Improved Competitive Ratio for the Matroid Secretary Problem

Sourav Chakraborty* Oded Lachish†‡§

Abstract

The Matroid Secretary Problem, introduced by Babaioff *et al.* (2007), is a generalization of the Classical Secretary Problem. In this problem, elements from a matroid are presented to an on-line algorithm in a random order. Each element has a weight associated with it, which is revealed to the algorithm along with the element. After each element is revealed the algorithm must make an irrevocable decision on whether or not to select it. The goal is to pick an independent set with the sum of the weights of the selected elements as large as possible.

Babaioff *et al* gave an algorithm for the Matroid Secretary Problem with a *competitive ratio* of $O(log\rho)$, where ρ is the rank of the matroid. It has been conjectured that a constant competitive-ratio is achievable for this problem. In this paper we give an algorithm that has a competitive-ratio of $O(\sqrt{log\rho})$.

1 Introduction

Dynkin [7] introduced the *Classical Secretary Problem*: There are n candidates for a secretarial post. Precisely one of them has to be selected for the post. The candidates are interviewed sequentially in a random order. After each interview a score is given to the candidate just interviewed. The goal is to select a candidate with the maximum score. But there is one "catch". Once a candidate has been interviewed, the decision on whether to hire the candidate or not has to be taken before the next candidate is interviewed. If that candidate is hired then the interview process stops. If the candidate is not hired, then the next candidate is interviewed, and the candidate that was not hired cannot be hired subsequently. Because of this "catch", the best that can be done is to maximize the probability of picking the best candidate. Lindley [12] and Dynkin [7] gave an algorithm that hires the best candidate with probability at least $1/e$. For the results in this paper we need an equivalent formulation of the problem, which we describe next.

Classical Secretary Problem: *Given a randomly ordered sequence of n elements, each with a positive weight, design an on-line algorithm for picking an element of the sequence that maximizes the expected weight of the chosen element.*

A natural generalization is to allow k secretaries to be hired. Now the goal is to hire the best k candidates. This problem is known as the *k-Secretary Problem*. Babaioff *et al* [4] generalized it even further to the *Matroid Secretary Problem*. In this problem there is a matroid $(\mathcal{U}, \mathcal{I})$. Each element $x \in \mathcal{U}$ has a non-negative weight $wt(x)$ and each $S \in \mathcal{I}$ has weight $\sum_{x \in S} wt(x)$. The algorithm has access to the matroid via an oracle that on input set $S \subseteq \mathcal{U}$ replies whether $S \in \mathcal{I}$ or not. The elements and their weights are revealed one at a time in a random order. As each element is revealed, a decision whether to select or reject it, must be made subject to the following constraints: the decision to select or reject is irrevocable; the decision must be made before the next element in the sequence is revealed; the set of selected elements must belong to $\mathcal{I}$ at all times. The algorithm's payoff is the weight of the set of selected elements. The goal is to maximize the expected payoff. The *competitive-ratio* of the algorithm is the ratio of the maximum weight of a set in $\mathcal{I}$ to the expected payoff.

Babaioff *et al.* [4] established a connection between the Matroid Secretary Problem and *mechanism design*. Essentially they show that each algorithm for a specific family of matroids can be converted into a *Truthful Mechanism* for what they call corresponding matroid domain. In the same paper they gave an algorithm for the Matroid Secretary Problem that achieves a competitive-ratio of $O(log\rho)$, where ρ is the rank of the matroid. Since then this problem has received significant attention. Yet, no algorithm with a better competitive-ratio has been found (until this paper). It has however often been conjectured that a constant competitive-ratio is achievable.

Related results: Although to our knowledge, the best known competitive-ratio for the Matroid Secretary Problem is the one presented in this paper, significantly better results are known when there are certain restrictions on either the matroid structure or the weights of the elements.

For example, constant-competitive-ratio algorithms

*Chennai Mathematical Institute, Chennai, India. Email: sourav@cmi.ac.in

†Birkbeck, University of London, London, UK. Email: oded@dcs.bbk.ac.uk

‡Research supported in part by EPSRC awards EP/G064679/1 and EP/G069034/1

§Research supported in part by an ERC-2007-StG grant number 202405.

have been obtained for specific families of matroids like graphic matroids [4], transversal matroids of bounded left degree, and their truncation [4], uniform/partition matroids [2, 10], transversal matroids [6, 11] and Laminar matroids [9]. There are also some constant-competitive-ratio results when various restrictions are imposed on the weights of the elements. For example, if the weights of the elements are randomly assigned, then there is a constant-competitive-ratio algorithm [13].

Many other generalizations of the Classical Secretary Problem have also been studied, for example, the *Submodular Secretary Problem* [5] and the *Knapsack Secretary Problem* [2]. For a detailed survey of this field, please refer to [8] and [3].

Our result: We give an algorithm for the Matroid Secretary Problem that has a competitive-ratio of $O(\sqrt{\log \rho})$. This may be an indication that the Matroid Secretary Problem actually admits a constant competitive-ratio.

2 Overview of our algorithm

This overview refers to a number of matroid properties that are formally defined in the following section. In order to simplify the exposition of the algorithm we set OPT to be the maximum weight of an independent set in the matroid. It is assumed that the weights of all the elements of the matroid are powers of 2. In the worst case this results in doubling the competitive-ratio of our algorithm. The advantage of this assumption is that it enables the partitioning of the elements into sets, called buckets, where each bucket is a maximal set of matroid elements that have the same weight. The main algorithm we present utilises three separate algorithms. The first we call the classical algorithm since it is the algorithm described in Lindley [12] and Dynkin [7] for the Classical Secretary Problem. The second we call the simple algorithm and the third we call the protection algorithm.

The main algorithm consists of two parts of which only one is used. The decision as to which one is made by a toss of a fair coin. The first part is simply the classical algorithm. The second part consists of two phases. In the first phase approximately half of the elements of the matroid are revealed without any of them being selected. In the second phase either the simple algorithm or the protection algorithm is deployed. The choice of which algorithm to deploy is done according to the knowledge about the elements revealed in the first phase.

The goal of the first part of the algorithm is to ensure the claimed competitive-ratio when the matroid has an element with a "large" weight, where by "large" we mean at least as large as the expected payoff of the algorithm. The second part of the algorithm is to ensure the claimed competitive-ratio when the matroid does not have a "large" element.

The assumption that there is no element of "large" weight is vital for the proof of correctness of the second part of the algorithm. Under this assumption, we show that there exist two sets of buckets, Core-Buckets and Good-Buckets, which satisfy a number of crucial properties: Good-Buckets is a superset of Core-Buckets; the cardinality of Good-Buckets is $O(\log \rho)$ (recall that ρ is the rank of the matroid); the rank of each bucket in Good-Buckets is "large" enough to enable us to use a combination of concentration inequalities and the union bound; the sum of rank of a bucket times weight of an element in it over all buckets in Core-Buckets is $\Omega(OPT)$. These properties enable us to show, for example, that with high probability the rank of the set of elements revealed in the first phase is at least $\rho/3$. This is used to show that with high probability by the end of the first phase the algorithm has a good approximation of ρ. Hence from here until the end of the overall description, we assume that ρ is known.

The importance of Core-Buckets and Good-Buckets stems from the fact that in the first phase the algorithm selects a set of buckets called Selected-Buckets, and we show that, with high probability, Selected-Buckets is a superset of Core-Buckets and is a subset of Good-Buckets. In the second phase the algorithm selects elements only from Selected-Buckets. We use the fact that Selected-Buckets includes Core-Buckets in order to upper bound the competitive-ratio. The fact that Selected-Buckets is a subset Good-Buckets implies that the concentration results we prove for Good-Buckets also hold for Selected-Buckets.

We now explain what the algorithm does in the second phase, starting with an explanation of how the simple algorithm works. This explanation is followed by a description of how and why the main algorithm uses the simple algorithm. The simple algorithm receives a subset of Selected-Buckets from the first phase of the main algorithm. It then starts revealing the remaining elements, selecting an element, provided it is in one of the given buckets and that adding it to the set of all previously selected elements results in an independent set.

