Free-Order Matroid Secretary Problem

Nalin Ranjan, Simon Park, and Alex Valtchanov

COS 521 Final Project

Abstract

In the classical secretary problem, an online algorithm is given a stream of elements, each with a non-negative value, and must decide whether to accept the element and terminate the algorithm, or to reject it and reveal the next element. The goal of the algorithm is to maximize the probability of choosing the element with the maximum value. The problem has also generalized to settings where the algorithm has to accept an independent set in a matroid. The goal is to maximize the expected sum of values of elements in the accepted independent set, or maximize the probability of choosing each element in the offline optimal independent set.

In this project, we study the free-order matroid problem, where an algorithm is given both the structure of a matroid and the power to determine an order in which to reveal values of elements. We begin by establishing the negative result that no algorithm for the free-order matroid problem can be strongly $(e - \Omega(1))$-competitive. We then turn our attention to the strongly 4-competitive algorithm proposed by Jailet, Soto, and Zenklusen [8], and show that the competitive analysis presented is tight in the general case. Motivated by this, we propose an algorithm for the free-order matroid secretary problem that we hypothesize is strongly e-competitive for any matroid.

We prove that our proposed algorithm indeed is strongly e-competitive on uniform matroids and on a certain graphic matroid first proposed in [3] that is known to be an adversarial input to many algorithms. We also discuss why analytically analyzing our algorithm is difficult. In addition, we provide numerical analysis on randomly generated matroids of various types that supports our hypothesis. On all matroids that we analyzed, we observe that the numerical evidence is consistent with the hypothesis that our algorithm is e-approximate.

We also provide evidence that a particular generalization of the "virtual algorithm" described in [3] may be strongly e-competitive when the rank of a matroid is significantly smaller than the universe size. However, we also show experimental evidence that the algorithm fails to be e-competitive when the rank is comparable to the universe size (e.g., when it is even some small constant of the universe size).

1 Background

1.1 Classical Secretary Problem

The secretary problem's origin has been disputed but goes back at least fifty years. It is an important motivating problem in optimal stopping theory that has seen many variants — each with their own applications, e.g. in mechanism design — posed.

Ordinal Version. An online algorithm is given a set $\mathcal{U}$. Each element $u \in \mathcal{U}$ has a value $v(u)$ that is *unknown* to the algorithm at the beginning of the problem, and the goal of the algorithm is to select one $u \in U$ that maximizes $v(u)$. The algorithm is given a stream of values $v(u)$, and each time it receives a new value, it needs to irrevocably decide whether or not to choose that u. If it does, the algorithm will terminate with u as the final answer; if not, it will wait for the next input to be provided. No element can be queried twice, and passed elements may not be retroactively accepted.

Let $u^* \in U$ have the maximum value $v(u^*)$. An online algorithm for the traditional secretary problem obtains a competitive ratio of c if $\Pr[u = u^*] \geq 1/c$, where u is the element chosen by the algorithm, and the probability is taken over a uniform random ordering of elements presented to the algorithm.

Cardinal Version. Same problem setting as above. Let $u^* \in U$ have the maximum value $v(u^*)$. An online algorithm for the traditional secretary problem obtains a competitive ratio of c if $\mathbb{E}[v(u)] \geq v(u^*)/c$, where u is

the element chosen by the algorithm, and the expectation is taken over a uniform random ordering of elements presented to the algorithm.

1.2 Matroid Secretary Problems

Definition 1.1. A tuple $(\mathcal{U}, \mathcal{I})$ is a *matroid* if $\mathcal{I} \subseteq \mathcal{P}(\mathcal{U})$ and the following conditions hold

1. $\emptyset \in \mathcal{I}$.
2. If $A \in \mathcal{I}$ and $A' \subseteq A$, then $A' \in \mathcal{I}$.
3. If $A, B \in \mathcal{I}$ and $|A| > |B|$, then there exists a $y \in A \setminus B$ such that $B \cup \{y\} \in \mathcal{I}$.

Usually, $\mathcal{I}$ is referred to as the set of *independent* subsets of the *universe* (or "ground set") $\mathcal{U}$.

There are at least three notable versions of the matroid secretary problem. Each roughly asks an online algorithm to choose an independent set of elements of maximum value. They are as follows:

Oblivious Matroid Secretary Problem. This is the hardest variant of the problem. There is a matroid $M = (\mathcal{U}, \mathcal{I})$ and an online algorithm is presented a stream of elements $u \in \mathcal{U}$ (potentially in adversarial order). Each element is associated with a non-negative value $v(u)$ and the goal of the algorithm is to select an independent set $S \in \mathcal{I}$ that maximizes its total value $v(S) := \sum_{u \in S} v(u)$. When an algorithm is presented an element, it learns its value; however, it must irrevocably decide whether or not it accepts the element in its answer. In other words, passed elements may not be retroactively accepted, nor may accepted elements be retroactively removed from the algorithm's answer. In this variation, it will be guaranteed that any element presented will maintain independence (but because of this, not all elements in $\mathcal{U}$ may be presented before the algorithm terminates). We refer to this variant as "oblivious" because an online algorithm is totally oblivious to the structure of the matroid.

Structure-Conscious Matroid Secretary problem. This variant is a relaxation of oblivious matroid secretary problem. The problem setting is the same as the oblivious case, but this time the online algorithm is given the underlying matroid $M = (\mathcal{U}, \mathcal{I})$. The stream of elements $u \in \mathcal{U}$ may still be presented in adversarial order.

Free-Order Matroid Secretary Problem. This is a relaxation of the structure-conscious matroid secretary problem. Again, the problem setup is the same, but in addition to the underlying matroid $M = (\mathcal{U}, \mathcal{I})$, the algorithm is also allowed to choose *any* order in which to reveal elements' values. In particular, the decision on which value the algorithm reveals next may be based off of any values of previously seen elements. As usual, upon discovering the value of some element, it must irrevocably decide whether or not to include it as part of its final answer.

Notions of Competitiveness. Let $S^*(M)$ be a max-value independent set in $M = (\mathcal{U}, \mathcal{I})$ and A be the independent set of elements accepted by an online algorithm for any of the above matroid secretary problems. There are two versions of metrics to evaluate the competitive ratio of such algorithm:

1. An algorithm is *weakly c-competitive* if $\mathbb{E}[v(A)] \geq v(S^*(M))/c$, regardless of the choice of M and values assigned to each element of the matroid. Importantly, the expectation is taken solely over the *algorithm's own randomness*.
2. An algorithm is *strongly c-competitive* if $\Pr[u \in A] \geq 1/c$ for any $u \in S^*(M)$, regardless of the choice of M and values assigned to each element of the matroid. Similarly, the probability is evaluated over the *algorithm's own randomness*.

