~~Graphical~~ Matroids Online Secretary

Momin

January 24, 2026

Contents

1 Bucketting Algorithm

The alg does solve the hat and recursive hat problem, but it fails for example in the "squished hat problem".
Is there are a way to modify is for this specific case?

Also, general matroids can not always be represented as graphical matroids with polynomial input. The proof comes from the fact that rank reduction in the graphical case is polynomial, but in the general case it is NP-hard.

2 Conservative - Liberal Framework

The General Framework:

1. Sample first $\left\lceil \frac{n}{k_1} \right\rceil$ elements
2. Conservatively select for the next $\left\lceil \frac{n}{k_2} \right\rceil$ elements
3. Liberally select for the next elements.

A simple idea

Sample the first $\frac{n}{2}$ elements, then both conservative and liberal phase lasts for $\frac{n}{4}$ elements.
Conservative: Only select the edge if the weight is the best you have seen so far.
Liberal: Greedily select the edge if the resulting graph is still acyclic.
This certainly works for the hat problem, and likely it's variants, but let's try to analyze generally.

Analysis

This really shouldn't work - too naive and doesnt use the graph structure.

Another idea

For the conservative phase, consider the cycles formed by adding e to the graph so far. If the weight is more than max-max, then accept it and kicked out the min from the sample.
The problem here is if the cycle doesnt form, then we might add low weight edges like in the case of the hat problem.

3 Immitation Strategy

Try to imitate the (offline) optimal solution.
Our options are Prim's, Kruskal's, and Boruvka's algorithms - the first one isn't really appicable since the edges arrive in random order.

Boruvka's Algorithm

Offline: For each component, select the maximum weight edge incident to it. Repeat until all components are connected.
Online: For each component, select the maximum weight edge (considering the sample and maybe vertices seen so far) incident to it

Analysis

Does this give a $\frac{1}{4}$-approximation?
Consider $e \in OPT(G)$ and let e connect the components C_1, C_2. Probability that second highest is in S is $\frac{1}{2}$, and probability that highest is not in S is $\frac{1}{2}$.
Would I get $\frac{1}{e}$ if I use $\frac{n}{e}$ samples? Seems too good to be true :/.

Kruskal's Algorithm

Offline: Keep selecting the heaviest edges that do not create a cycle until all vertices are connected.
Online: Maybe something similar to laminar where we keep track of top $|V|$ edges.
But the problem here is that for the alg to work, we need to go in the heaviest to lowest order - which greatly decreases the probability. Consider the 3-cycle of weights $1, 2, 100000$, then clearly order is very important.

4 Post-31st July

The problem with boruvka-online is that it fails the hat problem. The algorithm is too liberal when it ocmes to the nodes c_i.
We can try to make it a bit mroe conservative by requiring it to dominate both endpoints - this does solve the hat problem. However, it would perform poorly in for example an increasing straight line.
Maybe the problem is not about conservative-ness tho, because if we actually recieved the second-best, it would've been fine - or maybe that's what conservative is supposed to take into account.

Then, perhaps we can mash up the two. So let's say $\frac{1}{3}$ sample, $\frac{1}{3}$ conservative, and $\frac{1}{3}$ liberal - does that cover everything?

Recursive Hat Problem

Likely doesnt work because for each node we want second highest to be in sample and highest to be outside sample. For a few nodes it's fine, but it eventually decreases in probability.

If we allow fractional, maybe we can take $\frac{1}{2}$ in the consevative stage.

5 Using Fractional

Naive

Algorithm:

1. Sample the first $\frac{n}{2}$ elements.
2. At arrival, if $e \notin OPT_t(G)$ reject it.
3. If it's edition to T creates no cycle, accept it and add to T' with fraction $\frac{1}{2}$.
4. Otherwise accept it with fraction $\frac{1}{2^{k+1}}$ where k is the minimum k value of edges going out of $u, v \in T'$.

But we can create a problematic construction by allowing many many cycles that algo will pick wuth some constant probability.

Modifying Online-Boruvka

Algorithm:

1. Sample the first $\frac{n}{2}$ elements.
2. At arrival, if e is not the best (so far) edge coming out of either components, reject it.
3. Otherwise, consider the n cycles formed by adding e and let k_e be the maximum k value of the edges in the cycle.
4. Accept e with k value $k_e + 1$ (i.e. fractionally select it with weight $\frac{1}{2^{k_e+1}}$)

The algorithm may just work because even through the fractions are exponentially decaying, but so is the probability of that happening - we may just get constant factor in expectation.

Analysis

Let $e \in OPT(G)$ be an edge connecting components C_1 and C_2. Clearly, we are sure to accept e, but we want to find the expected fraction.
Let k_e be the fraction of e accepted by the algorithm and let $d = deg(C_1) + deg(C_2)$. We can write it as follows:

$$E[k_e] = \sum_{k=0}^{d-1} \frac{1}{2^{k+1}} \cdot P(k_e = k) = \sum_{k=0}^{\infty} \frac{1}{2^{k+1}} \cdot P(k_e = k)$$

We wish to acquire a lower bound on $P(k_e = k)$ for $k \in \mathcal{N} \cup \{0\}$.
Let e_i represent the i-th best edge incident to C_1 or C_2 and $a < b$ mean that a arrived before b.