There are two ways the simple algorithm may be employed. The first is when the first phase of the main algorithm gives the simple algorithm a set consisting of a single bucket. In this case the set of elements revealed in the first phase that are in this bucket, contains an independent subset with a large weight. If this had been known in advance, then the simple algorithm could have been used with this bucket on the elements

revealed in the first phase. This would have ensured the required payoff. This is because all of the elements in the given bucket have the same weight. Thus, the simple algorithm, described here, coincides with the greedy algorithm for matroids when applied only on the elements of the given bucket that were revealed in the first phase. However, although we do not know this in advance, since the bucket given is in Good-Buckets the concentration results for Good-Buckets imply that the subset of the buckets elements, revealed in the second phase, is similar with high probability.

The second way the simple algorithm is used is as follows. The first phase of the main algorithm gives the simple algorithm a "special" subset of the Selected-Buckets. The union of these "special" buckets contains an independent subset of large weight that consists of elements u of the following type: each element u was revealed in the first phase; $\{u\}$ forms an independent set together with any independent subset of elements that were revealed in the first phase and are contained in the union of the "special" buckets. Note that this implies that if we had used the simple algorithm on the elements that were revealed in the first phase and contained in the union of the "special" buckets, then every element such as u would have been selected. However, although we do not know this in advance, since the "special" buckets are Good-Buckets the concentration results for Good-Buckets imply that the subset of elements in the "special" buckets, revealed in the second phase, is similar with high probability.

Note that the simple algorithm will be employed if a bucket like the one described in the first way exists, or if a "special" set of buckets as described in the second way exists. If many such sets exist, then one of them is selected arbitrarily in the first phase of the main algorithm and this is the set given to the simple algorithm. If no such sets exist then the protection algorithm is used.

The protection algorithm is more involved than the simple algorithm and hence a more detailed intuition appears together with its pseudo-code. Here we shall describe some of the key ideas it incorporates. Assume that we had a "magic" oracle that can provide a basis for every Bucket. In an algorithm we could use this oracle as follows. When deciding whether to select a candidate element we act as follows: select the element only if the rank of the set obtained by adding the element to the set described next is larger the rank of this set. The set is the union of all elements already selected and all the elements that are of larger weight than the candidate and in bases given by the "magic" oracle. This ensures that it will never be the case that an element will not be selected because it forms a dependent set together with elements of lower weight that have already been selected. This holds since if indeed such a set of lower weight elements existed, then its union with all the bases of buckets of larger element weight that were given by the "magic" oracle is dependent. This in turn means that one of these lower weight elements was not selected, hence a contradiction. Thus the bases given by the "magic" oracle "protect" elements from elements of smaller weight.

In reality such a "magic" oracle does not exists. So instead of the basis given by the "magic" oracle we use a union of a number of buckets. This union may not include a basis for the protected bucket, yet it is similar in a manner that will be described later when we present the algorithm. We shall show that such sets exist if the sets needed for the simple algorithm to be used do not exist.

3 Preliminaries

We use $[\alpha]$ to denote $\{1,\ldots,\lfloor\alpha\rfloor\}$ for any non-negative real α. We use $\mathbb{N}$ to denote the non-negative integers, $\mathbb{N}_0$ the non-negative even numbers and $\mathbb{N}_1$ the non-negative odd numbers. All logarithms are base 2.

3.1 Properties of Matroids

DEFINITION 3.1. *Let $(\mathcal{U},\mathcal{I})$ be an ordered pair, where $\mathcal{U}$ is a set of elements and $\mathcal{I}$ is a family of subsets of $\mathcal{U}$, then $(\mathcal{U},\mathcal{I})$ is a* matroid *if the following holds*

- *If $A \in \mathcal{I}$ and $A' \subset A$, then $A' \in \mathcal{I}$*
- *If $A, A'' \in \mathcal{I}$ and $|A''| > |A|$, then there exists $a \in A'' \setminus A$ such that $(A \cup \{a\}) \in \mathcal{I}$.*

We call the sets in $\mathcal{I}$, independent sets, and any independent set that is maximal is called a basis of $\mathcal{U}$.

From here on $(\mathcal{U},\mathcal{I})$ is a fixed matroid and $n = |\mathcal{U}|$. We also have a fixed weight function wt from $\mathcal{U}$ to the non-negative reals. For every $u \in \mathcal{U}$ we refer to $wt(u)$ as the weight of u. For any $V \subseteq \mathcal{U}$ we define $wt(V) = \sum_{u\in V} wt(u)$. We shall assume that all the weights are powers of 2. That is, we round any value to the largest power of 2 that is smaller than it. In the worst case this results in doubling the competitive-ratio of our algorithm.

DEFINITION 3.2. **[Rank,Υ,OPT,$\widehat{OPT}$]** *For every $\mathcal{U}' \subseteq \mathcal{U}$ we define $Rank(\mathcal{U}') = \max\{|I| \mid I \in \mathcal{I} \cap 2^{\mathcal{U}'}\}$. We set $\Upsilon = \lceil \log Rank(\mathcal{U})\rceil$. For every $\mathcal{U}' \subseteq \mathcal{U}$ we define $OPT(\mathcal{U}') = \max\{wt(I) \mid I \in \mathcal{I} \cap 2^{\mathcal{U}'}\}$. We write just OPT when $\mathcal{U}' = \mathcal{U}$ and set $\widehat{OPT}$ to be the smallest power of 2 that is at least $\sum_i OPT(\{u \in \mathcal{U} \mid wt(u) = 2^i\})$.*

This definition of $\widehat{OPT}$ was chosen in order to simplify the exposition. Observe that $\widehat{OPT} \geq OPT$.

DEFINITION 3.3. **[Buckets]** *For every integer i we set $B_i = \{u \in \mathcal{U} \mid wt(u) = \frac{\widehat{OPT}}{2^i}\}$. For every set of integers S we define $B_S = \bigcup_{i \in S} B_i$.*

DEFINITION 3.4. **[CoreBs, GoodBs]** *CoreBs is the set of all $i \in \left[\lceil \frac{\log \Upsilon}{2} \rceil + 11, 2\Upsilon\right]$ such that $Rank(B_i) \geq \max\{\frac{2^{i-8}}{\Upsilon}, \frac{\sqrt{\Upsilon}}{2^8}\}$. GoodBs is the set of all $i \in \left[\lceil \frac{\log \Upsilon}{2} \rceil + 11, 2\Upsilon\right]$ such that $Rank(B_i) \geq \max\{\frac{2^{i-16}}{\Upsilon}, \frac{\sqrt{\Upsilon}}{2^{16}}\}$.*

In all the propositions until the end of this subsection it is assumed that $\Upsilon > 2^{64}$ and $\max\{wt(u) \mid u \in \mathcal{U}\} < \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$. Hence we omit this from their statement.

PROPOSITION 3.1. $\sum_{i \in CoreBs} OPT(B_i) \geq \frac{\widehat{OPT}}{4}$.

Proof. Let $\alpha = \lceil \frac{\log \Upsilon}{2} \rceil + 11$, and T_1 be the set of all $i \in [\alpha, 2\Upsilon]$ such that $Rank(B_i) < \frac{2^{i-8}}{\Upsilon}$, and T_2 be the set of all $i \in [\alpha, 2\Upsilon]$ such that $Rank(B_i) < \frac{\sqrt{\Upsilon}}{2^8}$ and T_3 be the set of all integers greater than 2Υ. Recall that, as $\max\{wt(u) \mid u \in \mathcal{U}\} < \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$, for every $i < \lceil \frac{\log \Upsilon}{2} \rceil + 11$ we have that $B_i = \emptyset$ and hence $\sum_{i \in CoreBs} OPT(B_i) \geq \widehat{OPT}/2 - \sum_{i \in [3]} OPT(T_i)$. Therefore, by Definition 3.2 and Definition 3.4, to prove the proposition we only need to show that $\sum_{i \in [3]} OPT(T_i) < \widehat{OPT}/4$. By definition $OPT(T_1) < \sum_{i \in [\alpha, 2\Upsilon]} \frac{2^{i-8}}{\Upsilon} \cdot \frac{\widehat{OPT}}{2^i} \leq \widehat{OPT}/16$ and also that $OPT(T_2) < \sum_{i \in [\alpha, 2\Upsilon]} \frac{\sqrt{\Upsilon}}{2^8} \cdot \frac{\widehat{OPT}}{2^i} \leq 2^{-\alpha} \sum_{i \in [2\Upsilon]} \frac{\sqrt{\Upsilon}}{2^8} \cdot \frac{\widehat{OPT}}{2^i} \leq \widehat{OPT}/16$. Finally, $OPT(T_3) < \sum_{i > 2\Upsilon} 2^{\Upsilon} \cdot \frac{\widehat{OPT}}{2^i} \leq \sum_{i > 0} 2^{\Upsilon} \cdot \frac{\widehat{OPT}}{2^{2\Upsilon} 2^i} \leq \widehat{OPT}/16$, where the first inequality is because for every i we have that $Rank(B_i) \leq 2^{\Upsilon}$ and the last inequality is because $\Upsilon > 2^{64}$.

