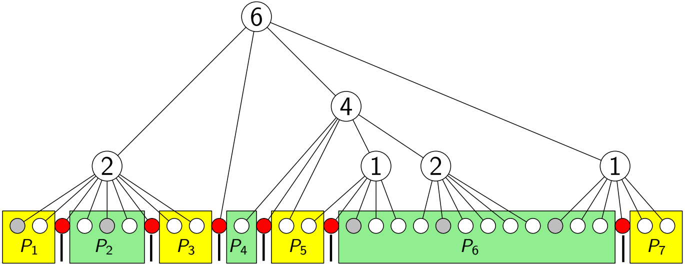
1. $A \leftarrow$ first $\mathrm{Bin}(n, 1/2)$ elements revealed. ● $\leftarrow A$.
2. Use $\mathrm{OPT}(A)$ to divide $E \setminus A$ into intervals $P_1, P_2, \ldots, P_k$.
3. $\mathcal{S} = \begin{cases} \text{Even intervals,} & \text{with prob. } 1/2. \\ \text{Odd intervals,} & \text{with prob. } 1/2. \end{cases}$
4. Run e-competitive alg. to select top element of each interval in $\mathcal{S}$.

Laminar Matroid algorithm:



1. $A \leftarrow$ first $\mathrm{Bin}(n, 1/2)$ elements revealed. ● $\leftarrow A$.
2. Use $\mathrm{OPT}(A)$ to divide $E \setminus A$ into intervals $P_1, P_2, \ldots, P_k$.
3. $\mathcal{S} = \begin{cases} \text{Even intervals,} & \text{with prob. } 1/2. \\ \text{Odd intervals,} & \text{with prob. } 1/2. \end{cases}$
4. Run e-competitive alg. to select top element of each interval in $\mathcal{S}$.

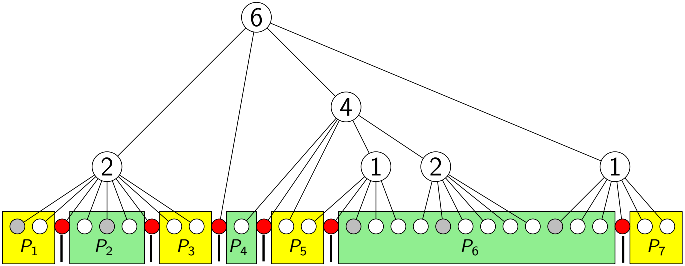
Laminar Matroid algorithm:



1. $A \leftarrow$ first $\mathrm{Bin}(n, 1/2)$ elements revealed. ● $\leftarrow A$.
2. Use $\mathrm{OPT}(A)$ to divide $E \setminus A$ into intervals $P_1, P_2, \ldots, P_k$.
3. $\mathcal{S} = \begin{cases} \text{Even intervals,} & \text{with prob. } 1/2. \\ \text{Odd intervals,} & \text{with prob. } 1/2. \end{cases}$
4. Run e-competitive alg. to select top element of each interval in $\mathcal{S}$.

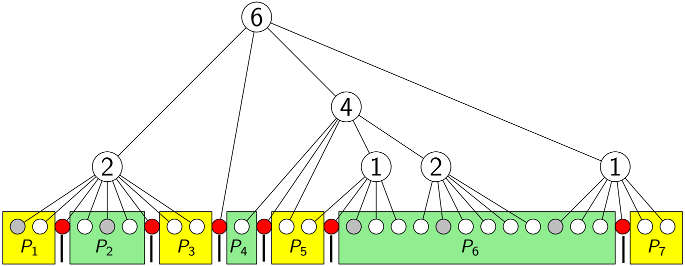
Correctness

Let $I \subseteq \bigcup \mathcal{S}$ and $|I \cap P| \leq 1$ for each $P \in \mathcal{S}$ then I is independent.



Correctness

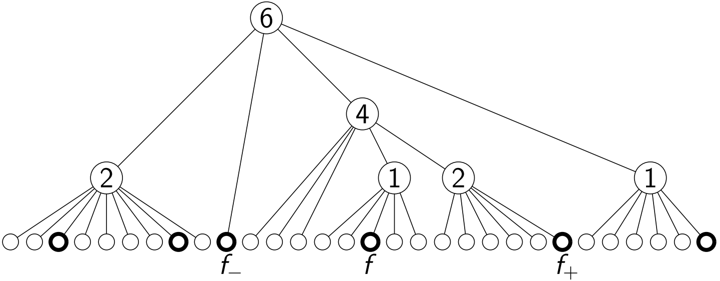
Let $I \subseteq \bigcup \mathcal{S}$ and $|I \cap P| \leq 1$ for each $P \in \mathcal{S}$ then I is independent.