$$P[k_e = 0] \geq P[e_2 \in S, e_1 \notin S] = \frac{1}{2^2}$$

$$P[k_e = 1] \geq P[e_3 \in S, e_2 \notin S, e_1 \notin S, e_2 < e_1] = \frac{1}{2^4}$$

$$P[k_e = k] \geq \begin{cases} \frac{1}{2^{2k+2}} & \text{if } k < d \\ 0 & \text{otherwise} \end{cases}$$

Putting this together, we get

$$E[k_e] \geq \frac{1}{8} \sum_{k=0}^{d-1} \frac{1}{2^{3k}} \geq^{\mathrm{d} \geq 1} \frac{1}{8}$$

Additionally, if d is big enough, we have the bound $\frac{1}{8} \cdot \frac{1}{1-\frac{1}{8}} - \epsilon = \frac{1}{7} - \epsilon$.
Since the analysis here is very loose, this works for the above naive algorithm as well. I wonder if this algorithm can tighten the bound - would be useful to consider the hat problem.
For example, in the hat problem, this algorithms solves it in factor 4.

Modifying Step 3

Let $e = (u, v)$. Then $k = \min\{\text{edges from } u \text{ or } v : k_{edge}\}$.

Correctness

At the accepetance of any edge e, either it connects two unconnected components or connects one connected component. In the first case, there's so cycle formed. In the second case, for the purposes of counting the fractions, it is equivalent to removing the edge e and adding the fraction to the edge e' that is "paying for it".

Problem

Consider the graph with edge (u, v) of small weight. But it is the only edge connecting the dense left and right half. In this case, the probability of it being selected is arbitrarily low.
This however, might be utility competitive.

Modifying Step 3 Again

To solve this problem, we can consider the idea "if the edge is the best so far that connects C_1 to C_2 (directly)".
But there should be some troublesome counterexample for this idea.
Consider graph of nodes a, b, c. Consider the case the a, c has many parallel nodes of low weight - so the correct answer is to go through $a - b - c$.

Modifying Step 3 yet Again

We accept the edge e if it is the best edge so far connecting **any** componenets C_1, C_2 where $u \in C_1, v \in C_2$.

6 Post 6th Aug

We had the "check all cuts, the max one pays" doesnt work for parallel edges - indepedences is violated.
Perhaps we can look for another way for an edge paying for other edges.

Naive Greedy (with better payment)

Algorithm

1. sample $\frac{n}{2}$ elements
2. Initialize G_1, G_2 as empty graphs.
3. At arrival of edge e at time T, if $e \notin OPT_t(G)$ reject it.
4. Otherwise, do

   If e does not create a cycle in G_2, add it with fraction $\frac{1}{2} \in G_1$ and $\in G_2$.

   Otherwise, let e' be the edge $\in G_1$ with the minimum fraction k. Then, add e with fraction $\frac{1-k}{2}$ to G_2 and update k to $\frac{1+k}{2}$.

The problem here is that subsets might not obey rank constraints. So for example, there is a big clique and many isolated edges. It can happen, with non zero probability, that the algorithm keeps on selecting edges in the clique that are being paid for by the isolated edges.

Naive Greedy (with modified payment)

Algorithm

Input: Edge stream of size n
Output: Set T of selected edges with fractions
Sample the first $\frac{n}{2}$ elements;
Initialize $G' \leftarrow \emptyset$;
Initialize $T \leftarrow \emptyset$ (set of pairs $(e, f(e))$);
Initialize $f(e) = g(e) = 0$ for all $e \in E$;
foreach *edge e arriving at time t* **do**
  if $e \notin OPT_t(G)$ **then**
    Reject e;
  end
  else
    Let $\{C_1, \ldots, C_k\}$ be the set of cycles formed by adding e to G';
    if $k = 0$ **then**
      Add e to G' with $f(e) = g(e) = \frac{1}{2}$;
      Add $(e, \frac{1}{2})$ to T;
    else
      For each C_i, let $m(C_i)$ be the minimum $f(e')$ for $e' \in C_i$;
      Let $f(e'') = \max(m(C_i))$;
      Let $g(e) = \frac{1-f(e'')}{2}$ and add $(e, g(e))$ to T;
      Update $f(e'') \leftarrow \frac{1+f(e'')}{2}$;
    end
  end
end

Algorithm 1: Naive Greedy (with modified payment)

Feesability

Note that

$$\sum_{e \in T} g(e) = \sum_{e \in G'} f(e) \leq \sum_{e \in G'} \leq |V| - 1$$

Therefore, for the subset E (the entire set), feesability is satisfied.
To show for any subset $E' \subset E$, the same argument works.

The problem

Even if this is feesable (which I have not shown yet), the ratio can be quite bad.
Consider the hat problem where light edges have many parallel edges.

General Matroids

Input: Element stream of size n
Output: Set T of selected elements with fractions
Sample the first $\frac{n}{2}$ elements;
Initialize $G' \leftarrow \emptyset$;
Initialize $T \leftarrow \emptyset$ (set of pairs $(e, f(e))$);
foreach *element e arriving at time t* **do**
  if $e \notin OPT_t(G)$ **then**
    Reject e;
  end
  else
    Let $f(e) = \frac{max(k)}{2}$ where $max(k) \in (0, 1]$ is the maximum fraction of e we can add with independence being preserved;
    Add $(e, f(e))$ to T;
  end
end

Algorithm 2: General Naive Greedy (with modified payment)

Feesability is clear, but the ratio can be quite bad - consider the hat problem where light edges have many parallel edges.