From these definitions, it is clear that an algorithm that is strongly c-competitive is also weakly c-competitve, but not necessarily the other way around.

1.3 Related Work

e-approximation Algorithm for Classical Secretary Problem. The optimal solution to the classical secretary problem is folklore: when we have a stream of n secretaries, we first reject the first $\lfloor pn \rfloor$ candidates,

where $p \in [0,1]$, but we store the maximum value of those candidates. Then we accept the first candidate that has a greater value than the threshold given by the sample we accessed. Then the probability of accepting the best applicant is

$$\sum_{j=\lfloor pn \rfloor+1}^{n} \frac{1}{n} \cdot \frac{\lfloor pn \rfloor}{j-1} = \frac{\lfloor pn \rfloor}{n} \sum_{j=\lfloor pn \rfloor+1}^{n} \frac{n}{j-1} \cdot \frac{1}{n} \overset{n\to\infty}{\longrightarrow} p \int_{pn}^{n} \frac{dt}{t} = -p \log(p)$$

It is clear that this probability will obtain a maximum of $\frac{1}{e}$ when $p = \frac{1}{e}$. Actually, it is known that this optimal stopping algorithm obtains a better approximation ratio than all other algorithms. At a high level, this is shown by first proving that any algorithm is dominated by a stopping time algorithm, and then showing that the optimal number of samples for a stopping algorithm is $\lfloor n/e \rfloor$ [6].

State-of-the-art for the Matroid Secretary Problem. There is no known constant-competitive (weakly or strongly) algorithm for the oblivious or structure-conscious matroid secretary problems (see the next section regarding the free-order model). However, many non-constant approximation ratios are known for even the oblivious variant. A seminal result of Babaioff, Immorlica, Kempe, and Kleinberg [3] showcased a weakly $O(\log \rho)$-competitive algorithm — where ρ is the rank of the matroid — with a moderate constant factor (~ 32). This result was eventually improved to a weakly $O(\sqrt{\log \rho})$-competitive result, at the cost of an extremely large constant factor ($\geq 2^{64}$) [4]. With the additional weaker assumption that the number of elements in the matroid is known, the current state of the art [7] is a weakly $O(\log \log \rho)$-competitive algorithm with a high constant factor (around 2500). For general matroids, the best strongly competitive algorithm is known to have competitive ratio $O(\log \rho)$ [12].

Significant progress has also been made on specific matroid classes. Uniform matroids — where any subset of $[n]$ of size $\leq k = \rho$ is independent — admit a weakly $1 - O(1/\sqrt{k})$-competitive algorithm that is known to be asymptotically optimal in k [9]. Uniform matroids are also known to admit a strongly $1 - O(\sqrt{\log k/k})$-competitive algorithm [12]. There are also simpler algorithms that are strongly e-competitive for all values of k, though these may not be asymptotically optimal [3]. Graphic matroids — where elements are edges of a graph and independent sets are acyclic subgraphs — are known to admit a 4-competitive algorithm [12], which was an improvement upon the previously known $2e$-competitive algorithm [11]. Laminar matroids — generated from a laminar family of sets and constraints — admit a strongly $3\sqrt{3}$-competitive procedure [12]. Transversal matroids — where elements are nodes on one side of a bipartite graph and independent sets are sets of nodes that can be perfectly matched — admit a strongly 8-competitive approximation [11].

4-approximation Algorithm for Free-Order Matroid Secretary Problem [JSZ]. In [8], a 4-approximation for the free-order matroid secretary problem is proposed. For completeness, we reproduce it below:

1. For each element $u \in \mathcal{U}$, flip a fair coin to determine whether or not to include it in a sample set S. Request and reject each element in S, observing each of their values.
2. Find the max-value basis of S (subject to $\mathcal{I}$). Let this basis be $\{X_1, \cdots, X_m\}$, where $m = \text{rank}(S)$ and $v(X_1) > v(X_2) > \cdots > v(X_m)$. Also denote $P = \mathcal{U} - S$.
3. Initialize $A = \emptyset$, which will contain the algorithm's answer, and a counter $i = 1$.
4. Let $\Delta_i = \text{span}(X_1, \cdots, X_i) - \text{span}(X_1, \cdots, X_{i-1})$. For all the elements $u \in P \cap \Delta_i$ enumerated in *any order*, add u to A if and only if $A \cup \{u\} \in \mathcal{I}$ and $v(u) > v(X_i)$. **Note:** By convention, $\Delta_{m+1} = \mathcal{U} - \text{span}(X_1, \cdots, X_m)$ and $v(X_{m+1}) = 0$.
5. If $i = m + 1$, then terminate; otherwise, increment i and go repeat step 4.

The original version of the paper was able to prove that the algorithm is 9-approximate. A simple analysis was given in [1] to show that the algorithm is 4-approximate.

1.4 Motivation for Research

The strongest negative result to our knowledge is the following:

Theorem 1.4.1. *No algorithm for the free-order matroid secretary problem can be strongly* $(e - \Omega(1))$*-competitive.*

Proof. Consider the free-order matroid secretary problem on the matroid $M = ([n], \{\{1\}, \cdots, \{n\}\})$. Clearly, the max-value basis of M is just the max-value item out of $[n]$. By definition, an algorithm that is strongly $(e - \Omega(1))$-competitive must be able to select the max-value element i^* (regardless of which one it is) with probability at least $1/(e - \Omega(1))$. But if this is the case, then the probability of finding the max-value element if given a *uniform random permutation* of weights will also be at least $1/(e - \Omega(1))$, which by [6] cannot be the case.[1] □ □

Remark: It is believed that no algorithm for the free-order matroid secretary problem can be weakly $(e-\Omega(1))$-competitive either. Professor Weinberg believes that similar techniques from reference [5] show that being weakly c-competitive for a rank 1 matroid M implies being strongly c-competitive on M. If this is true, the result above suffices to show that no algorithm for the free-order matroid secretary problem can be weakly $(e - \Omega(1))$-competitive either.

Let's label the 4-approximate algorithm provided in [8] as JSZ (for Jaillet, Soto, and Zenklusen). It turns out that the analysis of JSZ is tight if the enumeration order in step 4 is adversarially chosen.