Among other things the following Propositions are used to bound the effect of using concentration inequalities.

PROPOSITION 3.2. *For every i, j, k, T, where $k \geq j \geq 4\sqrt{\Upsilon}$ and $k - j \leq \sqrt{\Upsilon}$, and $i \in T \subseteq [j, k] \cap GoodBs$, we have that $Rank(B_i)^{3/4} \geq \sqrt{Rank(B_T)}$.*

Proof. The weight of every element in B_T is at least $\frac{\widehat{OPT}}{2^k}$ and hence $Rank(B_T) \leq 2^k$. Now, as $i \in GoodBs$, Definition 3.4 implies that $Rank(B_i) \geq 2^{j - 16 - \log \Upsilon} \geq 2^{k - \sqrt{\Upsilon} - 16 - \log \Upsilon}$. Note as $k \geq 4\sqrt{\Upsilon}$ and $\Upsilon > 2^{64}$ we get that $k/4 > 3(16 + \sqrt{\Upsilon} + \log \Upsilon)/4$ and hence $Rank(B_i)^{3/4} \geq 2^{3(k - \sqrt{\Upsilon} - 16 - \log \Upsilon)/4} > 2^{k/2} \geq \sqrt{Rank(B_T)}$.

PROPOSITION 3.3. *For every $T \subseteq GoodBs \cap [4\sqrt{\Upsilon}, 2\Upsilon]$ we have that $\sum_{i \in T} 16 \cdot \Upsilon \cdot Rank(B_i)^{3/4} \cdot \frac{\widehat{OPT}}{2^i} \leq \frac{\widehat{OPT}}{2^{20}\Upsilon^2}$.*

Proof. Since $|T| \leq 2\Upsilon$, it is sufficient to prove that for every $i \in T$ we have that $16 \cdot \Upsilon \cdot Rank(B_i)^{3/4} \cdot \frac{\widehat{OPT}}{2^i} \leq \frac{\widehat{OPT}}{2^{21}\Upsilon^3}$. Fix $i \in T$. By definition $Rank(B_i) \leq 2^i$ and hence $Rank(B_i)^{3/4} \leq 2^{\frac{3i}{4}}$. Therefore $16 \cdot \Upsilon \cdot Rank(B_i)^{3/4} \cdot \frac{\widehat{OPT}}{2^i} \leq 16 \cdot \Upsilon \cdot \widehat{OPT} \cdot 2^{-\frac{i}{4}} \leq \frac{\widehat{OPT}}{2^{21}\Upsilon^3}$, where the last inequality is because $i \geq 4\sqrt{\Upsilon}$ and $\Upsilon > 2^{64}$.

3.2 Permutations of $\mathcal{U}$ Let $\pi : \mathcal{U} \mapsto [n]$ be a uniformly selected bijection and ℓ selected according to the Binomial distribution $Binomial(n, 1/2)$. Let $L = \{u \in \mathcal{U} \mid \pi(u) \leq \ell\}$ and $R = \mathcal{U} \setminus L$.

DEFINITION 3.5. **[L,R Buckets]** *For every integer i we set $B_i^L = B_i \cap L$ and $B_i^R = B_i \cap R$.*

We next define an event $\mathcal{A}$ which contains all the conditions required for proving that the algorithm we present works. We use both the Chernoff and the Talagrand inequalities in order to prove that $\mathcal{A}$ holds with sufficiently high probability.

DEFINITION 3.6. **[Event $\mathcal{A}$]** *Let $\mathcal{A}$ be the event that L is such that the following holds. For every i, j, k, T, where $i \in GoodBs$, and $k \geq j \geq 4\sqrt{\Upsilon}$ and $k - j \leq \lceil \sqrt{\Upsilon} \rceil$, and $T \subseteq [j, k] \cap GoodBs$, we have that*

1. *$Rank(L) \geq \frac{Rank(\mathcal{U})}{3}$, $\sum_{i \in CoreBs} OPT(B_i^L) \geq \frac{\widehat{OPT}}{16}$,*
2. *$|Rank(B_i^L) - Rank(B_i^R)| \leq 4 \cdot \log(\Upsilon) \cdot \sqrt{Rank(B_i)}$,*
3. *$|Rank(B_T^L) - Rank(B_T^R)| \leq 8 \cdot \Upsilon \cdot \sqrt{Rank(B_T)}$,*
4. *if $i \in T$, then $|Rank(B_{T \setminus \{i\}}^L \cup B_i^R) - Rank(B_T^L)| \leq 8 \cdot \Upsilon \cdot \sqrt{Rank(B_T)}$.*

THEOREM 3.1. *If $\Upsilon > 2^{64}$ and $\max\{wt(u) \mid u \in \mathcal{U}\} < \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$, then event $\mathcal{A}$ holds with probability at least $\frac{13}{16}$.*

Proof. The proof follows from union bound and Proposition 3.6, Proposition 3.4 and Proposition 3.5, which are all proved further on.

From here until the end of this subsection, we shall assume that $\Upsilon > 2^{64}$ and $\max\{wt(u) \mid u \in \mathcal{U}\} < \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$. Hence we will omit this from the statements of the propositions appearing next.

To prove that with high probability conditions 2, 3 and 4 of event $\mathcal{A}$ hold, we use an application of Talagrand's concentration inequality which appears in

chapter 7 section 7 of Alon and Spencers second edition of the book *The Probabilistic Method* [1]. Denote the elements of $\mathcal{U}$ by $u_1, \ldots, u_n$. For every $i \in [n]$ let Ω_i be the probability space consisting of the complementing events $u_i \in L$ and $u_i \in R$. Set $\Omega = \prod_{i \in [n]} \Omega_i$. A function h from Ω to the reals is Lipschitz if $|h(x) - h(y)| \leq 1$ for every $x, y \in \Omega$ that differ on at most one coordinate. The next definition is quoted from [1].

DEFINITION 3.7. **[Definition 3 of [1]]** *Let $f : \mathbb{N} \longrightarrow \mathbb{N}$. h is f-certifiable if whenever $h(x) \geq s$ there exists $I \subseteq \{1, \ldots, n\}$ with $|I| \leq f(s)$ so that all $y \in \Omega$ that agree with x on the coordinates I have $h(y) \geq s$.*

The following theorem is an adaptation of Theorem 7.7.1 from [1] to our needs.

THEOREM 3.2. *If h is Lipschitz and f-certifiable, then for x selected uniformly from Ω and all b, t,*

$$Pr[h(x) \leq b - t\sqrt{f(b)}] \cdot Pr[h(x) \geq b] \leq e^{-t^2/4}.$$

We shall use Theorem 3.2 for the *Rank* function. To do so we need the following two facts. For every $V \subseteq \mathcal{U}$ define $Rank_V : \Omega \longrightarrow \mathbb{N}$ so that $Rank_V(x) = Rank(V \cap L)$

FACT 3.1. *$Rank_V$ is Lipschitz and f-certifiable, where $f(x) = x$ for every $x \in \Omega$.*

We abuse notations and treat L as if it was an element in Ω (that is, an indicator function for L).

FACT 3.2. *L is distributed uniformly over $2^{\mathcal{U}}$.*

Proof. Let $T \subseteq \mathcal{U}$. The probability that $L = T$ is the probability that $|L| = |T|$ times the probability that the elements of T are selected to be in L. This is $2^{-n} \cdot \binom{n}{|T|} \cdot \binom{n}{|T|}^{-1} = 2^{-n}$.