7 Post 12 August

Trying to look at specific classes that has not been studied yet.

Bond/Cographic Matroids

The bond matroid $B(G)$ of a graph G. The circuits of $B(G)$ are the minimal edge cuts, also known as the bonds of G. These are minimal collections of the edges of G which, when removed from G, increase the number of connected components.
The rank $r(X)$ is equal to the maximal number of edges deletion of which do not increase the number of connected components of the spanning subgraph which edges from X.
So we need to choose edges to cut such that the graph is still connected. The total weight of the edges we cut should be minimal.
This is also called co-graphic matroid since it is the dual of graphic matroid.
The algorithms for graphic matroid won't exactly work perfectly here. Graphic matroid algorithms guarantees that we do not have a cycle - but we may have a forests. To be feesable in bond matroid, we need to ensure that the graph is connected.
However, note that in this we want to minimize the weight of the set of elements selected - but in matroid secretary problem, we need to maximize. So instead, consider the dual of minimum spanning tree.

New Problem (?)

Input : The graph (G, E) where $e \in E$ arrives in online manner.
Feesability : Select a set of edges $T \subset E$ such that $(G, E \setminus T)$ is connected.
Objective : Maximize the weight of T.
This is equivalent to minimizing the weight of the edges that are not selected.
Again, the algorithm for graphic matroid does not directly transfer due to mismatch in feesability.
Also, this is still a matroid secretary problem, the feesability ensures matroids properties are satisfied.

Naive Greedy

Just keep on accepting edges that create a cycle.
The ratio here can be arbritrarily bad - consider the 2 nodes graph with many parallel edges. Most edges are high weight, but one is low weight. So here we should cut every edge till $\frac{1}{e}$, and then not cut the edge which is the lowest so far.

Greedy

Input: Edge stream of size n
Output: Set T of selected edges
Select the first $\frac{n}{e}$ edges as long as independence is preserved;
foreach *edge e arriving at time t* **do**
  if $e \notin OPT_t(G)$ **then**
    Reject e;
  end
  else
    Add e to T;
  end
end

Algorithm 3: Greedy for Bond Matroid

Similarly,
Input: Edge stream of size n
Output: Set T of selected edges
Select the first $\frac{n}{e}$ edges as long as independence is preserved;
foreach *edge e arriving at time t* **do**
  if e is the best edge so far connecting two components (any C_1, C_2 with $u \in C_1, v \in C_2$), select e if indepedence is preserved;
end

Algorithm 4: Greedy for Bond Matroid

Alg3,4 don't work on a simple cycle with many light weights and 1 heavy weight.

Using Graphic Matroid Algorithm

Input: Edge stream of size n
Output: Set T of selected edges
Fix a decreasing function $f(x)$;
foreach *edge e arriving at time e* **do**
  Feed $f(e)$ to the algorithm for graphic matroid secretary problem;
  If the edge is selected, don't select it;
  Otherwise select it if $s < t$ and indepedence is satisfied;
end

Algorithm 5: Using Graphic Matroid Algorithm for Bond Matroid

If we have a graphical matroid algorithm such that

$$P[e \notin T | e \notin S, e \notin OPT] \geq p$$

, then we have a ps-competitive algorithm for co-graphical matroids.
Considering the 4-competitive alg for graphic matroids, we have

$$\begin{aligned} P[e \notin T | e \notin S, e \notin OPT] &= P[e \notin OPT_t \cup (e \in OPT_t \cap G) | e \notin S, e \notin OPT] \\ &\leq P[e \notin OPT_t | e \notin S, e \notin OPT] + P[e \in OPT_t \cap G | e \notin S, e \notin OPT] \end{aligned}$$

Perhaps it's helpful to write out the alg.

Input: Edge stream of size n
Output: Set T of selected edges
ALG and A are the currently selected independent set and the
 orientation of its associated forest.;
Fix a decreasing function $f(x)$;
$ALG \leftarrow \emptyset, A \leftarrow \emptyset, T \leftarrow \emptyset, s = Bin(n, p)$;
for $i = s + 1$ **to** n **do**
  Let $a_i = (u_i, v_i)$ be the canonical orientation of edge$(r_i, OPT_g(R_i))$;
  if $f(r_i) \in OPT_g(R_i)$ *and* $deg_A(u_i)^- = 0 = deg_A(v_i)^-$ **then**
    $ALG \leftarrow ALG + f(r_i), A \leftarrow A + a_i$;
  end
  else
    if $T + r_i$ *is independent* **then**
      $T \leftarrow T + r_i$;
    end
  end
end
return T;

Algorithm 6: immitating 4-competitive alg

Here $OPT_g(R_i)$ refers to the optimal solution for the graphical matroid secretary problem on the graph R_i formed by the edges selected so far. By abuse of notation, $f(r_i)$ means edge r_i with weight $f(w(r_i))$.

Maybe if we consider the $2e-$ competitive alg that just randomizes the orientation, we can achieve an easier bound.

