2014 IEEE Annual Symposium on Foundations of Computer Science

O(log log *rank*) Competitive Ratio for the Matroid Secretary Problem

Oded Lachish
Department of Computer Science and Information Systems
Birkbeck, University of London
London, United Kingdom
Email: oded@dcs.bbk.ac.uk

***Abstract*—In the *Matroid Secretary Problem* (MSP), the elements of the ground set of a Matroid are revealed on-line one by one, each together with its value. An algorithm for the Matroid Secretary Problem is called *Matroid-Unknown* if, at every stage of its execution, it only knows (i) the elements that have been revealed so far and their values and (ii) an oracle for testing whether or not a subset the elements that have been revealed so far forms an independent set. An algorithm is called *Known-Cardinality* if it knows (i), (ii) and also knows from the start the cardinality n of the ground set of the Matroid.**

We present here a Known-Cardinality algorithm with a *competitive-ratio* of order log log the rank of the Matroid. The prior known results for a Known-Cardinality algorithm are a competitive-ratio of log the rank of the Matroid, by Babaioff *et al.* (2007), and a competitive-ratio of square root of log the rank of the Matroid, by Chakraborty and Lachish (2012).

***Keywords*-Competitive-Ratio; Matroid; Secretary;**

I. Introduction

The *Matroid Secretary Problem* is a generalization of the *Classical Secretary Problem*, whose origins seem to still be a source of dispute. One of the first papers on the subject [12], by Dynkin, dates back to 1963. Lindley [21] and Dynkin [12] each presented an algorithm that achieves a *competitive-ratio* of e, which is the best possible. See [14] for more information about results preceding 1983.

In 2007, Babaioff *et al.* [4] established a connection between the Matroid Secretary Problem and *mechanism design*. This is probably the cause of an increase of interest in generalizations of the *Classical Secretary Problem* and specifically the Matroid Secretary Problem.

In the Matroid Secretary Problem, we are given a Matroid $\{U, \mathcal{I}\}$ and a value function assigning non-negative values to the Matroid elements. The elements of the Matroid are revealed in an on-line fashion according to an unknown order selected uniformly at random. The value of each element is unknown until it is revealed. Immediately after each element is revealed, if the element together with the elements already selected does not form an independent set, then that element cannot be selected; however, if it does, then an irrevocable decision must be made whether or not to select the element. That is, if the element is selected, it will stay selected until the end of the process and likewise if it is not. The goal is to design an algorithm for this problem wit ha small competitive-ratio, that is the ratio between the maximum sum of values of an independent set and the expected sum of values of the independent set returned by the algorithm.

An algorithm for the *Matroid Secretary Problem* (MSP) is called *Matroid-Unknown* if, at every stage of its execution, it only knows (i) the elements that have been revealed so far and their values and (ii) an oracle for testing whether or not a subset the elements that have been revealed so far forms an independent set. An algorithm is called *Known-Cardinality* if it knows (i), (ii) and also knows from the start the cardinality n of the ground set of the Matroid. An algorithm is called *Matroid-Known*, if it knows, from the start, everything about the Matroid except for the values of the elements. These, as mentioned above, are revealed to the algorithm as each element is revealed.

Related Work: Our work follows the path initiated by Babaioff *et al.* in [4]. There they formalized the Matroid Secretary Problem and presented a Known-Cardinality algorithm with a competitive-ratio of $\log \rho$. This line of work was continued in [8], where an algorithm with a competitive-ratio of $O(\sqrt{\log \rho})$ was presented. In Babaioff *et al.* [4] (2007), it was conjectured that a constant competitive-ratio is achievable. The best known result for a *Matroid-Unknown* algorithm, implied by the works of Gharan and Vondrák [15] and Chakraborty and Lachish [8] (2012): for every fixed $\epsilon > 0$, there exists a Matroid-Unknown algorithm with a competitive-ratio of $O(\epsilon^{-1}(\sqrt{\log \rho}) \log^{1+\epsilon} n)$. Gharan and Vondrák showed that a lower bound of $\Omega(\frac{\log n}{\log \log n})$ on the competitive-ratio holds in this case.