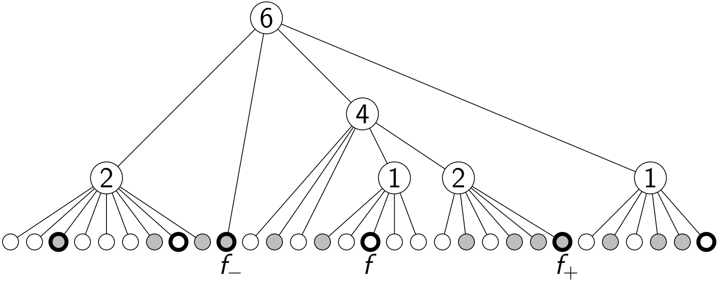
Proof: Let v be an internal node.

- If $|I \cap L(v)| \leq 1$, we are OK.
- If $|I \cap L(v)| \geq 2$.
  Between every pair of I there are ≥ 2 elements of $\mathrm{OPT}(A)$.
  Then: $|I \cap L(v)| \leq |\mathrm{OPT}(A) \cap L(v)| \leq b(v)$.

Analysis sketch: Let f_-, f, f_+ consecutive in OPT.

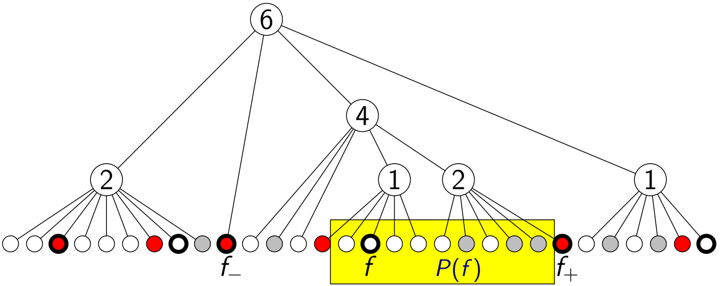


Analysis sketch: Let f_-, f, f_+ consecutive in OPT.



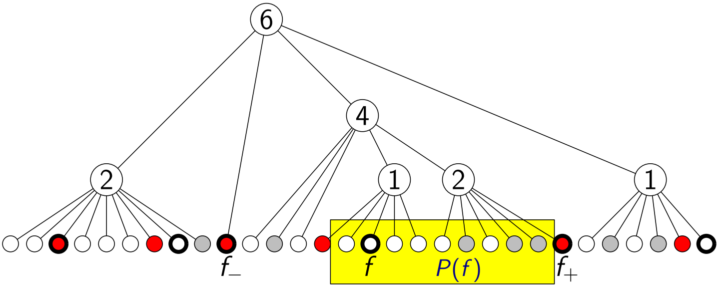
- With prob 1/8: $f_- \in A$, $f \notin A$, $f_+ \in A$.

Analysis sketch: Let f_-, f, f_+ consecutive in OPT.



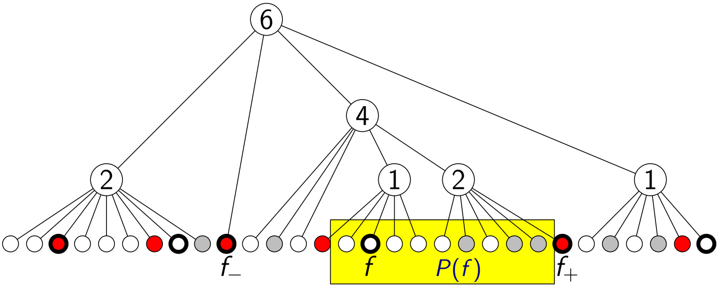
- With prob 1/8: $f_- \in A$, $f \notin A$, $f_+ \in A$.
  $\Rightarrow f_-, f_+ \in \mathrm{OPT}(A)$; $P(f) \in (f_- \ldots f_+)$; $P(f) \cap \mathrm{OPT} = \{f\}$.

Analysis sketch: Let f_-, f, f_+ consecutive in OPT.