Input: Edge stream of size n
Output: Set T of selected edges
Fix an ordering $v_1, \ldots, v_{|V|}$ and with equal probability oriented the edges from lower to higher or higher to lower. ;
$T \leftarrow \emptyset, s = Bin(n, p)$;
for $i = s + 1$ **to** n **do**
  if r_i *is not the worst (least weight) edge leaving* v_i **then**
    if $T + e$ *is indepdent* **then**
      $T \leftarrow T + r_i$;
    end
  end
end
return T;

Algorithm 7: immitating $2e$-competitive alg

Let us first consider the easier case of online secretary problem. First ignoring the indepdence constraint we have

$$P[x_i \notin T | x \notin OPT, x \notin S] = \frac{1}{n-1}[\frac{1}{2} + \frac{2}{3} + \ldots + \frac{s+1}{s+2}]$$

$$= \frac{1}{n-1}\sum_{i=1}^{s+1}\frac{i}{i+1} \approx \frac{1}{n-1}\int_1^{s+1}\frac{k}{k+1}dk = \frac{1}{n-1}(s+1-\log(s+1)-1+\log(2))$$

Considering $s = \frac{n}{e}$, we have

$$\geq \frac{1}{e} - \frac{\log(\frac{\frac{n}{e}+1}{2})}{n} \to \frac{1}{e}$$

Wait, if we consider x_i to be the i^{th} element, then it should be

$$P[x_i \notin T | x \notin OPT, x \notin S] = \frac{1}{(n-1)\cdot(i-1)}[1 + 2 + \ldots + i - 1]$$

$$= \frac{1}{(n-1)\cdot(i-1)} \cdot \frac{(i-1)i}{2} = \frac{i}{2(n-1)}$$

So, the probability of any suboptimal x not being selected is

$$P[x \notin T | x \notin OPT, x \notin S] = \frac{1}{2(n-1)}\sum_{i=1}^{s+1} i = \frac{1}{2(n-1)} \cdot \frac{(s+1)(s+2)}{2}$$

Now, we can consider the indepdence constraint. Oh, although we already have elements in the sample, so they won't be choosen anyways...

$$P[x \notin T | x \notin OPT, x \notin S] \approx \frac{1}{e}$$

Actually feesability is the biggest issue in immitation. Consider a long cycle with only one good edge. planned

General Observations

1. If $e \in OPT(G_t)$, then $e \in OPT(G_k)$ for $k \geq t$.
2. It may be that $e \notin OPT(G_t)$ but $e \in OPT(G_k)$ for some $k > t$.
3. Consider e connecting C_1, C_2 that can not be selected due to independence. Then, we know there will no longer by any edge connecting the two componenets.

Considering-cycle-algorithm

Input: Edge stream of size n
Output: Set T of selected edges
Sample the first $\frac{n}{e}$ elements;
foreach *edge e arriving at time t* **do**
  Consider the minimal set $E' \subset E_t \setminus T$ such that $T + e + E'$ is not indepdent, but $T + E'$ is. **if** $w(e) > w(e') \quad \forall e' \in E'_t$ **then**
    Accept e
  end
  else
    Reject e
  end
end

Reject parallel edges maybe? This would be good but we can't remove parallel edges that we have not seen ...

8 Known Matroid Model

Assuming we know the structure of the matroid before-hand, can we achieve something better?

Bond Matroid

Cycle-greedy

Input: Edge stream of size n
Output: Set T of selected edges
Sample the first $\frac{n}{e}$ elements;
foreach *edge e arriving at time t* **do**
  Consider $C_1, \ldots, C_k$ cycles that e is part of and let $f(e) = \frac{w(e)}{k}$;
  Select e if $f(e) > f(e')$ for all $e' \in C_e = \bigcup_{i=1}^{k} C_i$.
end

9 Post 21 Aug

Can we make a graphic matroid that ensures a spanning tree?
Can we assign "cycle" and then just attempt to remove the best edge from each cycle?
Maybe try forbidden sets technique.
Maybe back to graph matroids for a bit.
We can try forcing grpahic matroid to produce a spanning tree - but it might mess up the analysis.

Looking back

Consider the graphic setting, what if we don't add an edge if there exists a cycle such that it is worst(or equal) to any edge in it.
The problem here occurs for the arrival. It may be the even after sampling n/e, a cycle didnt have any edges in sample. So perhaps run the secretary alg on each cycle?

Considering the same alg again

Accept an edge if $s < t$ and $f(e) \notin ALG_g$ and you can take it.
Here isnt it the case that

$Pr[e \in ALG | e \in OPT, e \notin S] = Pr[e \notin ALG_g, e \text{ is not the last one connecting any unconnected component}|...$

Oh, the problem is that e may be in many connected components

Using fractional for Bond

10 Post 27th August

Either soft constraints or look ahead.
First let's try to define a model.
Let's say we can look ahead k elements and we sample s elements. So if we run the normal secretary algorithm, the analysis may look something like

$$Pr[e_1 \in ALG] = Pr[(e_1 \notin S \cap e_2 \in S) \cup (e_1 \notin S \cap e_2 < e_1 + k)] + ...$$

$$Pr[(e_1 \notin S \cap e_2 \in S) \cup (e_1 \notin S \cap e2 \notin S \cap e_2 < e_1 + k)] = \frac{s}{n} \cdot (1 - \frac{s}{n}) + (1 - \frac{s}{n}) \cdot \frac{k}{n-s}$$

If trying to focus on i^{th} item then

$$Pr[e \in ALG] = \frac{1}{n} \sum_{i=s+1}^{n} Pr[e \in ALG | e \text{ arrives at time } i]$$

$$\frac{1}{n}(\underbrace{1+\ldots+1}_{k \text{ times}}+.)$$

$$. = \frac{s}{n}(\frac{n-s-2}{n} + \frac{n-s-2}{n} \cdot \frac{n-s-3}{n})$$

ok it seems to be a geometric series.

$$. = \frac{s}{n} \sum_{k=0}^{s} \prod_{i=0}^{k} (1 - \frac{s+2+i}{n})$$

11 Lookahead

We may conjecture that if a algorithm achieves ratio r, then look-ahead l gives us $1 - \frac{1-l}{r}$.