Theorem 1.4.2. *For every $\epsilon > 0$, there exists a matroid M, a set of values $v(\cdot)$, and an enumeration order of elements in each iteration of step 4 such that*

1. *The offline optimum achieves a payoff (total value) of 1.*
2. *With this enumeration order, JSZ achieves an expected payoff of $\frac{1}{4} + \frac{\epsilon}{4}$.*

Proof. The counterexample, once again, is the 1-uniform matroid with n elements. Note that regardless of the sample S, JSZ goes through exactly one iteration of step 4, where it considers all elements that are not present in the sample. Without loss of generality, assume n is the max-value element and that $n-1$ is the element of second largest value. Assume $v(n) = 1$, and $v(n-1) = \epsilon$, where $\epsilon > 0$ can be chosen to be arbitrarily small. Clearly, n cannot be in S if the algorithm is to select n; this happens with probability $1/2$. Now if $n-1$ is also not in S, then JSZ will always choose $n-1$ as long as it enumerates $n-1$ before n, since in this case the single max-value basis element of S will have weight zero. Thus, the only case where it is guaranteed that n will be picked by JSZ is when $n \notin S$ and $n - 1 \in S$. This happens with probability $1/4$. Thus, the expected payoff of JSZ with this enumeration order is $\frac{1}{4} + \frac{\epsilon}{4}$. □ □

Note, however, that the most natural enumeration in step 4 — a random order — immediately improves the payoff of JSZ in the counterexample given in Theorem 1.4.2. Let us choose a total ordering of $[n]$, σ, so that $v(\sigma(1)) > v(\sigma(2)) > \cdots > v(\sigma(n))$. Of course, the true max-value basis will be the singleton set $\{\sigma(1)\}$. Now if $\sigma(i)$ is the element of maximal value in the sample S, there is a $\frac{1}{2(i-1)}$ chance that JSZ correctly chooses $\sigma(1)$ as its answer. (The $1/2$ is the probability that $\sigma(i)$ is not in S, and the $1/(i-1)$ is the probability that $\sigma(1)$ is the first element to appear in a uniform random permutation of the elements in $\{j \mid v(j) > v(\sigma(i))\}$.) So the probability that $\sigma(1)$ is chosen is

$$\frac{1}{2}\sum_{i=2}^{n+1} \frac{1}{(i-1)\cdot 2^{\min(i-1,n-1)}} > \frac{1}{2}\ln 2$$

with equality as $n \to \infty$. In fact, if the sample probability is changed from $1/2$ to $1/e$, then we see that this probability, equal to $p\sum_{i=2}^{n+1}(1-p)^{\min(i-1,n-1)}/(i-1) \geq -p\ln p$ is maximized and tends to $1/e$ as $n \to \infty$. This leads to the following conjecture:

Conjecture 1.4.3. *The algorithm JSZ(e) — which samples each element of $\mathcal{U}$ into S with probability $1/e$ and always enumerates candidates in each iteration of step 4 in uniform random order — but otherwise carries out the same steps as JSZ, is **strongly** e-competitive.*

[1] It is important to note that this probability is taken jointly over the algorithm's own probability, as well as that arising from the random ordering of elements by weight.

2 Positive Results for JSZ(e)

As noted before, Conjecture 1.4.3 holds for the case of 1-uniform matroids. It also holds in some other special cases. For the results that follow, we will be referring to steps of JSZ(e) exactly as we do in the previous section: for example, when we refer to "step 4" of JSZ(e), we are referring to the loop that iterates through the $\{\Delta_i\}$ one-by-one, trying to find elements to add to its answer.

2.1 Uniform Matroids

Theorem 2.1.1. *JSZ(e) is strongly $c(n,k)$-competitive whenever M is a k-uniform matroid of n elements, where*

$$\frac{1}{c(n,k)} = \sum_{j=0}^{k-1} \binom{n-1}{j} p^j (1-p)^{n-j} \min\left(1,\ \frac{k}{n-j}\right) + \sum_{i=k+1}^{n} \binom{i-2}{k-1} p^k (1-p)^{i-k} \min\left(1,\ \frac{k}{i-k}\right)$$

Proof. We will consider two cases: when the sample has less than k elements, and when it has at least k elements. First, when the sample has $j < k$ elements, then the max-weight basis of S is itself. Also, $P \cap \Delta_i = \emptyset$ for any $i \leq j$. The first time we consider any element in P is at the $(j+1)$-th iteration, where all elements of P are considered with equal probability. So for a specific max-weight element u^* in the universe to be accepted by the algorithm, it cannot be in the sample, and it should be chosen in the last iteration. The probability that $|S| = j$ given $u^* \notin S$ is $\binom{n-1}{j} p^j (1-p)^{n-j-1}$, and the probability that u^* is chosen given all the preceding events is $\min\left(1,\ \frac{k}{n-j}\right)$. Therefore,

$$\Pr[u^* \text{ accepted, sample has } j \text{ elements}] = \binom{n-1}{j} p^j (1-p)^{n-j} \min\left(1,\ \frac{k}{n-j}\right)$$

The second case is when the sample has at least k elements. Then the max-weight basis of S consists of the top k elements. Then the first time we consdier any element in P is at the k-th iteration, where all elements of P with value greater than the top k-th element of S are considered. So for a specific max-weight element u^* to be accepted, it cannot be in the sample, and it should be chosen in this iteration. The probability that the top i-th element in the universe is the top k-th element, given $u^* \notin S$ is $\binom{i-2}{k-1} p^k (1-p)^{i-k-1}$, and the probability that u^* is chosen given all the preceding events is $\min\left(1,\ \frac{k}{i-k}\right)$. If we sum the probability over all $i > k$, then

$$\Pr[u^* \text{ accepted, sample has at least } k \text{ elements}] = \sum_{i=k+1}^{n} \binom{i-2}{k-1} p^k (1-p)^{i-k} \min\left(1,\ \frac{k}{i-k}\right)$$

Adding the probabilities for the two cases yields

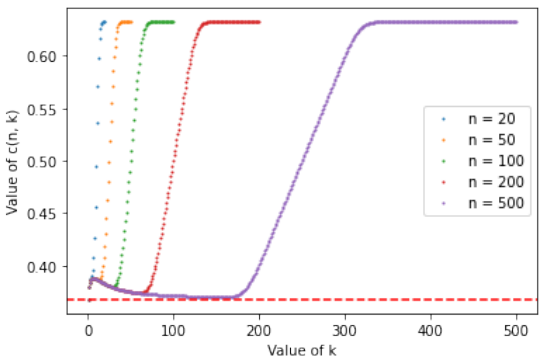
$$\Pr[u^* \text{ accepted}] = \sum_{j=0}^{k-1} \binom{n-1}{j} p^j (1-p)^{n-j} \min\left(1,\ \frac{k}{n-j}\right) + \sum_{i=k+1}^{n} \binom{i-2}{k-1} p^k (1-p)^{i-k} \min\left(1,\ \frac{k}{i-k}\right)$$

completing the proof. □

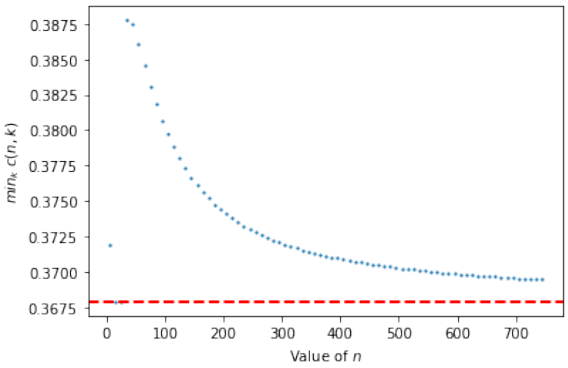
Numerical Behavior of $c(n,k)$. For a small k with respect to n, the first term dominates the sum, whereas for a large k, the second term takes over. In any case, it seems to be that $1/e \leq c(n,k) \leq 1 - 1/e$. We provide numerical evidence in Figure 1.