Another line of work towards resolving the Matroid Secretary Problem is the study of the Secretary Problem for specific families of Matroids. Most of the results of this type are for Matroid-Known algorithms and all achieve a constant competitive-ratio. Among the specific families of Matroids studied are *Graphic Matroids* [4], *Uniform/Partition Matroids* [3], [19], *Transversal Matroids* [9], [20], *Regular and Decomposable Matroids* [11] and *Laminar Matroids* [17]. For surveys that also include other variants of the Matroid Secretary Problem see [10], [18], [23].

There are also results for other generalizations of the Classical Secretary Problem, including the *Knapsack Secretary Problem* [3], *Secretary Problems with Convex Costs* [5], *Submodular Secretary Problems* [6], [13], [16] and *Secretary problems via linear programming* [7].

0272-5428/14 $31.00 © 2014 IEEE
DOI 10.1109/FOCS.2014.42

Main result: We present here a Known-Cardinality algorithm with a competitive-ratio of $O(\log\log\rho)$. The algorithm is also Order-Oblivious as defined by Azar *et al.* [2]). According to [15], this implies that, for every fixed $\epsilon > 0$, there exists a Matroid-Unknown algorithm with a competitive-ratio of $O(\epsilon^{-1}(\log\log\rho)\log^{1+\epsilon} n)$. We believe, but do not prove explicitly, that our algorithm is also Order-Oblivious as in Definition 1 of [2], and hence, by Theorem 1 of [2], this would imply that there exists a Single Sample Prophet Inequality for Matroids with a competitive-ratio of $O(\log\log\rho)$.

High level description of result and its relation to previous work.: As in [4] and [8], here we also partition the elements into sets which we call *buckets*. This is done by rounding down the value of each element to the largest possible power of two and then, for every power of two, defining a bucket to be the set of all elements with that value. Obviously, the only impact this has on the order of the competitive-ratio achieved is a constant factor of at most 2.

We call our algorithm the Main Algorithm. It has three consecutive stages: *Gathering stage*, *Preprocessing stage* and *Selection stage*. In the Gathering stage it waits, without selecting any elements, until about half of the elements of the matroid are revealed. The set F that consists of all the elements revealed during the Gathering stage is the input to the Preprocessing stage. In the Preprocessing stage, on out of the following three types of output is computed: (i) a non negative value, (ii) a set of bucket indices, or (iii) a critical tuple. Given the output of the Preprocessing stage, before any element is revealed the Main Algorithm chooses one of the following algorithms: the *Threshold Algorithm*, the *Simple Algorithm* or the *Gap Algorithm*. Then, after each one of the remaining elements is revealed, the decision whether to select the element is made by the chosen algorithm using the input received from the Preprocessing stage and the set of all the elements already revealed. Once all the elements have been revealed the set of selected elements is returned.

The Threshold Algorithm is chosen when the output to the Preprocessing stage is a non-negative value, which happens with probability half regardless of the contents of the set F. Given this input, the Threshold Algorithm, as in the algorithm for the Classical Secretary Problem, selects only the first element that has at least the given value. The Simple Algorithm is chosen when the output of Preprocessing stage is a set of bucket indices. The Simple Algorithm selects an element if it is in one of the buckets determined by the set of indices and if it is independent of all previously selected elements. This specific algorithm was also used in [8].

The Gap Algorithm is chosen when the output of Selection stage is a critical tuple, which we define further on. The Gap Algorithm works as follows: every element revealed is required to have one of a specific set of values and satisfy two conditions in order to be selected: it satisfies the first condition if it is in the closure of a specific subset of elements of F; it satisfies the second condition if it is not in the closure of the union of the set of elements already selected and a specific subset of elements of F (which is different than the one used in the first condition).