12 Look-back

The problem was considered before in the paper : A Secretary Problem with a Sliding Window for Recalling Candidates (https://arxiv.org/pdf/1508.07931). The setting is nearly the same as look-ahead with the following 2 exceptions :

1. It is a look-back and not look-ahead : We can accept any of the last K elements instead of knowing the values of future elements.
2. They consider the discrete setting while look-ahead considered the continuous setting.

The first point of difference does not really matter, analysis is the same. Let me try to give the same result for the discrete setting :

Analysis

Assume $s + k < n$, otherwise we can just solve optimally.
Case 1 : $i < s + k + 1$

$$P(s,k) = \sum_{i=s+1}^{s+k} P(\text{ applicant } i \text{ is selected } \cap \text{ applicant } i \text{ is the best }) = \frac{k}{n}$$

Case 2 : $i \geq s+k+1$

$$\begin{aligned}
P(s,k) &= \sum_{i=s+k+1}^{n} P(\text{ applicant } i \text{ is selected } \cap \text{ applicant } i \text{ is the best }) \\
&= \sum_{i=s+k+1}^{n} P(\text{ applicant } i \text{ is selected } \mid \text{ applicant } i \text{ is the best }) \cdot P(\text{ applicant } i \text{ is the best }) \\
&= \sum_{i=s+k+1}^{n} P\left(\begin{array}{c}\text{the best of the first } i-k-1 \text{ applicants is in the first } s \text{ applicants} \\ \mid \text{ applicant } i \text{ is the best}\end{array}\right) \cdot \frac{1}{n} \\
&= \left[\sum_{i=s+k+1}^{n} \frac{s}{i-k-1}\right] \cdot \frac{1}{n} \\
&= \frac{s}{n} \sum_{i=s+k+1}^{n} \frac{1}{i-k-1} \\
&\approx \frac{s}{n} \int_{s}^{n-k-1} \frac{1}{y} dy = -\frac{s}{n} \ln(\frac{s}{n-k-1}).
\end{aligned}$$

Combining the two cases, we get

$$P(s,k) = \frac{k}{n} + \frac{s}{n} \ln(\frac{n-k-1}{s})$$

Letting $s' = \lim_{n\to\infty} \frac{s}{n}, k' = \lim_{n\to\infty} \frac{k}{n}$, we get the same equation as before.

Trying a tighter analysis

The above conditions are not tight, so we'd like to get something tighter.
Consider Case 2 again :

$$\begin{aligned}
&P[i \text{ is selected } \mid i \text{ is best}] = P[\forall j \in [s+1, i-k-1] \text{ is rejected} \mid i \text{ is best}] \\
&= \frac{s+k}{s+k+1} \cdot P[\forall j \in [s+2, i-k-1] \text{ is rejected} \mid i \text{ is best}, s+1 \text{ is reject}] \\
&= (\frac{s+k}{s+k+1})^2 \cdot P[\forall j \in [s+3, i-k-1] \text{ is rejected} \mid i \text{ is best}, [s+1, s+2] \text{ is reject}] \\
&= (1 - \frac{1}{s+k+1})^{i-k-s-1}
\end{aligned}$$

Therefore we have

$$\begin{aligned}
P[\text{Win}] &= \frac{k}{n} + \frac{1}{n} \sum_{i=s+k+1}^{n} (1 - \frac{1}{s+k+1})^{i-k-s-1} \\
&= \frac{k}{n} + \frac{1}{n} \cdot \frac{1 - (\frac{s+k}{s+k+1})^{n-s-k-1}}{\frac{1}{s+k+1}} \approx \frac{s+2k+1}{n}
\end{aligned}$$

where the last approximation is for large n (assuming $s' + k' < c < 1$).
We should also be able to bound as such :

$$P[i \text{ is selected } \mid i \text{ is best}] = P[\forall j \in [s+1, i-k-1] \text{ is rejected} \mid i \text{ is best}]$$

$$= \frac{s+k}{s+k+1} \cdot P[\forall j \in [s+2, i-k-1] \text{ is rejected} \mid i \text{ is best}, s+1 \text{ is reject}]$$

$$\leq (\frac{s+k}{s+k+1})\frac{s+k-1}{s+k} \cdot P[\forall j \in [s{+}3, i{-}k{-}1] \text{ is rejected} \mid i \text{ is best}, [s{+}1, s{+}2] \text{ is reject}]$$

$$\leq (\frac{s+k}{s+k+1})(\frac{s+k-1}{s+k})^2 \cdot P[\forall j \in [s{+}4, i{-}k{-}1] \text{ is rejected} \mid i \text{ is best}, [s{+}1, s{+}2, s{+}3] \text{ is reject}]$$