2.2 Theoretical Evaluation of JSZ(e) on Particular Graphic Matroids

Consider the graph $G = (V, E)$ where $V = \{u_1, u_2\} \cup \{v_1, v_2, \cdots, v_n\}$ and $E = E_0 \cup E_1 \cup E_2$ where $E_0 = \{e_0 = (u_1, u_2)\}$, E_1 is the set of all edges of the form $e_{1j} = (u_1, v_j)$ and E_2 is the set of all edges of the form $e_{2j} = (u_2, v_j)$. The edge e_0 is given infinite weight, and all edges $e_{ij} \in E_i$ has weight i for any $j \in [n]$. This graph and similar variants are known to be a bad input for some simple heuristics, e.g. for a simple greedy algorithm that adds an edge if and only if it would be in the max-value basis of all the elements seen so far [3].



(a) Plot of $c(n,k)$ for various values of n.



(b) Plot of $f(n) = \min_k c(n,k)$ for $5 \le n \le 750$.

Figure 1: Numerical simulations to suggest that $\lim_{n\to\infty} \min_k c(n,k) \ge 1/e$. The dashed line is plotted at $y = \frac{1}{e}$.

Theorem 2.2.1. *Every element in the max-weight basis of the matroid is chosen by JSZ(e) with probability $0.43 > 1/e$ for this specific input. In particular, the max-weight element is chosen with probability $1 - \frac{q}{e}$, as $n \to \infty$.*

Proof. First, the optimal offline solution is clearly the set $E_0 \cup E_2$, since adding any edge $e_{1j} \in E_1$ to this subset will create the cycle $u_1 \to u_2 \to v_j \to u_1$.

Now we wish to prove that the online algorithm will select each optimal element with probability of at least $\frac{1}{e}$. First consider the element e_0. For it to be chosen by the online algorithm, it cannot have been chosen in the sample. This happens with probability $1 - \frac{1}{e}$. Now further assume that e_0 was not in the sample. Let

$$J_1 = \{j \in [n] : (u_1, v_j) \in E_1 \cap S\} = \{j_{1,1}, j_{1,2}, \cdots, j_{1,k_1}\}$$
$$J_2 = \{j \in [n] : (u_2, v_j) \in E_2 \cap S\} = \{j_{2,1}, j_{2,2}, \cdots, j_{2,k_2}\}$$

Consider constructing the max-weight basis of the sample set S with the greedy algorithm. Let $S_1, S_2, \cdots, S_k$ be the sequence of independent sets that the algorithm obtains during computing the max-weight basis. Notice that the greedy algorithm will first add all elements of $E_2 \cap S$. Notice that the span of these elements, intersected with P, is always empty. That is

$$\text{span}(S_i) \cap P = \emptyset \qquad i = 1, 2, \cdots, k_1$$

So there are no elements in P that we will consider. After adding all elements of $E_2 \cap S$, the algorithm will start adding elements of $E_1 \cap S$ to the max-weight basis. Let k' be the first index such that $S_k \setminus S_{k-1} = \{e_{1j}\}$ for some $j \in J_1 \cap J_2$. For any $i < k'$, we see that

$$\text{span}(S_i) \cap P = \emptyset \qquad i < k'$$

But once we add e_{1j}, the span consists of the edge e_0 and all the "partner edges" to all edges added so far to the max-weight basis of the sample. That is,

$$\text{span}(S_{k'}) \cap P = \{e_0\} \cup \{e_{1j} : j \in J_2\} \cup \{e_{2j} : j \in J_1, e_{1j} \in S_{k'-1}\}$$

Notice that even if we accept all edges in this span, we still have an acyclic graph. Therefore, e_0 will be accepted into the output. The only case this does not happen with certainty is when $J_1 \cap J_2 = \emptyset$. This happens with probability $(1 - \frac{1}{e^2})^n$ because for each of v_j, we can't have both e_{1j} and e_{2j} in the sample. To sum up, the probability that e_0 is included in the output is

$$\Pr[e_0 \text{ chosen by the algorithm}] = \left(1 - \frac{1}{e}\right) \cdot \left(1 - \left(1 - \frac{1}{e^2}\right)^n\right) \overset{n\to\infty}{\longrightarrow} 1 - \frac{1}{e} \simeq 0.632 > \frac{1}{e}$$

Indeed, the expression on the leftmost side is greater than $\frac{1}{e}$ for any $n > 6$.

Now consider any edge e_{2j}. First we need e_{2j} to not be in the sample. Then we condition on whether e_{1j} was in the sample. If e_{1j} was in the sample, then there is no way e_{2j} will create a cycle with remaining edges in P, so it will be accepted in the output with certainty. If e_{1j} was not in the sample, then the two edges e_{1j} and e_{2j} will both be considered in the last step of the algorithm. e_{2j} being considered before e_{1j} is a sufficient condition for adding e_{2j} to the output.[2] In equation, the probability is bounded below by

$$\Pr[e_{2j} \text{ chosen by the algorithm}] \geq \left(1 - \frac{1}{e}\right) \cdot \left(\frac{1}{e} + \frac{1}{2}\left(1 - \frac{1}{e}\right)\right) \simeq 0.432 > \frac{1}{e}$$

completing the proof. □

2.3 Discussion of Difficulty in Analyzing JSZ(e)

There are at least two reasons as to why reasoning about JSZ(e) may be difficult. *First, the asymmetric sampling probability prevents certain arguments of symmetry from holding.* For example, in [1] the authors use a nice observation to argue that a certain event (that would imply that an element is not spanned when the algorithm isn't encountered and therefore would be added) occurs with probability 1/2. This observation depended on the fact that the sample S and unsampled elements P are indistinguishable if the sampling probability is 1/2. One future avenue of research could be to simply study the original JSZ algorithm that samples with probability 1/2, but still randomizes the order in which certain elements are considered in the loop of step 4.