The proof that the Main Algorithm achieves the claimed competitive-ratio consists of the following parts: a guarantee on the output of the Simple Algorithm as a function of the input and $U \setminus F$, where U is the ground set of the matroid; a guarantee on the output of the Gap Algorithm as a function of the input and $U \setminus F$; a combination of a new structural result for matroids and probabilistic inequalities that imply that if the matroid does not have an element with a large value, then it is possible to compute an input for either the Simple Algorithm or the Gap Algorithm that, with high probability, ensures that the output set has a high value. This guarantees the claimed competitive-ratio, since the case when the matroid has an element with a large value is dealt with by the Threshold Algorithm.

One of the probabilistic inequalities we use is the key ingredient for ensuring that the Gap Algorithm works. It is not clear whether it is possible to prove such an inequality using only the techniques in [8], because they rely strongly on symmetry.

The paper is organized as follows:: Section II contains the preliminaries; Section III presents Main Algorithm; Section IV is devoted to the Simple Algorithm and the Gap Algorithm; Section V contains the required concentrations; the structural trade-off result is proved in Section VI; and the main result appears in Section VII.

II. Preliminaries

All logarithms are to the base 2. We use $\mathbb{Z}$ to denote the set of all integers, $\mathbb{N}$ to denote the non-negative integers and $\mathbb{N}^+$ to denote the positive integers. We use $[\alpha]$ to denote $\{1, 2, \ldots, \lfloor\alpha\rfloor\}$ for any non-negative real α. We use $[\alpha, \beta]$ to denote $\{i \in \mathbb{Z} \mid \alpha \leq i \leq \beta\}$ and $(\alpha, \beta]$ to denote $\{i \in \mathbb{Z} \mid \alpha < i \leq \beta\}$, and so on. For every $j \in \mathbb{Z}$ and $I \subseteq \mathbb{Z}$, we define $j > I$ if and only if $j > i$ for every $i \in I$, and $j < I$ if and only if $j < i$ for every $i \in I$. For every $I, J \subseteq \mathbb{Z}$, we define $J > I$ if and only if $\min J > \max I$, and we say I and J are *comparable* if either $J > I$, or $I > J$ or $I = J$. We use $\mathrm{med}\,(f)$ to denote the *median* of a function f from a finite set to the non-negative reals. If there are two possible values for $\mathrm{med}\,(f)$ the smaller one is chosen.

We define $\beta(n, 1/2)$ to be a random variable whose value is the number of successes in n independent probability $1/2$ Bernoulli trials.

Observation 1: Let $A = \{a_1, a_2, \ldots, a_n\}$ and $W = \beta(n, 1/2)$; let $\pi : [n] \longrightarrow [n]$ be a permutation selected uniformly at random, and let $D = \{a_{\pi(i)} \mid i \in [W]\}$. For every $i \in [n]$, we have that $a_i \in D$ independently with probability $1/2$.

Proof: To prove the proposition we only need to show that for every $C \subseteq A$, we have $D = C$ with probability 2^{-n}. Fix C. There are $\binom{n}{|C|}$ subsets of A of size $|C|$. D is equally likely to be one of these subsets. Hence, the probability that $|D| = |C|$ is $\binom{n}{|C|} \cdot 2^{-n}$ and therefore the probability that $D = C$ is $\binom{n}{|C|} \cdot 2^{-n} / \binom{n}{|C|} = 2^{-n}$. ∎

A. Matroid definitions, notations and preliminary results

Definition 2: **[Matroid]** A **matroid** is an ordered pair $M = (U, \mathcal{I})$, where U is a set of **elements**, called the *ground set*, and $\mathcal{I}$ is a family of subsets of U that satisfies the following:

- If $I \in \mathcal{I}$ and $I' \subset I$, then $I' \in \mathcal{I}$
- If $I, I' \in \mathcal{I}$ and $|I'| < |I|$, then there exists $e \in I \setminus I'$ such that $I' \cup \{e\} \in \mathcal{I}$.