LEMMA 3.1. *Let $V \subseteq \mathcal{U}$ and m be the median value of $Rank_V(x)$ over all $x \in \Omega$, then for every $t \geq 2$*

$$Prob[|Rank_V(L) - m| \geq t\sqrt{m}] \leq e^{2-t^2/4}.$$

Proof. To prove the lemma we apply Theorem 3.2 twice. In both times we take $h \equiv Rank_V$ and f to be the identity function. In the first time we set $b = m$ to get

$$Pr[Rank_V(L) \leq m - t\sqrt{m}] \cdot Pr[h(x) \geq m] \leq Pr[Rank_V(L) \leq m - t\sqrt{m}]/2 \leq e^{-t^2/4}.$$

In the second time we set $b = m + t\sqrt{m}$ to get

$$Pr[Rank_V(L) \leq m + \delta] \cdot Pr[h(x) \geq m + t\sqrt{m}] \leq Pr[Rank_V(L) \geq m + t\sqrt{m}]/2 \leq e^{-t^2/4},$$

where the first inequality is because $\delta > 0$ since $t > 2$. The lemma follows by the union bound.

PROPOSITION 3.4. *With probability at least 15/16, for every $i \in GoodBs$ we have that*

$$|Rank(B_i^L) - Rank(B_i^R)| \leq 4 \cdot \log(\Upsilon) \cdot \sqrt{Rank(B_i)}.$$

Proof. Fix $i \in GoodBs$. Let m be the median of the value of $Rank(B_i^L)$ over all choices of L. Observe that $Rank(B_i^L) \equiv Rank_{B_i}(L)$. Since $i \in GoodBs$ and $\Upsilon > 2^{64}$ by Fact 3.1 and Fact 3.2, Lemma 3.1 asserts that $|Rank(B_i^L) - m| \geq 2 \cdot \log(\Upsilon)\sqrt{m}$, with probability at most $e^{2-2\log\Upsilon} \leq 2^{-6}\Upsilon^{-1}$, where the inequality is because $\Upsilon > 2^{64}$.

Thus, by the union bound with probability at least 31/32 for every $i \in GoodBs$ we have that $|Rank(B_i^L) - m| \leq 2 \cdot \log(\Upsilon)\sqrt{m}$. Note that for our purposes there is no difference between B_i^R and B_i^L, since there is a trivial bijection between every possible L and every possible R. Consequently, the same proof implies that with probability at least 31/32 for every $i \in GoodBs$ we have that $|Rank(B_i^R) - m| \leq 2 \cdot \log(\Upsilon)\sqrt{m}$. Now by applying the union bound and the triangle inequality proposition follows.

PROPOSITION 3.5. *With probability at least 15/16, for every i, j, k, T, where $k \geq j \geq 8\sqrt{\Upsilon}$ and $k - j \leq \sqrt{\Upsilon}$, and $i \in T \subseteq [j, k] \cap GoodBs$, we have that*

- $|Rank(B_T^L) - Rank(B_T^R)| \leq 8 \cdot \Upsilon \cdot \sqrt{Rank(B_T)}$.
- $|Rank(B_{T\setminus\{i\}}^L \cup B_i^R) - Rank(B_T^L)| \leq 8 \cdot \Upsilon \cdot \sqrt{Rank(B_T)}$.

Proof. Fix i, j, k, T. Let m be the median of the values of $Rank(B_{T\setminus\{i\}}^L \cup B_i^R)$ over all choices of L. Note that the for our purposes there is no difference between $B_{T\setminus\{i\}}^L \cup B_i^R$ and B_T^L, since there is a trivial bijection between every possible value of L and every possible value of $(L \setminus B_i^L) \cup B_i^R$. By the same reasoning as in Proposition 3.4 we infer $|Rank(B_{T\setminus\{i\}}^L \cup B_i^R) - m| \geq 2 \cdot \log(\Upsilon)\sqrt{m}$, with probability at most $e^{2-2\Upsilon} \leq 2^{-10}\Upsilon^{-3}2^{-\Upsilon}$.

Since $\Upsilon > 2^{64}$, the union bound implies that with probability at least 63/64 for every i, j, k, T as in the proposition we have that $|Rank(B_{T\setminus\{i\}}^L \cup B_i^R) - m| \leq 2 \cdot \Upsilon\sqrt{m}$. Note that the for our purposes there is no difference between B_T^R and B_T^L, since there is a trivial bijection between every possible value of L and every possible value of R. Thus same proof implies that with probability at least 63/64 for every i, j, k, T, as in the statement of the proposition, we have that $|Rank(B_T^L) - m| \leq 2 \cdot \Upsilon\sqrt{m}$ and the same for $Rank(B_T^R)$. Now by using the union bound and the triangle inequality the proposition follows.

We next prove that Condition 1 of event $\mathcal{A}$ holds with high probability. To do so we need the Chernoff inequality which requires the following fact that is an immediate result of Fact 3.2.

FACT 3.3. *For every i and $u \in B_i$ independently, $u \in B_i^L$ with probability $\frac{1}{2}$.*

PROPOSITION 3.6. *$Rank(L) \geq Rank(\mathcal{U})/3$ and $\sum_{i \in CoreBs} OPT(B_i^L) \geq \frac{\widehat{OPT}}{16}$ with probability at least 15/16.*

Proof. Let Z be a basis for $\mathcal{U}$. By Fact 3.3 for every $z \in \mathcal{U}$ independently $z \in L$ with probability 1/2. Hence by the Chernoff inequality $|Z \cap L| \geq Rank(\mathcal{U})/3$ with probability at least $1 - e^{-Rank(\mathcal{U})/18} > 31/32$, where the inequality is because $\Upsilon > 2^{64}$.

For every $i \in CoreBs$ let $Z_i \subseteq B_i$ be an optimal basis for B_i. Fix $i \in CoreBs$. By Fact 3.2 for every $z \in Z_i$ independently, $z \in L \cap Z_i$ with probability 1/2. Since $Rank(B_i) \geq \frac{\sqrt{\Upsilon}}{2^8}$, by the Chernoff bound, $Rank(Z_i \cap L) \geq Rank(Z_i)/3$ with probability at least $1 - e^{-\frac{\sqrt{\Upsilon}}{2^8} \cdot \frac{1}{18}} \geq 1 - \frac{31}{64 \cdot \Upsilon}$, where the inequality is because $\Upsilon > 2^{64}$. By the union bound, with probability at least 15/16, for every $i \in CoreBs$ we have that $Rank(Z_i \cap L) \geq Rank(Z_i)/3$. Thus, with probability at least 15/16, $\sum_{i \in CoreBs} OPT(B_i^L) \geq \sum_{i \in CoreBs} OPT(B_i)/3 \geq \frac{\widehat{OPT}}{16}$, where the second inequality is by Proposition 3.1. The proposition follows by an additional use of the union bound.

4 The Main Algorithm

THEOREM 4.1. *Let P be the set returned by Algorithm 4.1, then the expected value of $OPT(P)$ is $\Omega(\frac{OPT}{\sqrt{\Upsilon}})$.*

The proof of this theorem appears immediately before Subsection 4.1. The pseudo-code for Algorithm 4.1 appears further on and is essential for understanding what is written in this text. Note that all the matroid related operations used in Algorithm 4.1 can be done by using only an oracle to the matroid.

In the first line of Algorithm 4.1 it is decided whether to employ the classical algorithm or not. If the classical algorithm was not employed, then Algorithm 4.1 proceeds to the initialization of sets and the revealing of about half the matroids elements. The set of revealed elements is denoted by L since it is selected exactly as L is selected in the preliminaries. Once L is known the algorithm proceeds to computing the parameters $\widetilde{OPT}$ and $\widetilde{\Upsilon}$. These parameters are computed so that with high probability $\widetilde{OPT} = \widehat{OPT}$ and $\widetilde{\Upsilon} = \Upsilon$. This is done mostly in order to simplify notations. At Line 9 the algorithm picks Selected-Buckets. Line 10 is used for checking whether there exists a set of buckets, for the simple algorithm as described in Section 2. If there does not exist such a set, then the protection algorithm is employed. If there exist such sets, then the algorithm selects an arbitrary one and proceeds to the simple algorithm, which consists of Line 12.

ALGORITHM 4.1.