The second reason is that it is both 1) difficult to predict the max-value basis of a random sample of a general matroid, and 2) difficult to control the impact of the algorithm's own randomness. When the given matroid is rank 1, both of these tasks are extremely straightforward, and hence a tractable probability calculation can be obtained. Why? The max-value basis was simply the maximum element of the sample, and the algorithm's randomness did not have any impact on which elements we considered in the only iteration of step four of JSZ(e).

In general, however, if two samples S_1 and S_2 differ by some s elements, the max-value bases of the two samples could differ by up to s elements. This could change the value of any of the Δ_i (recall $\Delta_i = \text{span}\{X_1, \cdots, X_i\} - \text{span}\{X_1, \cdots, X_{i-1}\}$) quite drastically (even if $s = 1$), making it difficult to compare the performance of the algorithm if S_1 were the sample versus if S_2 were the sample. In particular, our rank-1 analysis of the chance that a max-value basis element $y \in \Delta_i$ gets included in the particular iteration its considered depended on it coming relatively early in the random ordering of Δ_i the algorithm enumerates. This would rely upon the sample basis element X_i to serve as a "cutoff" that would prevent elements of low value in Δ_i from being included. However, it is not straightforward in general to evaluate what would happen if one of the elements $y' \in \Delta_i$ was actually in the sample, because this could change the max-value basis of the sample. That could in turn result in different values for many Δ_i in the modified run. If it were in fact true that the sequence of Δ_i considered would remain the same in response to the sample being perturbed by one element (or just differ in a local sense that doesn't totally change the entire order in which the algorithm considers elements), then the analysis would become significantly easier.

3 Numerical Evaluation of JSZ(e)

In this section, we implement and empirically evaluate JSZ(e) on various matroids. We provide the code used for all implementations and experiments at `github.com/nalin1729/COS521_FinalProject`.

3.1 Experiment Setup

For a particular matroid $M = (\mathcal{U}, \mathcal{I})$ with universe size $|\mathcal{U}| = n$, we sample a uniformly random valuation function $v : \mathcal{U} \mapsto [0, 1]$ and run JSZ(e) t times. We record the ratio between the number of times each element of $\mathcal{U}$ is present in the algorithm's output and the number of trials, notated $f_t(u)$. For large t, $f_t(u)$ approximates the probability that the algorithm outputs an independent set containing u. The worst case performance of the algorithm is therefore captured by the minimum of f_t over all elements in the optimal offline solution $S^*(M)$.

The experiments are designed to verify Conjecture 1.4.3 (recall that this hypothesis that JSZ(e) is *strongly* e-competitive as $n \to \infty$). Since JSZ(e) is *absolute weight-agnostic* — in the sense that only the relative ordering

[2] Again, this is because any cycle that e_{2j} is a part of necessarily includes e_{1j}.

of the elements by weight changes the behavior of the algorithm in expectation — and we are measuring the probabilities of inclusion for the max-weight basis elements[3], the actual weights that we assign to elements of the matroid do not matter. Thus, the uniform random valuations come without loss of generality.

If Conjecture 1.4.3 holds, then we expect that for any $u^* \in \mathcal{U}$ that is in the max-value basis $S^*(M)$,

$$\lim_{t\to\infty} f_t(u^*) = \mathbb{P}\{u^* \text{ included in JSZ's answer}\} \geq 1/e$$

for any matroid M and in particular,

$$\min_{u^*\in S^*(M)} \lim_{t\to\infty} f_t(u^*) \geq 1/e$$

We test this hypothesis for various classes of matroids, as described in the next section.

3.2 Simulated Matroids

We implement JSZ(e) in `Python` and utilize a Matroid class from the `SageMath` toolkit. The specific classes of Matroids we assess JSZ(e) on are as follows:

- **Uniform Matroids.** We vary the rank uniformly between 1 and $|\mathcal{U}|$.
- **Graphic Matroids.** For a particular universe size n, we generate a random graph by sampling the number of vertices $m \in [\lceil 1+\sqrt{1+8n}/2\rceil, n]$ and including n edges at all locations with equal probability.[4] Independence of a set is then implemented by looking for cycles.
- **Transversal Matroids.** We generate random bipartite graphs (L, R, E) according to three parameters: $|L|$, $|R|$, and p, where p is the probability that any given edge from L to R is included. Independence of set $S \subseteq L$ is implemented by calculating a maximum matching from S to R (using max-flow) and checking if the size of this matching is equal to $|S|$.
- **Linear Matroids over Finite Fields.** We generate n random vectors over $\mathbb{F}_p^n$ (for some prime p). We add a mechanism to introduce dependence amongst vectors[5] as follows: with some probability q a newly generated vector will be independent of all previously generated vectors; otherwise, it will be set to a (random) nonzero linear combination of a uniform random subset of the previously generated vectors.
- **General Matroids**. To generate random general matroids, we follow an expensive algorithm proposed by Knuth [10] that is tractable for small universe sizes ($n \leq 15$). With this methodology, every possible matroid with $|\mathcal{U}| = n$ can be generated, but each such matroid isn't necessarily sampled with the same chance.

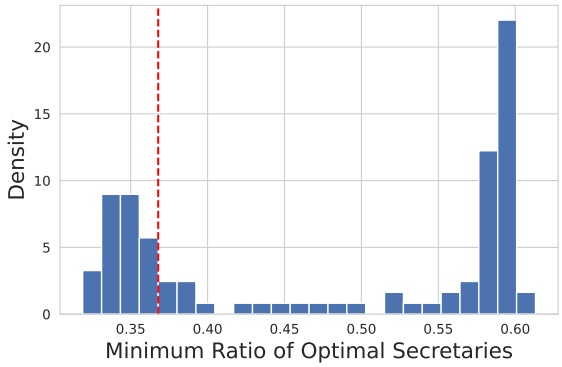
3.3 Statistical Analysis of Results

Figure 2 plots the distribution of $X_t(M) := \min_{u\in S^*(M)} f_t(u)$ with $t = 1000$ trials for various matroids M. One will find in Figure 2 that by and large, most elements in the max-weight basis are *included* in the algorithm's answer at least $1/e$ of the time. We will explain observed deviations are not anomalies; in fact, they are to be expected.