The sets in $\mathcal{I}$ are called **independent** sets and a maximal independent set is called a **basis**.

A *value function* over a Matroid $M = (U, \mathcal{I})$ is a mapping from the elements of U to the non-negative reals. Since we deal with a fixed Matroid and value function, we will always use $M = (U, \mathcal{I})$ for the Matroid. We set $n = |U|$ and, for every $e \in U$, we denote its value by $val(e)$.

Definition 3: **[rank and Closure]** For every $S \subseteq U$, let

- $\text{rank}(S) = \max\{|S'| \mid S' \in \mathcal{I} \text{ and } S' \subseteq S\}$ and
- $\text{Cl}(S) = \{e \in U \mid \text{rank}(S \cup \{e\}) = \text{rank}(S)\}$.

The following proposition captures a number of standard properties of Matroids; the proofs can be found in [22]. We shall only prove the last assertion.

Proposition 4: Let S_1, S_2, S_3 be subsets of U and $e \in U$ then

1) $\text{rank}(S_1) \leq |S_1|$, where equality holds if and only if S_1 is an independent set,
2) if $S_1 \subseteq S_2$ or $S_1 \subseteq \text{Cl}(S_2)$, then $S_1 \subseteq \text{Cl}(S_1) \subseteq \text{Cl}(S_2)$ and $\text{rank}(S_1) \leq \text{rank}(S_2)$,
3) if $e \notin \text{Cl}(S_1)$, then $\text{rank}(S_1 \cup \{e\}) = \text{rank}(S_1) + 1$,
4) $\text{rank}(S_1 \cup S_2) \leq \text{rank}(S_1) + \text{rank}(S_2)$,
5) $\text{rank}(S_1 \cup S_2) \leq \text{rank}(S_1) + \text{rank}(S_2 \setminus \text{Cl}(S_1))$, and
6) suppose that S_1 is minimal such that $e \in \text{Cl}(S_1 \cup S_2)$, but $e \notin \text{Cl}((S_1 \cup S_2) \setminus \{e^*\})$, for every $e^* \in S_1$, then $e^* \in \text{Cl}(\{e\} \cup ((S_1 \cup S_2) \setminus \{e^*\}))$, for every $e^* \in S_1$.

Proof: We prove Item 6. The rest of the items are standard properties of Matroids.

Let $e^* \in S_1$. By Item 3, $\text{rank}(\{e\} \cup ((S_1 \cup S_2) \setminus \{e^*\}))$ is equal to $\text{rank}(S_1 \cup S_2)$ which is equal to $\text{rank}(\{e\} \cup S_1 \cup S_2)$ which in turn is equal to $\text{rank}(\{e\} \cup ((S_1 \cup S_2) \setminus \{e^*\}) \cup \{e^*\})$. Thus, again by Item 3, this implies that $e^* \in \text{Cl}(\{e\} \cup ((S_1 \cup S_2) \setminus \{e^*\}))$. ∎

Assumption 5: $val(e) = 0$, for every $e \in U$ such that $\text{rank}(\{e\}) = 0$. For every $e \in U$ such that $val(e) > 0$, there exists $i \in \mathbb{Z}$ such that $val(e) = 2^i$.

In the worst case, the implication of this assumption is an increase in the competitive ratio by a multiplicative factor that does not exceed 2, compared with the competitive ratio we could achieve without this assumption.

Definition 6: **[Buckets]** For every $i \in \mathbb{Z}$, the i'th bucket is $B_i = \{e \in U \mid val(e) = 2^i\}$. We also use the following notation for every $S \subseteq U$ and $J \subset \mathbb{Z}$:

- $B_i^S = B_i \cap S$, $B_J = \bigcup_{i \in J} B_i$ and $B_J^S = \bigcup_{i \in J} B_i^S$.

Definition 7: **[OPT]** For every $S \subseteq U$, let $\text{OPT}(S) = \max\left\{\sum_{e \in S'} val(e) \,\middle|\, S' \subseteq S \text{ and } S' \in \mathcal{I}\right\}$. We note that if S is independent, then $\text{OPT}(S) = \sum_{e \in S} val(e)$.