1. With probability $\frac{1}{2}$ employ the classical secretary algorithm and return its output
2. $\ell \in Binomial(n, 1/2)$
3. Reveal the first ℓ elements without selecting any of them, set L to be all the elements revealed
4. Select $\xi \in \{0,1,2,3\}$ and $\eta \in \{0,1,2\}$, each independently and uniformly
5. $P \longleftarrow \emptyset$, $\widetilde{\Upsilon} \longleftarrow \eta + \log Rank(L)$
6. Set $\widetilde{OPT}$ to be 2^ξ times the smallest power of 2 that is at least $\sum_i OPT(\{u \in L \mid wt(u) = \frac{\widetilde{OPT}}{2^i}\})$
7. $L_i \longleftarrow \{u \in L \mid wt(u) = \frac{\widetilde{OPT}}{2^i}\}$ for every $i \in [2\widetilde{\Upsilon}]$
8. $L_M \longleftarrow \bigcup_{i \in M} L_i$ for every $M \subseteq [2\widetilde{\Upsilon}]$
9. Set $SelectedBs$ to be the set of all $i \in [2\widetilde{\Upsilon}]$ such that $Rank(L_i) \geq \max\left\{\frac{2^{i-12}}{\widetilde{\Upsilon}}, \frac{\sqrt{\widetilde{\Upsilon}}}{2^{12}}\right\}$
10. If there do not exist κ_1, κ_2 that satisfy the following, then proceed to Algorithm 4.2
    (a) $\kappa_2 \geq \kappa_1 \geq 4\sqrt{\widetilde{\Upsilon}}$ and $\kappa_2 - \kappa_1 \leq \sqrt{\widetilde{\Upsilon}}$, or $\kappa_1 = \kappa_2 < 4\sqrt{\widetilde{\Upsilon}}$
    (b) For $M = [\kappa_1, \kappa_2] \cap SelectedBs$, we have $\sum_{i \in M} \left(Rank(L_M) - Rank(L_{M \setminus \{i\}})\right) \cdot \frac{\widetilde{OPT}}{2^i} \geq \min\{\frac{\widehat{OPT}}{2^{13}\sqrt{\widetilde{\Upsilon}}}, \frac{\widehat{OPT}}{2^{64}}\}$
11. Select an arbitrary pair κ_1, κ_2 satisfying the previous condition
12. For every element u revealed do
    (a) If $wt(u) = \frac{\widetilde{OPT}}{2^i}$, for some $i \in [\kappa_1, \kappa_2] \cap SelectedBs$ and $P \cup \{u\} \in \mathcal{I}$ then do $P \longleftarrow P \cup \{u\}$
13. **Return** P

PROPOSITION 4.1. *With probability at least 1/8, if $\Upsilon < 2^{64}$, then $OPT(P) \geq \frac{\widehat{OPT}}{2^{64}}$ and if $\max\{wt(u) \mid u \in \mathcal{U}\} > \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$, then $OPT(P) \geq \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$.*

Proof. By Line 1 with probability 1/2 the classical algorithm is employed (see Lindley [12] and Dynkin [7]). Assume that this is indeed the case. As a result, with probability at least $1/e$, a maximum weight element is returned.

PROPOSITION 4.2. *If* $\Upsilon > 2^{64}$ *and* $\max\{wt(u) \mid u \in \mathcal{U}\} < \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$, *then event* $\mathcal{A}$ *holds* $\widetilde{\Upsilon} = \Upsilon$ *and* $\widetilde{OPT} = \widehat{OPT}$, *with probability at least* 1/32.

Proof. Assume event $\mathcal{A}$ holds. Thus, we have that $\sum_{i>0} OPT(B_i^L) \geq \sum_{i \in CoreBs} OPT(B_i^L) \geq \frac{\widehat{OPT}}{16}$ and $Rank(L) \geq Rank(\mathcal{U})/3$. By definition $\sum_{i>0} OPT(B_i^L) \leq \widehat{OPT}$ and the fact that ξ is selected uniformly at random from $\{0,1,2,3\}$, with probability 1/4 we have that $\widetilde{OPT} = \widehat{OPT}$. As $L \subseteq \mathcal{U}$ we have that $Rank(L) < Rank(\mathcal{U})$. Since event $\mathcal{A}$ holds, by the definition of $\Upsilon, \widetilde{\Upsilon}$ and the fact that η is selected independently and uniformly at random from $\{0,1,2\}$, implies that with probability 1/3 we have that $\widetilde{\Upsilon} = \Upsilon$. Finally, since Theorem 3.1 asserts that the probability that event $\mathcal{A}$ holds is at least $\frac{13}{16}$, we conclude the proposition.

From here on we assume the following assumption holds unless explicitly stated otherwise. Hence we do not state that the assumption holds in any of the succeeding propositions, lemmatta and theorems.

Assumption $\widetilde{\Upsilon} = \Upsilon > 2^{64}$, $\max\{wt(u) \mid u \in \mathcal{U}\} < \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$, $\widetilde{OPT} = \widehat{OPT}$ and event $\mathcal{A}$ holds.

Now since $\widetilde{OPT} = \widehat{OPT}$, we have that $B_i^L = L_i$ for every i. So from here on, excluding in the description of Algorithm 4.2, which appears further on, we shall use B_i^Ls instead of L_is, $\widehat{OPT}$ instead of $\widetilde{OPT}$ and Υ instead of $\widetilde{\Upsilon}$.

PROPOSITION 4.3. $CoreBs \subseteq SelectedBs \subseteq GoodBs$.

Proof. By definition for every $i \in CoreBs$ we have that $i \in [\lceil \frac{\log \Upsilon}{2} \rceil + 11, 2\Upsilon]$ and $Rank(B_i) \geq \max\{\frac{2^{i-8}}{\Upsilon}, \frac{\sqrt{\Upsilon}}{2^8}\}$. Since event $\mathcal{A}$ holds, for every $i \in CoreBs$ we have that $Rank(B_i^L) \geq \max\{\frac{2^{i-8}}{\Upsilon}, \frac{\sqrt{\Upsilon}}{2^8}\}/3$. Hence by Line 9 for every $i \in CoreBs$ we have that $i \in SelectedBs$.

Since $\max\{wt(u) \mid u \in \mathcal{U}\} < \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$, we get that $i \geq \lceil \frac{\log \Upsilon}{2} \rceil + 11$ for every $i \in SelectedBs$ By Line 9 for every $i \in SelectedBs$ we have that $i \leq 2\Upsilon$ and $Rank(B_i^L) \geq \max\{\frac{2^{i-12}}{\Upsilon}, \frac{\sqrt{\Upsilon}}{2^{12}}\}$. Hence by Line 9 for every $i \in SelectedBs$ we have that $i \in [\lceil \frac{\log \Upsilon}{2} \rceil + 11, 2\Upsilon]$ and $Rank(B_i) \geq \max\{\frac{2^{i-16}}{\Upsilon}, \frac{\sqrt{\Upsilon}}{2^{16}}\}$. Thus, by the definition of $GoodBs$, for every $i \in SelectedBs$ we have that $i \in GoodBs$.

PROPOSITION 4.4. *Let* $M = [\kappa_1, \kappa_2] \cap SelectedBs$. *If Line 12 was reached, then* $Rank(P) = Rank(B_M^R)$.

Proof. Algorithm 4.1 adds only elements from B_M^R to P and therefore $Rank(P) \leq Rank(B_M^R)$. Assume for the sake of contradiction that $Rank(P) < Rank(B_M^R)$. Let $Z \subseteq B_M^R$ be a basis of B_M^R. Since $Rank(P) < Rank(B_M^R) = |Z|$, by the properties of matroids there exists $z \in Z \setminus P$ such that $\{z\} \cup P$ is independent. Yet this cannot be true since it implies that Algorithm 4.1 would have added z to P.

LEMMA 4.1. *If Line 12 was reached, and* $\kappa_2 = \kappa_1 < 4\sqrt{\Upsilon}$, *then* $OPT(P) \geq \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$.

Proof. Set $i = \kappa_1$. Since the condition of Line 10 is satisfied, we know that $OPT(B_i^L) \geq \frac{\widehat{OPT}}{2^{13}\sqrt{\Upsilon}}$. Hence $Rank(B_i^L) \geq \frac{2^i}{2^{13}\sqrt{\Upsilon}}$. Now as $i \in SelectedBs$ we have that $i \geq \lceil \frac{\log \Upsilon}{2} \rceil + 11$ and therefore $Rank(B_i^L) \geq 4\sqrt{\Upsilon}$. We also have $Rank(B_i^R) \geq Rank(B_i^L) - 4 \cdot \log(\Upsilon) \cdot \sqrt{Rank(B_i)} \geq Rank(B_i^L)/2$, where the second inequality is because event $\mathcal{A}$ holds, $Rank(B_i^L) \geq 4\sqrt{\Upsilon}$ and $\Upsilon \geq 2^{64}$. Consequently $Rank(B_i^R) \geq Rank(B_i^L)/2$. Proposition 4.4 asserts that $Rank(P) = Rank(B_M^R) = Rank(B_i^R)$ and hence $OPT(P) = OPT(B_i^R) \geq OPT(B_i^L)/2$.