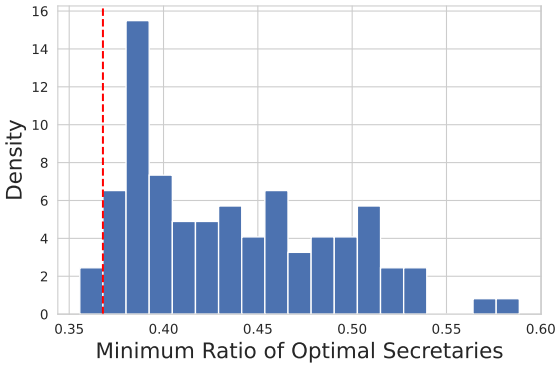
Figure 2a shows around 30% of the distribution of $X_t(M)$ to lie below the threshold of $1/e$. Recall from a previous section that the exact probability of any max-value basis element of being included for the given parameters is $c(n, k) = c(200, k) \geq 0.37$ for any $k \leq n = 200$. The minimum of $c(200, k)$ is obtained around $k = 52$, and $c(200, k) \approx 0.37$ for around $40 \leq k \leq 90$ (i.e. around 25% of the time due to ranks being chosen uniformly at random from the range $[1, n]$). In these cases, the distribution of the random variable

$$Y_u := \frac{1}{t}\sum_{j=1}^{t} \mathbb{1}_{u \text{ included in trial } j}$$

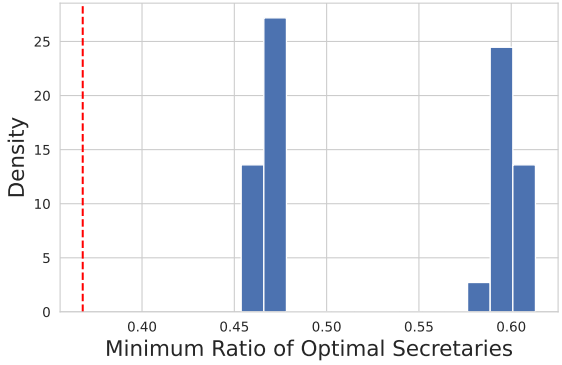
tends to a normal distribution with mean ≈ 0.37 and standard deviation $\approx \sqrt{(0.37)(1-0.37)/1000} \approx 0.015$. Thus, $X_t(M)$ will be the minimum of the Y_u, which are approximately normally distributed and are at least 40



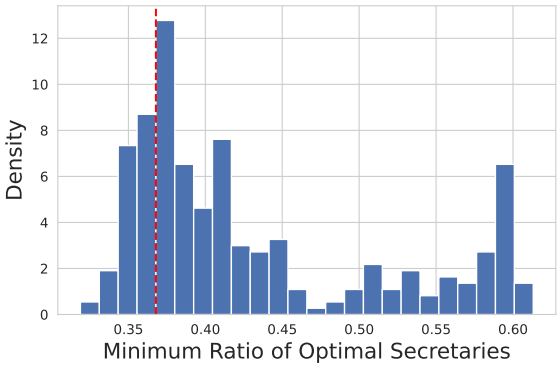
(a) 100 Uniform Matroids ($n = 200$)



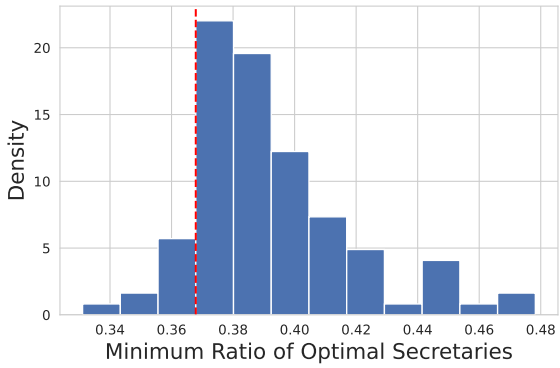
(b) 100 Graphic Matroids ($n = 100$)



(c) 30 Transversal Matroids ($|L| = n \in \{25, 50\}), |R| \in \{25, 50\}, p \in \{0.3, 0.5\}$)



(d) 300 Linear Matroids ($n = 15, \mathbb{F}_{37}$)



(e) 100 General Matroids ($n = 15$)

Figure 2: Distribution of the ratio of the least frequently occurring element in the optimal offline solution. The data is gathered over with $t = 1000$ trials of JSZ(e) on each matroid. The dashed line is plotted at $x = \frac{1}{e}$.

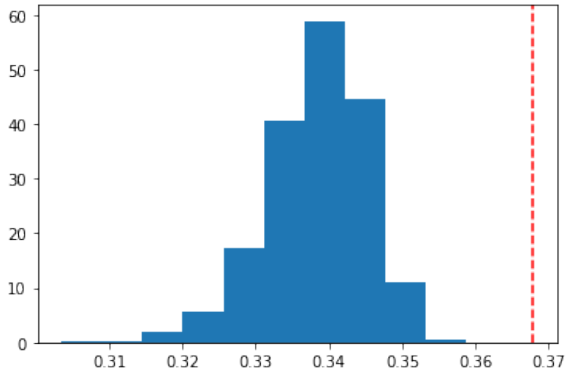


Figure 3: With the worst case rank of 52 if $n = 200$ according to Theorem 2.1.1, a distribution of the minimum of 52 $\mathcal{N}(0.37, 0.015^2)$ random variables.

in number. An approximation of this distribution is plotted in Figure 3 which emulates the results in Figure 2a quite closely. As another sanity check, we can see that the minimum observed value, which is around 0.325, is some 2.7 standard deviations from the mean. Since all the Y_u have the same approximately normal distribution D, we can view the experiment as drawing at least $40 \times 100 = 4000$ samples from D. By typical normal distribution estimates, the expected number of observations 2.7 standard deviations below the mean out of 4000 samples is 12 (well above the true amount observed).

An essentially identical methodology can be used to explain all other observed values below $1/e$. The only thing to keep in mind is that if Conjecture 1.4.3 holds, the probability that any element of an optimal offline solution is included in the algorithm's output is at least $\frac{1}{e}$. Estimates of variance can be obtained likewise, and similar normal distribution estimates can be used from here.

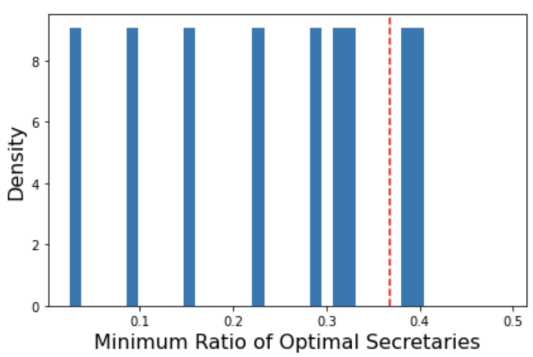
Another thing to note is that when linear matroids are randomly generated, they are very likely to be similar to uniform matroids. Therefore, it is reasonable to expect that the performance of the algorithm is dependent on k, the rank of the matroid as in Figure 1. We divided the result into five categories based on the value of $\frac{k}{n}$. Consistent with the earlier result, the competitiveness of the algorithm was nearest to e when $\frac{k}{n} \in [0.2, 0.4)$, but it decreased significantly as it grew larger. The graphs for are provided in Figure 6 in the Appendix.