Observation 8: For every independent $S \subseteq U$, $\text{OPT}(S) = \sum_{i \in \mathbb{Z}} 2^i \cdot \text{rank}\left(B_i^S\right)$.

Definition 9: **[LOPT]** For every $S \subseteq U$, we define $\text{LOPT}(S) = \sum_{i \in \mathbb{Z}} 2^i \cdot \text{rank}\left(B_i^S\right)$.

Observation 10: For every $S \subseteq U$ and $J_1, J_2 \subseteq \mathbb{Z}$,

1) $\text{LOPT}(S) \geq \text{OPT}(S)$,
2) $\text{LOPT}\left(B_{J_1}^S\right) = \sum_{i \in J_1} 2^i \cdot \text{rank}\left(B_i^S\right)$ and
3) if $J_1 \cap J_2 = \emptyset$, then $\text{LOPT}\left(B_{J_1 \cup J_2}^S\right) = \text{LOPT}\left(B_{J_1}^S\right) + \text{LOPT}\left(B_{J_2}^S\right)$.

III. Overview

The input to the Main Algorithm is the number of indices n in a randomly ordered input vector $(e_1, \ldots, e_n)$, where $\{e_1, \ldots, e_n\}$ are the elements of the ground set of the matroid. These are revealed to the Main Algorithm one by one in an on-line fashion in the increasing order of their indices. The Main Algorithm executes the following three stages:

Gathering stage.: Let $W = \beta(n, 1/2)$. Wait until W elements are revealed without selecting any. Let F be the set of all these elements.

Preprocessing stage.: Given only F, before any item of $U \setminus F$ is revealed, one of the following three types of output is computed: (i) a non-negative value, (ii) a set of bucket indices, or (iii) a critical tuple which is defined in Subsection IV-B.

Selection stage.: One out of three algorithms is chosen and used in order to decide which elements from $U \setminus F$ to select, when they are revealed. If the output of Preprocessing stage is a non-negative value, then the *Threshold Algorithm* is chosen, if it is a set of bucket indices, then the *Simple Algorithm* is chosen and if it is a critical tuple, then the *Gap Algorithm* is chosen.

With probability $\frac{1}{2}$, regardless of F, the output of the Preprocessing stage is a non-negative value. The Threshold Algorithm, that is used in this case, selects the first revealed element of $U \setminus F$ that has a value at least as large as the output of the Preprocessing stage. This ensures that if $\max\{val(e) \mid e \in U\}$ is large, then the claimed competitive-ratio is achieved.

The paper proceeds as follows: in Subsection IV-A, we present the Simple Algorithm and formally prove a guarantee on its output; in Subsection IV-B, we define critical tuple, describe the Gap Algorithm and formally prove a guarantee on its output; in Section V, prove the required concentrations; in Section VI, we prove our structural trade-off result; and in Section VII, we prove the main result.

IV. The Simple Algorithm and the Gap Algorithm

In this section we present the pseudo-code for the Simple Algorithm and the Gap Algorithm, and prove the guarantees on the competitive-ratio they achieves. We start with the Simple Algorithm, which is also used in [8].

A. The Simple Algorithm

Algorithm 1 Simple Algorithm

Input: a set J of bucket indices

1) $P \longleftarrow \emptyset$
2) immediately after each element $e \in U \setminus F$ is revealed, do
   a) if $\log val(e) \in J$ do
      i) if $e \notin \mathrm{Cl}(P)$ do $P \longleftarrow P \cup \{e\}$