THEOREM 4.2. *If Line 12 was reached, and* $\kappa_2 \geq \kappa_1 \geq 4\sqrt{\Upsilon}$, *then* $OPT(P) \geq \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$.

Proof. Let $M = [\kappa_1, \kappa_2] \cap SelectedBs$. Observe that for every $i \in M$ we have that $Rank(P \cap B_i^R) \geq Rank(P) - Rank(B_{M\setminus\{i\}}^R)$ and hence

$$OPT(P) = \sum_{i \in M} Rank(P \cap B_i^R) \cdot \frac{\widehat{OPT}}{2^i} \geq \sum_{i \in M} \Big(Rank(P) - Rank(B_{M\setminus\{i\}}^R)\Big) \cdot \frac{\widehat{OPT}}{2^i}.$$

Proposition 4.4 asserts that $Rank(P) = Rank(B_M^R)$ and thus,

(4.1)
$$OPT(P) \geq \sum_{i \in M} \Big(Rank(B_M^R) - Rank(B_{M\setminus\{i\}}^R)\Big) \cdot \frac{\widehat{OPT}}{2^i}.$$

Since event $\mathcal{A}$ holds and $SelectedBs \subseteq GoodBs$, by Proposition 4.3, we have that

$$Rank(B_M^R) - Rank(B_{M\setminus\{i\}}^R) \geq Rank(B_M^L) - Rank(B_{M\setminus\{i\}}^L) - 16 \cdot \Upsilon \cdot \sqrt{Rank(B_M)}$$

and therefore together with (4.1)

(4.2)
$$OPT(P) \geq \sum_{i \in M} \Big(Rank(B_M^L) - Rank(B_{M\setminus\{i\}}^L)\Big) \frac{\widehat{OPT}}{2^i} - \sum_{i \in M} 16 \cdot \Upsilon \cdot \sqrt{Rank(B_M)} \cdot \frac{\widehat{OPT}}{2^i}.$$

According to Line 10b
(4.3)

$$\sum_{i\in M}\left(Rank(B^L_M)-Rank(B^L_{M\setminus\{i\}})\right)\cdot\frac{\widehat{OPT}}{2^i}\geq\frac{\widehat{OPT}}{2^{13}\sqrt{\Upsilon}}$$

By Proposition 3.2 and Proposition 3.3

$$\begin{array}{c}\sum_{i\in M}16\cdot\Upsilon\cdot\sqrt{Rank(B_M)}\cdot\frac{\widehat{OPT}}{2^i}\leq\\ \sum_{i\in M}16\cdot\Upsilon\cdot Rank(B_i)^{\frac{3}{4}}\cdot\frac{\widehat{OPT}}{2^i}\leq\frac{\widehat{OPT}}{2^{20}\sqrt{\Upsilon}}.\end{array}\tag{4.4}$$

The theorem follows from (4.2), (4.3) and (4.4).

Proof of Theorem 4.1 Recall that $\widehat{OPT}\geq OPT$ by definition. By Proposition 4.1, if $\Upsilon<2^{64}$ or $\max\{wt(u)\mid u\in\mathcal{U}\}>\frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$, then $OPT(P)\geq\min\{\frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}},\frac{\widehat{OPT}}{2^{64}}\}$ with probability at least 1/8. Assume that $\Upsilon>2^{64}$ and $\max\{wt(u)\mid u\in\mathcal{U}\}\leq\frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$. By Proposition 4.2 event $\mathcal{A}$ holds $\widetilde{\Upsilon}=\Upsilon$ and $\widehat{\widetilde{OPT}}=\widehat{OPT}$, with probability at least 1/32. With probability 1/2, Algorithm 4.1 does not employ the classical algorithm. Hence it reaches Line 10. By Lemma 4.2 and Theorem 4.2 if the condition in Line 10 does not hold, then $OPT(P)\geq\frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$. If the condition in Line 10 does hold, then P is the set returned by Algorithm 4.2, which appears in the following subsection. By Theorem 4.3, which also appears in the following subsection, we have that $OPT(P)\geq\frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$.

4.1 The Protection Algorithm The pseudo-code for Algorithm 4.2 (the protection algorithm) appears further on and is essential for understanding what is written in this text. Note that all the matroid related operations used in Algorithm 4.2 can be done by using only an oracle to the matroid.

Algorithm 4.2 starts by creating a partition of *SelectedBs* into sets (Tub_js) bounded in a range of size at most $\sqrt{\Upsilon}$ (recall that we assume that $\widetilde{\Upsilon}=\Upsilon$). Elements will be selected from either buckets whose index is in Tub_js, where j is odd, or from buckets in Tub_j's, where j is even, but not both. The decision on which of the two options is selected is made on Line 3.

The importance of the Tub_js is that they replace the magic oracle described in Section 2. Specifically, the fact that Algorithm 4.2 was employed implies the following. For each Tub_j if $OPT(L_{Tub_j})$ is sufficiently large there exists i such that $OPT(L_i)$ is sufficiently large and the following holds: $Rank(L_{Tub_{t(i)}})-Rank(L_{Tub_{t(i)}\setminus\{i\}})<\frac{Rank(L_i)}{2}$ (on Line 4c), where $t(i)$ is the index of the Tub that contains i as defined in the pseudo-code. Thus, $L_{Tub_{t(i)}\setminus\{i\}}$ could have been used in order to protect L_i in the same manner as was done with a basis given by the magic oracle described in Section 2. Regretfully, $L_{Tub_{t(i)}\setminus\{i\}}$ is only known after all the elements of L_i are revealed. However, all is not lost since the concentration results implies that $L_{Tub_{t(i)}\setminus\{i\}}$ can be used to protect R_i.

The exact manner in which the protection is implemented can be seen in Line 6a. Observe that each Tub is used to protect exactly one *Bucket*. That is only elements from such protected buckets are selected and elements from other buckets are ignored. Thus ideally we would have wanted the Tubs to have a range that is as small as possible. Nonetheless the range of each Tub is $\Theta(\sqrt{\widetilde{\Upsilon}})$, since having a smaller range would increase the competitive ratio of the simple algorithm.

Note that using protection may cause damage by preventing the selection of elements from buckets of lower element weight. In Proposition 4.7 we show that using this damage is bounded damage because of the following. Every protected bucket has a rank that is significantly larger than that of the union of all elements used to protect buckets of larger element weight.

Algorithm 4.2.

1. $S\longleftarrow\emptyset$, $\nu\longleftarrow 0$
2. For every $j\in[\sqrt{\widetilde{\Upsilon}}]$ do
   (a) $Tub_j\longleftarrow\left[\sqrt{\widetilde{\Upsilon}}(3+j),\sqrt{\widetilde{\Upsilon}}(4+j)\right)\cap SelectedBs$
   (b) $t(i)\longleftarrow j$ for every $i\in\left[\sqrt{\widetilde{\Upsilon}}(3+j),\sqrt{\widetilde{\Upsilon}}(4+j)\right)$
3. If $\sum_{j\in[\sqrt{\widetilde{\Upsilon}}]\cap\mathbb{N}_1}\sum_{k\in Tub_j}OPT(L_k)\geq\sum_{j\in[\sqrt{\widetilde{\Upsilon}}]\cap\mathbb{N}_0}\sum_{k\in Tub_j}OPT(L_k)$, then do $\nu\longleftarrow 1$
4. For every $j\in[\sqrt{\widetilde{\Upsilon}}]\cap\mathbb{N}_\nu$, if there exist $i\in Tub_j$ that satisfy the following pick one of them arbitrarily and do $S\longleftarrow S\cup\{i\}$
   (a) $\sum_{k\in Tub_j}OPT(L_k)\geq\frac{\widehat{OPT}}{2^8\sqrt{\widetilde{\Upsilon}}}$
   (b) $OPT(L_i)\geq\frac{\sum_{k\in Tub_j}OPT(L_k)}{4|Tub_j|}$
   (c) $Rank(L_{Tub_j})-Rank(L_{Tub_j\setminus\{i\}})<\frac{Rank(L_i)}{2}$
5. For every $i\in S$ do
   (a) $Heavy_{t(i)}\longleftarrow\bigcup_{k<i,k\in S,j\in Tub_{t(k)}\setminus\{k\}}L_j$
6. For every element u viewed, if $wt(u)=\frac{\widehat{OPT}}{2^i}$, for some $i\in S$, then do