4 Experimental Evaluation of Virtual Algorithm

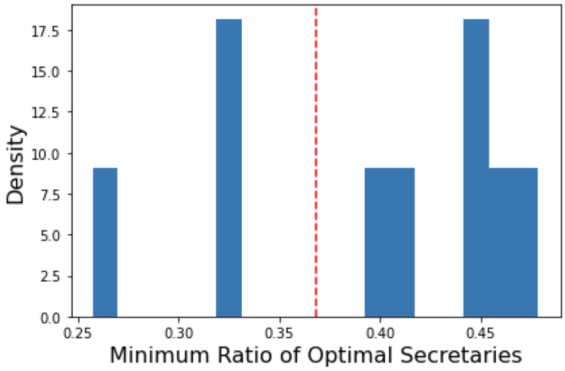
In this section, we present another algorithm that we think is of interest. Reference [2] provides two algorithms ("Virtual" and "Optimistic") for the multiple choice secretary problem, where the algorithm chooses a fixed number of k elements instead of 1. This is equivalent to the matroid secretary problem given a k-uniform matroid. Our algorithm JSZ(e) is somewhat similar to the Optimistic algorithm, but it is known that the Virtual algorithm is easier to analyze analytically than the Optimistic algorithm. Therefore, we propose a generalization of the Virtual algorithm in hopes that it will yield a result that is easier to analyze. Consider the following algorithm

1. Sample the first $\lfloor \frac{n}{e} \rfloor$ elements and store them in S.
2. Initialize $R = S = \{j_1, \cdots, j_{|R|}\}$, where the elements are in decreasing order of weight.
3. Find the max-weight basis $\{X_1, \cdots, X_m\}$ for S where $m = \text{rank}(S)$ and the elements are in decreasing order of weight. Also denote $P = \mathcal{U} - S$.
4. Initialize $A = \emptyset$, which will contain the algorithm's answer, and a counter $i = 1$
5. Let $\Delta_i = \text{span}(X_1, \cdots, X_i) - \text{span}(X_1, \cdots, X_{i-1})$. For all elements $u \in P \cap \Delta_i$ enumerated in a random order, add u to A if and only if $A \cup \{u\} \in \mathcal{I}$ and $v(u) > v(j_i)$ and $j_i \in S$. Add element u to R while maintaining the invariant that R is in decreasing order of values

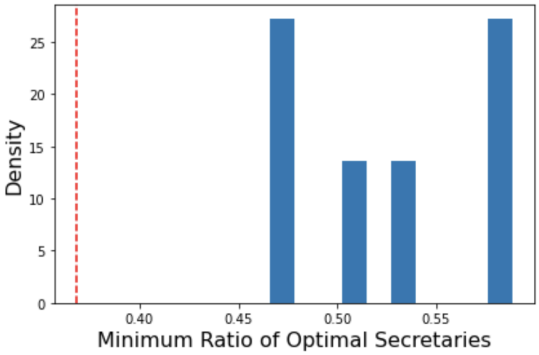
[3] As opposed to calculating the ratio of the weight of JSZ(e)'s answer and the weight of the optimal set.
[4] The smallest value of m is calculated by setting $\binom{m}{2} = n$. The largest value is chosen to avoid overly sparse graphs.
[5] Otherwise, uniformly generating vectors almost certainly yields a full-rank matroid.



(a) 10 Graphic Matroids ($n = 101$)



(b) 10 Graphic Matroids; $\frac{1}{2}|E| \leq |V| \leq |E|$ ($n = 101$)



(c) 10 Graphic Matroids; $|E| \leq |V| \leq 2|E|$ ($n = 101$)

Figure 4: Numerical analysis suggests that the algorithm works better for sparse graphs with low rank

6. If $i < m$, increment i and go repeat step 5.
7. For all remaining elements i of P, accept if and only if $|R| < k$ or $j_k \in S$ where k is the rank of $\mathcal{U}$

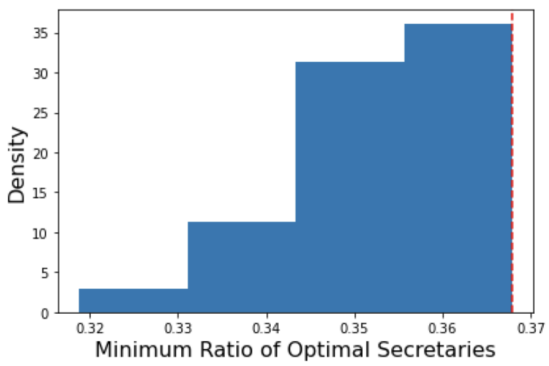
Similar to the earlier section, we also analyzed this algorithm numerically. First, we generated 10 random graphic matroids with 3 different parameters. The first graph in 5 shows the results for 10 random graphic matroids; the second graph is for 10 random graphic matroids where $\frac{1}{2}|E| \leq |V| \leq |E|$; and the third is for 10 random graphic matroids where $|E| \leq |V| \leq 2|E|$. As the graph gets sparser, the rank of the matroid should also decrease. This numerical analysis suggests that the algorithm performs better on graphic matroids of low rank.

We also ran numerical analysis on uniform matroids with $n = 100$ elements, first with rank $k = 3$ and also with rank $k = 50$. The results show no significant difference between the two examples. Further research is needed to properly understand the cases where this algorithm performs well.

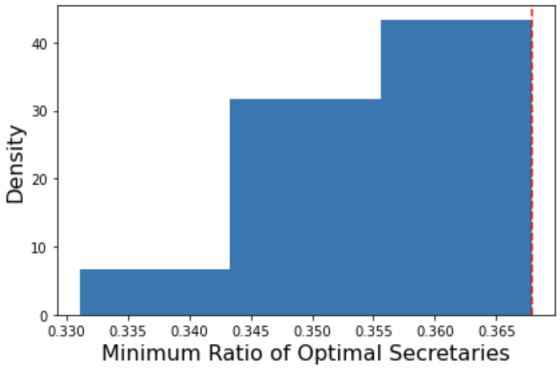
5 Conclusion

In this project, we showcased a few negative results: that no algorithm for the free-order matroid secretary problem could be strongly $(e - \Omega(1))$-competitive, and that the analysis of the algorithm presented in [8] is tight. We then proposed a modified version of the algorithm that we hypothesize is e-competitive for all matroids (as the universe size tends to infinity), and provide some theoretical and experimental evidence for this claim. Finally, we ran lightweight experiments on a generalization of the virtual algorithm of [3] for matroids.