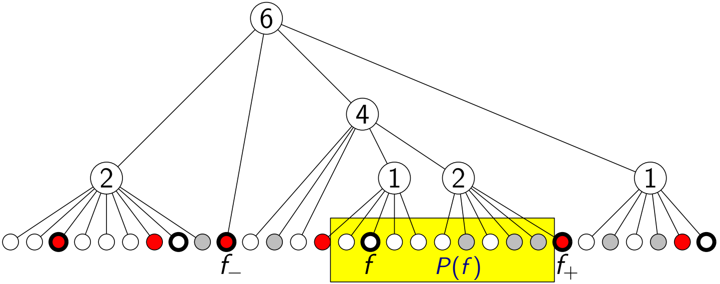
- With prob 1/8: $f_- \in A$, $f \notin A$, $f_+ \in A$.
  $\Rightarrow f_-, f_+ \in \mathrm{OPT}(A)$; $P(f) \in (f_- \dots f_+)$; $P(f) \cap \mathrm{OPT} = \{f\}$.
- With prob 1/2: $P(f) \in \mathcal{S}$ (good parity).

Analysis sketch: Let f_-, f, f_+ consecutive in OPT.



- With prob 1/8: $f_- \in A$, $f \notin A$, $f_+ \in A$.
  $\Rightarrow f_-, f_+ \in \mathrm{OPT}(A)$; $P(f) \in (f_- \dots f_+)$; $P(f) \cap \mathrm{OPT} = \{f\}$.
- With prob 1/2: $P(f) \in \mathcal{S}$ (good parity).
- With probability $1/e$, in $P(f)$ we recover weight $\geq w(f)$.

Analysis sketch: Let f_-, f, f_+ consecutive in OPT.



- With prob 1/8: $f_- \in A$, $f \notin A$, $f_+ \in A$.
  $\Rightarrow f_-, f_+ \in \mathrm{OPT}(A)$; $P(f) \in (f_- \ldots f_+)$; $P(f) \cap \mathrm{OPT} = \{f\}$.
- With prob 1/2: $P(f) \in \mathcal{S}$ (good parity).
- With probability $1/e$, in $P(f)$ we recover weight $\geq w(f)$.

$$\mathbb{E}[w(\mathrm{ALG})] \geq \mathbb{E}[w(\mathrm{OPT})]/(16e)$$

Matroid Secretary Problem: Outline

Free Order Model

We can choose the order in which elements reveal their weight.

Theorem [JSZ12]

There is a simple 9-competitive algorithm for any matroid in FOM.

Plan: Try to accept each $x \in \mathrm{OPT}$ with constant probability ($\geq 1/9$).

Free Order Model

We can choose the order in which elements reveal their weight.

Theorem [JSZ12]

There is a simple 9-competitive algorithm for any matroid in FOM.

Plan: Try to accept each $x \in \mathrm{OPT}$ with constant probability ($\geq 1/9$).

Good elements

An element e is good for $X \subseteq E \setminus \{e\}$ if $e \in \mathrm{OPT}(X \cup \{e\})$.

Elements in OPT are Good for any set!

First attempt

(Incorrect) Algorithm:

$\mathrm{ALG} \leftarrow \emptyset$.
Every element flips a coin partitioning *E* into *A* and *B*.
Observe *A*.

For every *e* of *B* in random order:
    If (*e* is good for *A*) and ($\mathrm{ALG} + e \in \mathcal{I}$)
        Then add *e* to ALG.

First attempt

(Incorrect) Algorithm:

$\mathrm{ALG} \leftarrow \emptyset$.
Every element flips a coin partitioning E into A and B.
Observe A.

For every e of B in random order:
 If (e is good for A) and ($\mathrm{ALG} + e \in \mathcal{I}$)
  Then add e to ALG.

Problem:

Might accept low-weight good elements that later block high-weight good elements.

First attempt

(Incorrect) Algorithm:

$\mathrm{ALG} \leftarrow \emptyset$.
Every element flips a coin partitioning E into A and B.
Observe A.

For every e of B in random order:
  If (e is good for A) and ($\mathrm{ALG} + e \in \mathcal{I}$)
    Then add e to ALG.

Problem:

Might accept low-weight good elements that later block high-weight good elements.

Idea:

Let $A_i = \{a_1, \ldots, a_i\}$ be the top i weights in A.

- Good elements for A_i in $B \cap \mathrm{span}(A_i)$ have weight at least $w(a_i)$.

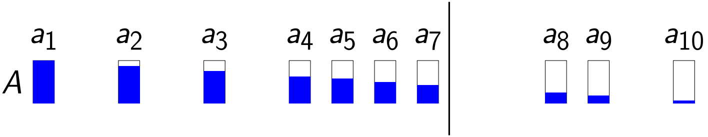
Algorithm

Simplifying assumption:
$\forall f: \ \Pr(f \in \mathrm{span}(A - f)) \approx 1.$

Online.

$\mathrm{ALG} \leftarrow \emptyset.$
Every element flips a coin partitioning E into A and B.
Observe and sort $A = \{a_1, \ldots, a_s\}$ by weight.