(a) If $Rank(\{u\} \cup P \cup Heavy_{t(i)}) > Rank(P \cup Heavy_{t(i)})$ then do $P \longleftarrow P \cup \{u\}$

7. Return P

THEOREM 4.3. *Let P be the set returned by Algorithm 4.2, then $OPT(P) \geq \frac{\widehat{OPT}}{2^{16}\sqrt{\Upsilon}}$.*

Proof. In Lemma 4.2, which we prove further on, we show that for every $i \in S$ we have that $Rank(B_i^R \cap P)$ is at least

$$Rank(B_i^L) - \left(Rank(B^L_{Tub_{t(i)}}) - Rank(B^L_{Tub_{t(i)}\setminus\{i\}})\right) - \left(2Rank(Heavy_{t(i)}) + 2\sum_{k\in S,k<i} Rank(B_k^R)\right) - 16\cdot\Upsilon\cdot\sqrt{Rank(B^L_{Tub_{t(i)}})}.$$

Since the condition of Line 4c is satisfied, for every $i \in S$ we have that

$$Rank(B^L_{Tub_{t(i)}}) - Rank(B^L_{Tub_{t(i)}\setminus\{i\}}) < \frac{Rank(B_i^L)}{2}.$$

Proposition 4.7, which we prove further on, asserts that

$$2Rank(Heavy_{t(i)}) + 2\sum_{k\in S,k<i} Rank(B_k^R) \leq \frac{Rank(B_i^L)}{32}$$

for every $i \in S$. Now by the fact that $SelectedBs \subseteq GoodBs$ by Proposition 4.3, Proposition 3.2 and Proposition 3.3 imply that

$$\sum_{i\in S} 16\cdot\Upsilon\cdot\sqrt{Rank(B_{t(i)})}\cdot\frac{\widehat{OPT}}{2^i} \leq \sum_{i\in S} 16\cdot\Upsilon\cdot Rank(B_i)^{\frac{3}{4}}\cdot\frac{\widehat{OPT}}{2^i} \leq \frac{\widehat{OPT}}{2^{20}\sqrt{\Upsilon}}.$$

Thus, $OPT(P)$ is at least

$$\sum_{i\in S} OPT(B_i^R \cap P) \geq \sum_{i\in S}\left(Rank(B_i^L) - \frac{Rank(B_i^L)}{2} - \frac{Rank(B_i^L)}{32}\right)\cdot\frac{\widehat{OPT}}{2^i} - \frac{\widehat{OPT}}{2^{20}\sqrt{\Upsilon}}.$$

Finally by Proposition 4.6, which we prove further on, we have that $\sum_{i\in S} OPT(B_i^L) \geq \frac{\widehat{OPT}}{2^8\sqrt{\Upsilon}}$ and the theorem follows.

PROPOSITION 4.5. *Let $i \in S$, then*

$$Rank(P \cup Heavy_{t(i)} \cup B_i^R \cup \textstyle\bigcup_{k\in S,k<i} B_k^R) = Rank(P \cup Heavy_{t(i)} \cup \textstyle\bigcup_{k\in S,k<i} B_k^R).$$

Proof. Assume for the sake of contradiction that the proposition does not hold. Let Z be a basis and subset of $P \cup Heavy_{t(i)} \cup B_i^R \cup \bigcup_{k\in S,k<i} B_k^R$. Since

$$Rank(P \cup Heavy_{t(i)} \cup B_i^R \cup \textstyle\bigcup_{k\in S,k<i} B_k^R) > Rank(P \cup Heavy_{t(i)} \cup \textstyle\bigcup_{k\in S,k<i} B_k^R),$$

there exists $z \in Z \cap B_i^R$ such that $\{z\} \cup P \cup Heavy_{t(i)} \cup \bigcup_{k\in S,k<i} B_k^R$ is independent. Yet this cannot happen since z satisfies the condition on Line 6a of Algorithm 4.2 and hence $z \in P$.

LEMMA 4.2. *Let $i \in S$, then $Rank(B_i^R \cap P)$ is at least*

$$Rank(B_i^L) - \left(Rank(B^L_{Tub_{t(i)}}) - Rank(B^L_{Tub_{t(i)}\setminus\{i\}})\right) - 2Rank(Heavy_{t(i)}) - 2\textstyle\sum_{k\in S,k<i} Rank(B_k^R) - 16\cdot\Upsilon\cdot\sqrt{Rank(B^L_{Tub_{t(i)}})}.$$

Proof. Let $Y = Heavy_{t(i)} \cup \bigcup_{k\in S,k<i} B_k^R$. Since Proposition 4.5 asserts that $Rank(P \cup Y \cup B_i^R) = Rank(P \cup Y)$ we get that

$$Rank(P \cup Y \cup B^L_{Tub_{t(i)}\setminus\{i\}}) = Rank\left((P \cup Y \cup B_i^R) \cup (B^L_{Tub_{t(i)}\setminus\{i\}} \cup B_i^R)\right).$$

Hence

$$Rank(P \cup Y \cup B^L_{Tub_{t(i)}\setminus\{i\}}) \leq Rank(P \cup Y \cup B_i^R) + Rank(B^L_{Tub_{t(i)}\setminus\{i\}} \cup B_i^R) - Rank(B_i^R)$$

Thus, again by Proposition 4.5,

$$Rank(P \cup Y \cup B^L_{Tub_{t(i)}\setminus\{i\}}) \leq Rank(P) + Rank(Y) + Rank(B^L_{Tub_{t(i)}\setminus\{i\}} \cup B_i^R) - Rank(B_i^R) \quad (4.5)$$

We also have

$$Rank(P \cup Y \cup B^L_{Tub_{t(i)}\setminus\{i\}}) \geq Rank\left((P \setminus (B_i^R \cup Y)) \cup B^L_{Tub_{t(i)}\setminus\{i\}}\right)$$

As a result of the condition in Line 5a, we get that

$$Rank(P \cup Y \cup B^L_{Tub_{t(i)}\setminus\{i\}}) \geq Rank(P \setminus (B_i^R \cup Y)) + Rank(B^L_{Tub_{t(i)}\setminus\{i\}}).$$

Therefore,

$$Rank(P \cup Y \cup B^L_{Tub_{t(i)}\setminus\{i\}}) \geq Rank(P) - Rank(Y) - Rank(P \cap B_i^R) + Rank(B^L_{Tub_{t(i)}\setminus\{i\}}) \quad (4.6)$$

By Equation 4.5 and Equation 4.6 we get that

$$Rank(P) + Rank(Y) + Rank(B^L_{Tub_{t(i)}\setminus\{i\}} \cup B_i^R) - Rank(B_i^R) \geq Rank(P) - Rank(Y) - Rank(P \cap B_i^R) + Rank(B^L_{Tub_{t(i)}\setminus\{i\}})$$

The lemma follows since event $\mathcal{A}$ holds.

LEMMA 4.3. *Let* $j \in [\sqrt{\Upsilon}] \cap \mathbb{N}_\nu$ *and* $\sum_{i \in Tub_j} OPT(B_j^L) \geq \frac{\widehat{OPT}}{2^8\sqrt{\Upsilon}}$, *then* $Tub_j \cap S \neq \emptyset$.