Output: P

We note that according to Steps 2a and 2(a)i, the output P of the Simple Algorithm always satisfies, $B_J^{U\setminus F} \subseteq \mathrm{Cl}(P)$. Thus, since $P \subseteq B_J^{U\setminus F}$, the output P of the Simple Algorithm always satisfies, $\mathrm{rank}(P) = \mathrm{rank}\left(B_J^{U\setminus F}\right)$. As a result, for every $j \in J$, we are guaranteed that P contains at least $\mathrm{rank}\left(B_J^{U\setminus F}\right) - \mathrm{rank}\left(B_{J\setminus\{j\}}^{U\setminus F}\right)$ elements from $B_j^{U\setminus F}$. We capture this measure using the following definition:

Definition 11: **[uncov]** $\mathrm{uncov}(R,S) = \mathrm{rank}(R\cup S) - \mathrm{rank}(R)$, for every $R, S \subseteq U$.

According to this definition, for every $j \in J$, we are guaranteed that P contains at least $\mathrm{uncov}\left(B_{J\setminus\{j\}}^{U\setminus F}, B_j^{U\setminus F}\right)$ elements from $B_j^{U\setminus F}$. We next prove this in a slightly more general setting that is required for the Gap Algorithm.

Lemma 12: Suppose that the input to the Simple Algorithm is a set $J \subset \mathbb{Z}$ and, instead of the elements of $U \setminus F$, the elements of a set $S \subseteq U$ are revealed in an arbitrary order to the Simple Algorithm. Then the Simple Algorithm returns an independent set $P \subseteq S$ such that, for every $j \in J$, $\mathrm{rank}\left(B_j^P\right) \geq \mathrm{uncov}\left(B_{J\setminus\{j\}}^S, B_j^S\right)$.

Proof: By the same reasoning as described in the beginning of this section, for every $j \in J$, we are guaranteed that P contains at least $\mathrm{uncov}\left(B_{J\setminus\{j\}}^{U\setminus F}, B_j^{U\setminus F}\right)$ elements from $B_j^{U\setminus F}$ and the result follows. ∎

We next prove the following guarantee on the output of the Simple Algorithm, by using the preceding lemma.

Theorem 13: Given a set $J \subset \mathbb{Z}$ as input, the Simple Algorithm returns an independent set $P \subseteq U \setminus F$ such that

$$\mathrm{OPT}(P) \geq \sum_{j\in J} 2^j \cdot \mathrm{uncov}\left(B_{J\setminus\{j\}}^{U\setminus F}, B_j^{U\setminus F}\right).$$

Proof: By Observation 8, $\mathrm{OPT}(P)$ is at least $\sum_{j\in J} 2^j \cdot \mathrm{rank}\left(B_j^P\right)$, which is at least $\sum_{j\in J} 2^j \cdot \mathrm{uncov}\left(B_{J\setminus\{j\}}^{U\setminus F}, B_j^{U\setminus F}\right)$, by Lemma 12. The result follows. ∎

We note that the above guarantee is not necessarily the best possible. However, it is sufficient for our needs because, as we show later on, with very high probability, for a specific family of sets J and every j in such J, we have that $\mathrm{uncov}\left(B_{J\setminus\{j\}}^{U\setminus F}, B_j^{U\setminus F}\right) \approx \mathrm{uncov}\left(B_{J\setminus\{j\}}^{F}, B_j^{F}\right)$. Thus, in relevant cases, we can approximate this guarantee using only the elements of F.

B. The Gap Algorithm

The subsection starts with a description of the input to the Gap Algorithm and how it works; afterwards it provides a formal definition of the Gap Algorithm and its input and then concludes with a formal proof of the guarantee on the Gap Algorithm's output.

Like the Simple Algorithm the elements of $U \setminus F$ are revealed to the Gap Algorithm one by one in an on-line manner. The input to the Gap Algorithm is a tuple $(Block, Good, Bad)$, called a critical tuple. $Block$ is a mapping from the integers $\mathbb{Z}$ to the power set of the integers, such that if $Block(i)$ is not empty then $i \in Block(i)$. $Block$ determines from which buckets the Gap Algorithm may select elements. Specifically, an element $e \in U \setminus F$ may be selected only if $Block(\log val(e))$ is not empty. Every pair of not empty sets $Block(i)$ and $Block(j)$, where $i \geq j$, are such that either $Block(i) = Block(j)$ or $Block(i) > Block(j)$ and the latter may occur only if $i > j$. We recall that, as defined in the preliminaries, $Block(i) > Block(j)$ means that $\min Block(i) > \max Block(j)$. We next formally define the critical tuple.