Therefore we get

$$P[\text{ win }] \leq \frac{k}{n} + \frac{1}{n} \sum_{i=s+k+1}^{n} (\frac{s+k-1}{s+k})^{i-k-s-1}$$

$$= \frac{k}{n} + \frac{1}{n} \cdot \frac{1 - (\frac{s+k-1}{s+k})^{n-s-k-1}}{1 - \frac{s+k-1}{s+k}} \approx \frac{s+2k}{n}$$

Trying again to get a tighter bound

Consider case 2 again. Let r_i be the index of rank i element - we wish to find the probability of selecting r_1.

$$E_1 = r_2 \in S, r_1 \notin S$$

$$E_2 = r_3 \in S, r_2 \notin S, r_2 \in [i-k, n]$$

$$E_3 = r_4 \in S, r_3, r_2 \notin S, I(r_3) - I(r_2) \leq k \text{ or } r_3 \in [i-k, n]$$

The events are already disjoint.

$$Pr[E_1] = \frac{s(1-s)}{n^2}$$

$$Pr[E_2] = \frac{s(1-s)^2}{n^3} \cdot k$$

$$Pr[E_3] = \frac{s(1-s)^3}{n^4} \cdot ()$$

$$Pr[\text{ win }] = Pr[E_1 \cup E_2 \cup \ldots]$$

I give up on this

$$\sum_{i=0}^{\infty} \left(\frac{s}{t+k}\right)^{y^{i+1}} \left(\frac{2k}{t+k}\right)^{\frac{y(y^i-1)}{y-1}}$$

$$= \left(\frac{s}{t+k}\right)^{y} T\left(\left(\frac{s}{t+k}\right)^{y-1} \frac{2k}{t+k}\right)$$

, where $T(z) = 1 + z^y T(z^y)$ for $|z| < 1$, with $T(0) = 1$.
optimality of threshold,
divide into δ buckets.
try to apply the analysis on the bucketing alg.

There is a clear $O(k)$ algorithm by just select the opt at some time $(i, i+k)$.

13 Optimality of thresholding for sliding window

We wish to show that the optimal algorithm is a thresholding algorithm.
It is sufficient to show that $\exists d^*$ such that for $d \neq d^*$, $P[Win]$ is lesser. In particular, we want d^* to be the smallest d such that $P_A[Win] > P_R[Win]$.
I think it should suffice to just show monotonicity since

$$P_A[Win_n] = 1 \quad , \quad P_A[Win_1] = \frac{1}{n} \quad , \quad P_R[Win_n] = 0$$

Monotoncity of acceptance is clear since

$$P_A[Win_i] = \frac{i}{n}$$

. Monotonicity of rejection is not so clear.

$$P_R[Win_i] = P[Win_k]$$

where $P[Win_k] = max(P_A[Win_A[k], Win_R[k]])$ where k is the next candidate after i. But perhaps it is fruitfull to even derive the exact probabilities.
Let's try to do backwards induction.

$$P_R[Win_n] = 0 \implies P[n] = 1$$

$$P_R[Win_{n-1}] = \frac{1}{n} P_A[Win_n] = \frac{1}{n} \implies P[n-1] = \max(\frac{1}{n}, 1-\frac{1}{n}) = \frac{n-1}{n} \quad \forall n \geq 2$$

$$P_R[Win_{n-2}] = \frac{1}{n-1} P[n-1] + (\frac{n-2}{n-1}) \frac{1}{n-1} P[n] = \frac{1}{n} + \frac{n-2}{(n-1)^2}$$

$$
\begin{aligned}
P_R[Win_{n-3}] &= \frac{1}{n-2}P[n-2] + (\frac{n-3}{n-2})(\frac{1}{n-2})\frac{1}{n-2}P[n-1] + (\frac{n-3}{n-2})^2(\frac{1}{n-2})P[n] \\
&= \frac{1}{n-2}(\frac{1}{n} + \frac{n-2}{(n-1)^2}) + (\frac{n-3}{n-2})(\frac{1}{n-2})\frac{1}{n-1} + (\frac{n-3}{n-2})^2(\frac{1}{n-2}) \\
&= \frac{1}{n} + \frac{1}{(n-1)^2} + \frac{n-3}{(n-2)(n-1)} + \frac{(n-3)^2}{(n-2)^2}\cdot\frac{1}{n} \\
&= \frac{1}{n} + \frac{1}{(n-1)^2} + \frac{n-3}{(n-2)(n-1)} + \frac{(n-3)^2}{(n-2)^2}\cdot\frac{1}{n}
\end{aligned}
$$