Proof. Let T be the set of all $i \in Tub_j$ such that $OPT(B_i^L) > \frac{\sum_{k \in Tub_j} OPT(B_k^L)}{4|Tub_j|}$. Assume for the sake of contradiction that $Tub_j \cap S = \emptyset$. Since for every $i \in T$ the conditions in Line 4a and Line 4b hold, we infer that for every $i \in T$ the condition in Line 4c does not hold. Thus, for every $i \in T$ we have that $Rank(B^L_{Tub_j}) - Rank(B^L_{Tub_j \setminus \{i\}}) \geq Rank(B_i^L)/2$. Consequently,

(4.7) $$\sum_{i \in T} \left(Rank(B^L_{Tub_j}) - Rank(B^L_{Tub_j \setminus \{i\}}) \right) \cdot \frac{\widehat{OPT}}{2^i} > \sum_{i \in T} \frac{OPT(B_i^L)}{2}.$$

Observe that

$$|Tub_j| \cdot \frac{\sum_{i \in Tub_j \setminus T} OPT(B_i^L) \leq \sum_{k \in Tub_j} OPT(B_k^L)}{4|Tub_j|} \leq \frac{\sum_{k \in Tub_j} OPT(B_k^L)}{4}$$

and therefore

(4.8) $$\sum_{i \in T} OPT(B_i^L) \geq \sum_{i \in Tub_j} OPT(B_i^L) - \sum_{i \in Tub_j \setminus T} OPT(B_i^L) \geq \frac{3 \cdot \sum_{k \in Tub_j} OPT(B_k^L)}{4}.$$

As a direct result of (4.7) and (4.8) and the fact that $T \subseteq Tub_j$ we get that

$$\sum_{i \in Tub_j} \left(Rank(B^L_{Tub_j}) - Rank(B^L_{Tub_j \setminus \{i\}}) \right) \cdot \frac{\widehat{OPT}}{2^i} > \frac{3 \cdot \sum_{k \in Tub_j} OPT(B_k^L)}{4 \cdot 2} > \frac{\widehat{OPT}}{2^{13}\sqrt{\Upsilon}}.$$

This implies that the condition in Line 10b of Algorithm 4.1 is not satisfied, which implies that Algorithm 4.2 was not executed. Hence a contradiction.

PROPOSITION 4.6. $\sum_{i \in S} OPT(B_i^L) \geq \frac{\widehat{OPT}}{2^8\sqrt{\Upsilon}}$.

Proof. Let T be the set of all j such that $Tub_j \cap S \neq \emptyset$. By Line 4b and Lemma 4.3 we have that

(4.9) $$\sum_{i \in S} OPT(B_i^L) \geq \sum_{i \in S} \frac{\sum_{k \in Tub_{t(i)}} OPT(B_k^L)}{4|Tub_{t(i)}|} \geq \frac{1}{4\sqrt{\Upsilon}} \sum_{i \in S} \sum_{k \in Tub_{t(i)}} OPT(B_k^L),$$

where the second inequality is because $|Tub_{t(i)}| \leq \sqrt{\Upsilon}$ for every $i \in S$, by Line 2a. Observe that by definition $\sum_{i \in S} \sum_{k \in Tub_{t(i)}} OPT(B_k^L)$ is at least $\sum_{i \in SelectedBs} OPT(B_i^L)$ minus the following

(4.10) $$\sum_{i \in [4\sqrt{\Upsilon}]} OPT(B_i^L) + \sum_{j \in [\sqrt{\Upsilon}] \cap \mathbb{N}_{1-\nu}} \sum_{k \in Tub_j} OPT(B_k^L) + \sum_{j \in ([\sqrt{\Upsilon}] \cap \mathbb{N}_\nu) \setminus T} \sum_{k \in Tub_j} OPT(B_k^L)$$

where the last inequality is by Proposition 3.1. By Line 3 we have that

$$\sum_{j \in [\sqrt{\Upsilon}] \cap \mathbb{N}_{1-\nu}} \sum_{k \in Tub_j} OPT(B_k^L) \leq \sum_{k \in SelectedBs} \frac{OPT(B_k^L)}{2}.$$

By Line 4a we have that $\sum_{k \in Tub_j} OPT(B_k^L) < \frac{\widehat{OPT}}{2^8\sqrt{\Upsilon}}$ for every $j \in ([\sqrt{\Upsilon}] \cap \mathbb{N}_\nu) \setminus T$ and hence

$$\sum_{j \in ([\sqrt{\Upsilon}] \cap \mathbb{N}_\nu) \setminus T} \sum_{k \in Tub_j} OPT(B_k^L) \leq \sqrt{\Upsilon} \frac{\widehat{OPT}}{2^8\sqrt{\Upsilon}} \leq \frac{\widehat{OPT}}{2^8}.$$

As the conditions of Line 10 of Algorithm 4.1 hold for every $i \in [4\sqrt{\Upsilon}]$ we have that $OPT(B_i^L) < \frac{\widehat{OPT}}{2^{13}\sqrt{\Upsilon}}$ and therefore

$$\sum_{i \in [4\sqrt{\Upsilon}]} OPT(B_i^L) \leq \frac{\widehat{OPT}}{2^8}.$$

Together with (4.9) and (4.10) we get

$$\sum_{i \in S} \sum_{k \in Tub_{t(i)}} OPT(B_k^L) \geq \sum_{i \in SelectedBs} OPT(B_i^L) - \sum_{i \in SelectedBs} \frac{OPT(B_i^L)}{2} - \frac{\widehat{OPT}}{2^8} - \frac{\widehat{OPT}}{2^8}$$

The proposition follows, since event $\mathcal{A}$ holds and hence by Proposition 4.3 we have that $CoreBs \subseteq SelectedBs \subseteq GoodBs$ and thus

$$\sum_{i \in S} \sum_{k \in Tub_{t(i)}} OPT(B_k^L) \geq \sum_{i \in SelectedBs} \frac{OPT(B_i^L)}{4} \geq \sum_{i \in CoreBs} \frac{OPT(B_i)}{4} \geq \frac{\widehat{OPT}}{16}$$

and together with Equation 4.9 we conclude the Proposition.

PROPOSITION 4.7. *Let* $i \in S$, *then* $Rank(Heavy_{t(i)}) + \sum_{k \in S, k<i} Rank(B_k^R) \leq Rank(B_i^L)/2^8$.

Proof. Since the conditions of Line 4a and Line 4b, hold we know that $OPT(B_i^L) \geq \frac{\widehat{OPT}}{4|Tub_{t(i)}|2^8\sqrt{\Upsilon}}$. By Line 2a the maximum weight of an element in $Tub_{t(i)}$ is $\frac{\widehat{OPT}}{2^{\sqrt{\Upsilon}(4+t(i))}}$ and hence $Rank(B_i^L) \geq \frac{2^{\sqrt{\Upsilon}(4+t(i))}}{4 \cdot 2^8 \Upsilon}$, where the Υ is because $|Tub_{t(i)}| \leq \sqrt{\Upsilon}$, by Line 2a. According to definition $OPT(Heavy_{t(i)} \cup \bigcup_{k \in S, k<i} B_k^R) \leq \widehat{OPT}$. By Line 5a the minimum weight of an element in $Heavy_{t(i)} \cup \bigcup_{k \in S, k<i} B_k^R$ is $\frac{\widehat{OPT}}{2^{\sqrt{\Upsilon}(5+t(i)-2)}}$ and therefore $Rank(Heavy_{t(i)} \cup \bigcup_{k \in S, k<i} B_k^R) \leq 2^{\sqrt{\Upsilon}(3+t(i))}$. Consequently,

$$\frac{Rank(Heavy_{t(i)} \cup \bigcup_{k \in S, k<i} B_k^R)}{Rank(B_i^L)} \leq \frac{4 \cdot 2^8 \Upsilon}{2^{\sqrt{\Upsilon}}}.$$

The proposition follows since $\Upsilon \geq 2^{64}$.

Acknowledgements

We would like to thank Ran El-Yaniv, Eldar Fischer and Ilan Newman for useful discussions. We would also like to thank Oren Ben-Zwi, Trevor Fenner, Sven Helmer and Alex Popa for helping to edit the paper.

References

[1] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley, 2000.

[2] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. A knapsack secretary problem with applications. In *APPROX/RANDOM*, pages 16–28, 2007.

[3] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exch.*, 7:7:1–7:11, June 2008.

[4] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *SODA*, pages 434–443, 2007.

[5] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. In *APPROX/RANDOM*, pages 39–52, 2010.

[6] Nedialko B. Dimitrov and C. Greg Plaxton. Competitive weighted matching in transversal matroids. In *ICALP*, pages 397–408, 2008.

[7] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Sov. Math. Dokl.*, 4, 1963.

[8] P. R. Freeman. The secretary problem and its extensions: a review. *Internat. Statist. Rev.*, 51(2):189–206, 1983.

[9] Sungjin Im and Yajun Wang. Secretary problems: Laminar matroid and interval scheduling. In *SODA*, pages 1265–1274, 2005.

[10] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *SODA*, pages 630–631, 2005.

[11] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *ICALP*, pages 508–520, 2009.

[12] D. V. Lindley. Dynamic programming and decision theory. *Applied Statistics*, 10:39–51, 1961.

[13] José A. Soto. Matroid secretary problem in the random assignment model. In *SODA*, pages 1275–1284, 2011.