For $i = 1$ to s.
  For every $e \in (B \cap \mathrm{span}(A_i))$ not yet seen
    If $(\mathrm{ALG} + e \in \mathcal{I})$ and $(w(e) > w(a_i))$
      then add e to ALG.



Seen: Blue + Span(blue elements before line)

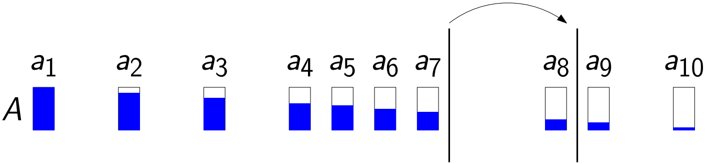
Algorithm

Simplifying assumption:
$\forall f:\ \Pr(f \in \mathrm{span}(A - f)) \approx 1.$

Online.

$\mathrm{ALG} \leftarrow \emptyset$.
Every element flips a coin partitioning E into A and B.
Observe and sort $A = \{a_1, \ldots, a_s\}$ by weight.

For $i = 1$ to s.
  For every $e \in (B \cap \mathrm{span}(A_i))$ not yet seen
    If $(\mathrm{ALG} + e \in \mathcal{I})$ and $(w(e) > w(a_i))$
      then add e to ALG.



Seen: Blue + Span(blue elements before line)

May accept a seen element from B heavier than a_8

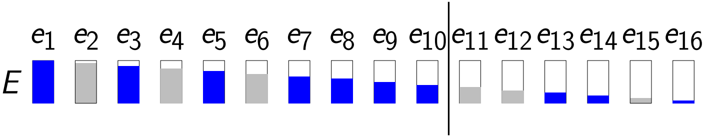
Algorithm

Simplifying assumption:
$\forall f: \Pr(f \in \mathrm{span}(A - f)) \approx 1.$

Offline simulation.

$\mathrm{ALG} \leftarrow \emptyset$.
Every element flips a coin partitioning E into A and B.
Sort $E = \{e_1, \ldots, e_n\}$ by weight. "See" A.

For $i = 1$ to n.
  For every $e \in (B \cap \mathrm{span}(A \cap E_i))$ not yet seen.
    If $(\mathrm{ALG} + e \in \mathcal{I})$ and $(w(e) > w(e_i))$
      then add e to ALG.



Seen: Blue + Span(blue elements before line)

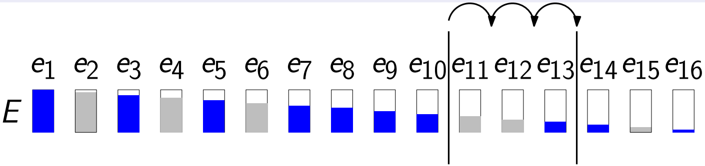
Algorithm

Simplifying assumption:
$\forall f: \Pr(f \in \mathrm{span}(A - f)) \approx 1.$

Offline simulation.

$\mathrm{ALG} \leftarrow \emptyset.$
Every element flips a coin partitioning E into A and B.
Sort $E = \{e_1, \ldots, e_n\}$ by weight. “See” A.

For $i = 1$ to n.
  For every $e \in (B \cap \mathrm{span}(A \cap E_i))$ not yet seen.
    If $(\mathrm{ALG} + e \in \mathcal{I})$ and $(w(e) > w(e_i))$
      then add e to ALG.



Seen: Blue + Span(blue elements before line)

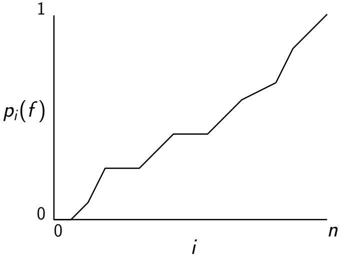
May accept a seen element from B heavier than e_{13}

Analysis (1): Let $f \in \mathrm{OPT}$.

Let $E = \{e_1, e_2, \ldots, e_n\}$ sorted by weights, and $E_i = \{e_1, \ldots, e_i\}$.

Let $p_i(f) = \Pr(f \in \mathrm{span}(A \cap E_i - f))$.

- $p_n(f) \approx 1$.
- $p_0(f) = 0$
- $p_i(f) \leq p_{i+1}(f)$.

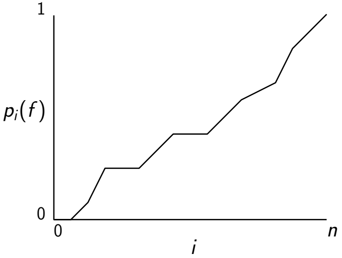


Analysis (1): Let $f \in \mathrm{OPT}$.