Further research would be needed to analytically verify that the $c(n, k)$ ratio we obtained for uniform matroids is indeed above $1/e$ for any pair of n and k. Similarly, more thorough analytical analysis would be required to prove that our numerical analysis is true for any matroid.



(a) 100 Uniform Matroids ($n = 100, k = 3$)



(b) 100 Uniform Matroids ($n = 100, k = 50$)

Figure 5: Numerical analysis suggests that the performance of the algorithm is irrelevant of the rank

References

[1] P. D. Azar, R. Kleinberg, and S. M. Weinberg. Prophet inequalities with limited information. In C. Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1358–1377. SIAM, 2014.

[2] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. A knapsack secretary problem with applications. In M. Charikar, K. Jansen, O. Reingold, and J. D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 16–28, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[3] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Matroid secretary problems. *J. ACM*, 65(6), nov 2018.

[4] S. Chakraborty and O. Lachish. Improved competitive ratio for the matroid secretary problem. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, page 1702–1712, USA, 2012. Society for Industrial and Applied Mathematics.

[5] J. Correa, P. Dütting, F. Fischer, and K. Schewior. Prophet inequalities for i.i.d. random variables from an unknown distribution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, EC '19, page 3–17, New York, NY, USA, 2019. Association for Computing Machinery.

[6] E. B. Dynkin. Optimal choice of the stopping moment of a markov process. *Dokl. Akad. Nauk SSSR*, 150(2):238–240, 1963.

[7] M. Feldman, O. Svensson, and R. Zenklusen. A simple $o(\log\log(\text{rank}))$-competitive algorithm for the matroid secretary problem, 2014.

[8] P. Jaillet, J. A. Soto, and R. Zenklusen. Advances on matroid secretary problems: Free order model and laminar case. In M. X. Goemans and J. R. Correa, editors, *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings*, volume 7801 of *Lecture Notes in Computer Science*, pages 254–265. Springer, 2013.

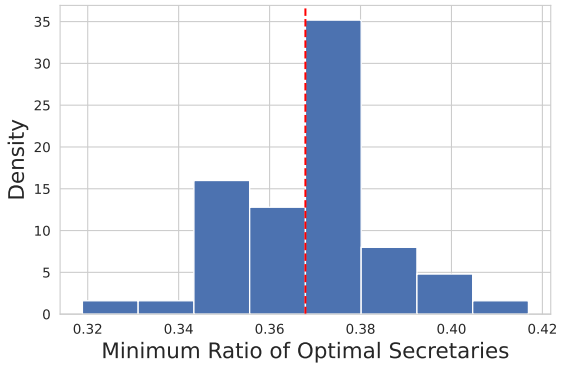
[9] R. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, page 630–631, USA, 2005. Society for Industrial and Applied Mathematics.

[10] D. E. Knuth. Random matroids. *Discrete Mathematics*, 12(4):341–358, 1975.

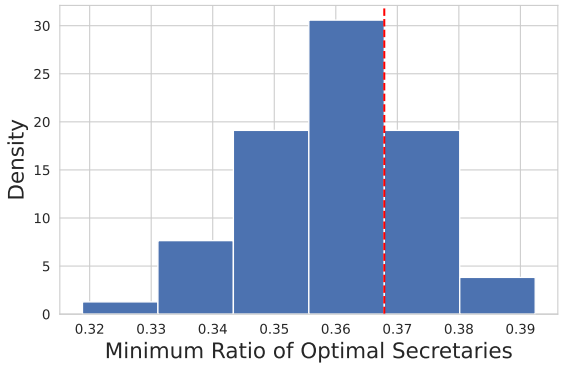
[11] N. Korula and M. Pal. Algorithms for secretary problems on graphs and hypergraphs, 2008.

[12] J. A. Soto, A. Turkieltaub, and V. Verdugo. Strong algorithms for the ordinal matroid secretary problem, 2018.

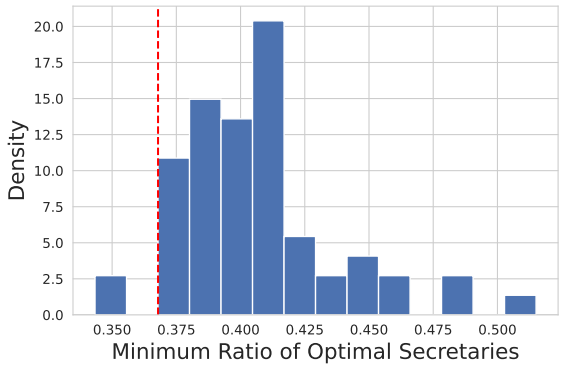
A Rank Breakdown of JSZ(e) Linear Matroid Simulation



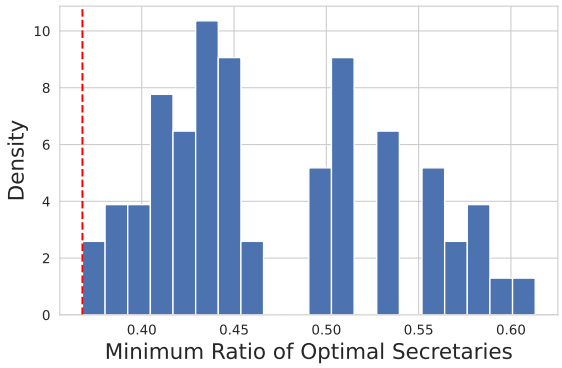
(a) Rank $k \in [0, 20)$



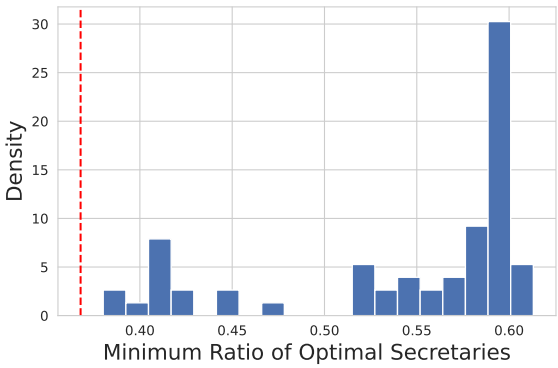
(b) Rank $k \in [20, 40)$



(c) Rank $k \in [40, 60)$



(d) Rank $k \in [60, 80)$



(e) Rank $k \in [80, 100]$

Figure 6: A breakdown of the results for 300 linear matroids with universe size $n = 100$, presented in Figure 2d, according to the matroid's rank. As with the explanation of section 3.3, we hypothesize that the reason for a relatively large proportion of the observations lying below the $1/e$ threshold is due to the simulated matroids being closer to uniform for lower ranks (less than half of the universe size).