In general for $k \geq d^*$, we have

$$
\begin{aligned}
P_R[Win_k] &= \frac{1}{k+1}\sum_{i=k+1}^{n}(\frac{k}{k+1})^{i-k-1}P[i] =_{P_A \geq P_B} \frac{1}{k+1}\sum_{i=k+1}^{n}(\frac{k}{k+1})^{i-k-1}\frac{i}{n} \\
&\approx \frac{1}{k+1}\int_{k+1}^{n}(\frac{k}{k+1})^{x-k-1}\frac{x}{n}dx = \frac{1}{n(k+1)}\int_{1}^{n-k}(\frac{k}{k+1})^{y-1}(y+k)dy \\
&= \frac{1}{n(k+1)}\left[k\int_{1}^{n-k}(\frac{k}{k+1})^{y-1}dy + \int_{1}^{n-k}y(\frac{k}{k+1})^{y-1}dy\right] \\
&= \frac{1}{n(k+1)}\left[k\frac{1-(\frac{k}{k+1})^{n-k}}{1-\frac{k}{k+1}} + \left(\frac{k}{k+1}\right)^{n-k}\frac{n-k}{\ln(\frac{k}{k+1})} + \frac{\frac{k}{k+1}}{\ln(\frac{k}{k+1})^2}\right] \\
&= \frac{1}{n}\left[k + \frac{1}{\ln(\frac{k}{k+1})} + \left(\frac{k}{k+1}\right)^{n-k}\left(\frac{n-k}{(k+1)\ln(\frac{k}{k+1})} - k\right)\right] \\
&\approx \frac{1}{n}\left[k + \frac{1}{\ln(\frac{k}{k+1})} - \frac{(n-k)k}{(k+1)\ln(\frac{k+1}{k})}\right]
\end{aligned}
$$

Note that this is a decreasing function and it must also cross $P_A[Win_k]$ somewhere.

Using continuous model

Consider time t.

$$P_A(t) = t$$

$$P_R(t) = \int_t^1 \frac{x-t}{x}P(x)dx$$

. Whatever $P(x)$ is, $P_A(t)$ is increasing and $P_R(t)$ is decreasing with $P_A(1) = 1, P_A(0) = \frac{1}{n}, P_R(1) = 0$. So there must be a crossing point.

Trying this with sliding window

Let

$P_A[i]$ = Probability of winning upon accepting candidate i

$P_R[i]$ = Probability of winning upon rejecting candidate i

$P[i]$ = Probability of winning upon reaching candidate $i = \max(P_A[i], P_R[i])$

. We wish to show that $\exists d^*$ such that

$$P_A[i] \geq P_R[i] \quad \forall i \geq d^*$$

$$P_A[i] < P_R[i] \quad \forall i < d^*$$

. For $P_A[i]$, it is clear that

$$P_A[i] = \frac{i+k}{n}$$

, so we have $P_A[1] = \frac{1+k}{n}$, and $P_A[j] = 1$ for $j > n - k$.
To get $P_R[i]$, we build recursively from the end.
Base case : $j > n - k$
It is clear that $P_R[j] = 0 \leq 1 = P_A[j]$.
Inductive case : Assume $P_R[j] \leq P_A[j]$ for some j.
In particular, this means that $P[j] = P_A[j]$.

$$P_R[j-1] = \sum_{i=j+k-1}^{n} \frac{j+k-1}{i} \cdot \frac{1}{i} P[i] = \sum_{i=j+k-1}^{n} \frac{j+k-1}{i} \cdot \frac{1}{j+k-1} \cdot \frac{i+k}{n}$$

$$= \sum_{i=j+k-1}^{n} \frac{i+k}{i \cdot n} = \frac{1}{n} \sum_{i=j+k-1}^{n} 1 + \frac{k}{i}$$

, we can see that $P_R[k]$ is decreasing in k while $P_A[k]$ is increasing.

Using continuous model

Let

$P_A(i)$ = Probability of winning upon accepting candidate at time i

$P_R(i)$ = Probability of winning upon rejecting candidate at time i

$P(i)$ = Probability of winning upon reaching candidate at time $i = \max(P_A(i), P_R(i))$

Consider time t.

$$P_A(t) = t + \delta$$

$$P_R(t) = \int_{t+\delta}^{1} \frac{x - t - \delta}{x} P(x) dx$$

. Whatever $P(x)$ is, $P_A(t)$ is strictly increasing and $P_R(t)$ is strictly decreasing with $P_A(1) = 1, P_A(0) = \frac{1}{n}, P_R(1) = 0$. So there must be a crossing point. This implies that $\exists d^*$ such that

$$P_A[i] \geq P_R[i] \quad \forall i \geq d^*$$

$$P_A[i] < P_R[i] \quad \forall i < d^*$$

which is sufficient to show optimality of thresholding.

At k vs $k+1$ for the ith term, we have

$$P[i] \cdot \frac{k}{k+1}^{i-k-1} \geq P[i] \cdot \frac{k+1}{k+2}^{i-k-2}$$

which simplifies to

$$(\frac{k}{k+1})^c \geq (\frac{k+1}{k+2})^{c+1}$$

Using monotonicity

We wish to show that

$$P_R[i-\delta] \geq P_R[i]$$

. We proceed by induction.

$$\begin{aligned}
P_R[i-\delta] &= \frac{1}{i+\delta}P[i] + \frac{1}{i+\delta+1}P[i+1] + \ldots + \frac{1}{i+2\delta-2}P[i+\delta] + (1-\frac{1}{i+\delta})(\frac{1}{i+2\delta-1})P[i+\delta] + \ldots \\
&\geq \frac{1}{i+\delta}P_R[i] + \frac{1}{i+\delta+1}P[i+1] + \ldots + \frac{1}{i+2\delta-2}P[i+\delta] + (1-\frac{1}{i+\delta})(\frac{1}{i+2\delta-1})P[i+\delta] + \ldots \\
&= \frac{1}{i+\delta+1}P[i+1] + \ldots + \frac{1}{i+2\delta-2}P[i+\delta] + \big((1-\frac{1}{i+\delta})(\frac{1}{i+2\delta-1}) + \frac{1}{(i+\delta)(i+2\delta)}\big)P[i+\delta] +
\end{aligned}$$