Let $E = \{e_1, e_2, \ldots, e_n\}$ sorted by weights, and $E_i = \{e_1, \ldots, e_i\}$.

Let $p_i(f) = \Pr(f \in \mathrm{span}(A \cap E_i - f))$.

- $p_n(f) \approx 1$.
- $p_0(f) = 0$
- $p_i(f) \leq p_{i+1}(f)$.



Can show that there is j such that $1/3 \leq p_j(f) \leq 2/3$.

Analysis (2)

Let $f \in \mathrm{OPT}$, and j s.t.

$$1/3 \leq \underbrace{\Pr(f \in \mathrm{span}(A \cap E_j - f))}_{p_j(f)} \leq 2/3.$$

Offline simulation.

$\mathrm{ALG} \leftarrow \emptyset$.

Every element flips a coin partitioning E into A and B.

Sort $E = \{e_1, \ldots, e_n\}$ by weight. "See" A.

For $i = 1$ to n.

For every $e \in (B \cap \mathrm{span}(A \cap E_i))$ not yet seen.

If $(\mathrm{ALG} + e \in \mathcal{I})$ and $(w(e) > w(e_i))$

then add e to ALG.

Analysis (2)

Let $f \in \mathrm{OPT}$, and j s.t.

$$1/3 \leq \underbrace{\Pr(f \in \mathrm{span}(A \cap E_j - f))}_{p_j(f)} \leq 2/3.$$

Offline simulation.

$\mathrm{ALG} \leftarrow \emptyset$.
Every element flips a coin partitioning E into A and B.
Sort $E = \{e_1, \ldots, e_n\}$ by weight. "See" A.

For $i = 1$ to n.
For every $e \in (B \cap \mathrm{span}(A \cap E_i))$ not yet seen.
If $(\mathrm{ALG} + e \in \mathcal{I})$ and $(w(e) > w(e_i))$
then add e to ALG.

Consider the events:

$\mathcal{E}_1$ $f \in B$.

$\mathcal{E}_2$ $f \in \mathrm{span}(A \cap E_j - f)$.

$\mathcal{E}_3$ $f \notin \mathrm{span}(B \cap E_j - f)$.

$\Rightarrow$

- f is not "sampled".
- f is "called" on some iteration $i \leq j$.
- f is not in the span of ALG when called.

Analysis (2)

Let $f \in \mathrm{OPT}$, and j s.t.

$$1/3 \leq \underbrace{\Pr(f \in \mathrm{span}(A \cap E_j - f))}_{p_j(f)} \leq 2/3.$$

Offline simulation.

$\mathrm{ALG} \leftarrow \emptyset$.

Every element flips a coin partitioning E into A and B.

Sort $E = \{e_1, \ldots, e_n\}$ by weight. "See" A.

For $i = 1$ to n.

For every $e \in (B \cap \mathrm{span}(A \cap E_i))$ not yet seen.

If $(\mathrm{ALG} + e \in \mathcal{I})$ and $(w(e) > w(e_i))$

then add e to ALG.

Consider the events:

$\mathcal{E}_1$ $f \in B$.

$\mathcal{E}_2$ $f \in \mathrm{span}(A \cap E_j - f)$.

$\mathcal{E}_3$ $f \notin \mathrm{span}(B \cap E_j - f)$.

$\Rightarrow$

$$\begin{aligned}
\Pr[\mathcal{E}_1 \cap \mathcal{E}_2 \cap \mathcal{E}_3] &= \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2 \cap \mathcal{E}_3] \\
&\underset{Pos.Corr.}{\geq} \Pr[\mathcal{E}_1] \cdot \Pr[\mathcal{E}_2] \cdot \Pr[\mathcal{E}_3] \\
&= (1/2) \cdot p_j(f) \cdot (1 - p_j(f)) \geq 1/9.
\end{aligned}$$

Conclusion

Our algorithm returns a set ALG such that

$$\forall f \in \mathrm{OPT},\ \Pr(f \in \mathrm{ALG}) \geq 1/9.$$

In particular,

$$\mathbb{E}[w(\mathrm{ALG})] \geq \frac{1}{9} w(\mathrm{OPT}).$$

9-competitive algorithm for Free Order Model!

Final Words

- Simple constant competitive algorithm for Laminar Matroids on Random Order Model.
- Constant competitive algorithm for Free Order Model.

Open

- Free order generalized secretary problem?
  (special cases, e.g. matroid intersections, etc.)
- Use ideas of free order to get constant in random order?
- Random Order for Gammoids and general matroids.