If we cheat a litle bit,

$$\begin{aligned}
&> \big((1-\frac{1}{i+\delta})(\frac{1}{i+2\delta}) + \frac{1}{(i+\delta)(i+2\delta)}\big)P[i+\delta] + \ldots \\
&= (\frac{1}{i+2\delta})P[i+\delta] + \ldots \\
&= P_R[i]
\end{aligned}$$

Questioning the logic

I do not think this logic is correct. If we are considering "Pr that the algoirthm wins upon rejecting" instead of the "max probability that the algorithm wins on rejecting", then we can construct a stupid alg such that P_R is not decreasing. For example, in some interval $[i, i+\delta]$, if we see a candidate but reject, then our alg throws a tantrum and does not choose anything upto another interval $[n-\delta, n]$.

14 Post 3 Oct

simulating for reductions

$$c = 1 - \frac{1+\frac{1}{\delta}}{n}$$

15 Reductions

Assume we have look ahead of k elements.

15.1 Single Simulation Reduction

For each $k+1$ elements, we require k elements to be stacked up. So, we need $s = k \cdot \frac{n}{k+1}$ stack $\implies$

$$c = 1 - \frac{s}{n} = 1 - \frac{k}{k+1}$$

15.2 Multiple Simulation Reduction

Atleast for the 1-uniform matroid, we can do better - perhaps bound both ways. We always first sample first $k-1$ elements. After that, for each index i, we perform a query by placed element i and $i-k$.
Similarly, we can query for without lookback.
Let us give the reduction ratio

$$r = (1 - \frac{k}{n})$$

.
If we don't distrubt the distribution, then we need to multiply by extra probabilities of elements not being repeated selected.
If we disturb the distribution, we can make it for example ascending, but then the algorithm's guarantee doesn't hold.11

15.3 Guiding the algorithm

Assume we have a c-competitive algo for the lookahead version. If we can show the reduction that we have at least kc algorithm for the normal version, it follows that

$$c \leq \frac{1}{e} \cdot \frac{1}{k}$$

So, we can try get a large k.
Assume it is dynkin.
We can guide the algorithm to the highest element.
We can keep the stack constant and make sure that the algorithm does not stop before.

$$k = \frac{\Delta - 1}{n}$$

15.4 Multi-Multiple simultaions

We could use the following fact : If alg guarantees c-competitiveness, then running it on all permutations should gives us that best element x is select in at least $c\%$ of them.

15.5 Δ-Flexible elements

We can also consider relaxation where a Δ portion of the elements are essentially always available options/backups/offline.
The input is thus

$$I = I_\Delta \cup I_O = \{x_1, \ldots, x_\Delta\} \cup \{x_{\Delta+1}, \ldots, x_n\}$$

Let s be the stopping point of the sample, then

$$\begin{aligned} Pr[Win] &= \frac{\Delta}{n} + \sum_{i=\Delta+s+1}^{n} Pr[\text{Best of } [1, i-1] \in [1, \Delta+s]] \cdot \frac{1}{n} \\ &= \frac{\Delta}{n} + \frac{1}{n} \sum_{i=\Delta+s+1}^{m} \frac{\Delta+s}{i-1} \\ &= \frac{\Delta}{n} + \frac{\Delta+s}{n} \sum_{i=\Delta+s+1}^{m} \frac{1}{i-1}. \end{aligned}$$

Using the continuous model, using $y = \frac{\Delta}{n}, x = \frac{s}{n}$, we have

$$\begin{aligned} Pr[Win] &= y + (x+y) \int_{x+y}^{1} \frac{1}{z} dz \\ &= y + (x+y) \ln(\frac{1}{x+y}) \\ &= y - (x+y) \ln(x+y) \end{aligned}$$

Taking the differential wrt x, we have

$$\frac{dPr[Win]}{dx} = -\ln(x+y) - 1$$

Setting to 0, we have $x + y = \frac{1}{e} \implies x = \frac{1}{e} - y$. Thus the optimal stopping point is

$$s = n(\frac{1}{e} - \frac{\Delta}{n}) = \frac{n}{e} - \Delta$$

with optimal probability of

$$Pr[Win] = \boxed{\frac{\Delta}{n} + \frac{1}{e}}$$

, that is if $\Delta \leq \frac{n}{e}$.
If $\Delta > \frac{n}{e}$, then we set $s = 0$ and get

$$Pr[Win] = \frac{\Delta}{n} + \frac{\Delta}{n} \ln(\frac{n}{\Delta}) = \boxed{\frac{1}{n}(\Delta - \Delta \ln(\Delta))}$$

which is increasing in Δ.

Analysing

The above should only hold for $0 \leq x = \frac{1}{e} - y \implies \Delta \leq \frac{n}{e}$.

15.6 Batches?

16 Possible Directions

- Relax Matroid constraint - allow Δ violations.
- Hardness results
- Remove Matroid constraint - arbritrary is too hard, but have knapsack etc.
- Waitlist - allow Δ elements to be deferred to the end.