Definition 14: **[critical tuple]** $(Block, Good, Bad)$, where $Good$, Bad and $Block$ are mappings from $\mathbb{Z}$ to $2^{\mathbb{Z}}$, is a **critical tuple** if the following hold for every $i, j \in \mathbb{Z}$ such that $i \geq j$ and $Block(i)$ and $Block(j)$ are not empty:

1) $i \in Block(i)$,
2) if $i > j$ either $Block(i) = Block(j)$ or $Block(i) > Block(j)$, i.e. $\min Block(i) > \max Block(j)$,
3) if $Block(i) = Block(j)$, then $Good(i) = Good(j)$ and $Bad(i) = Bad(j)$,
4) $Block(i) \cup Bad(i) \subseteq Good(i)$,
5) if $Block(i) > Block(j)$, then $Bad(i) \subseteq Good(i) \subseteq Bad(j) \subseteq Good(j)$,
6) $Block(i) < Bad(i)$.

For a depiction of preceding structure see Figure 1.

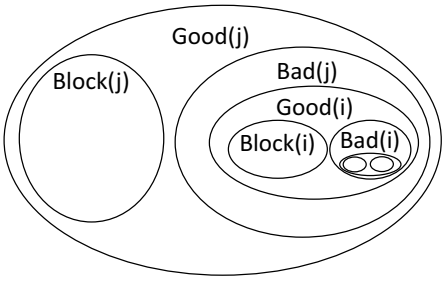

Figure 1. [critical tuple, where $Block(i) > Block(j)$]

The following observation, follow directly from the preceding definition.

Observation 15: If $(Block, Good, Bad)$ is a **critical tuple**, then

1) the sets in $\{Block(j)\}_{j\in\mathbb{Z}}$ are pairwise-disjoint,
2) $Block(i) \cap Bad(i) = \emptyset$, for every $i \in \mathbb{Z}$, and
3) for every i and j in $\bigcup_{\ell\in\mathbb{Z}} Block(\ell)$, if $j \notin Good(i)$, then $i > j$ and $Good(i) \subseteq Bad(j)$.

The mappings $Good$ and Bad are used in order to determine if an element can be selected as follows: an element $e \in U \setminus F$ such that $\log val(e) \in Block(\log val(e))$ is selected if it satisfies two conditions: (i) $e \in \text{Cl}\left(B^F_{Good(i)}\right)$; and (ii) e is in the closure of the union of $B^F_{Bad(i)}$ and all the previously selected elements. We next explain why this strategy works.

Clearly, the only elements in $B_i^{U\setminus F}$ that do not satisfy condition (i) are those in $B_j^{U\setminus F} \setminus \text{Cl}\left(B^F_{Good(i)}\right)$. An essential part of our result is an upper bound on the rank of the set $B_i^{U\setminus F} \setminus \text{Cl}\left(B^F_{Good(i)}\right)$ and hence we use the following definition to capture this quantity.

Definition 16: **[loss]** For every $R, S \subseteq U$, let $\text{loss}(R, S) = \text{rank}(S \setminus \text{Cl}(R))$.

According to this definition and the preceding explanation we are guaranteed that the rank of the set of elements in $B_i^{U\setminus F}$ that satisfy condition (i) is at least $\text{rank}\left(B_i^{U\setminus F}\right) - \text{loss}\left(B^F_{Good(i)}, B_i^{U\setminus F}\right)$.

For every $j \in Block(i)$, let $S_j = B_j^{U\setminus F} \cap \text{Cl}\left(B^F_{Good(j)}\right)$, that is, the elements of S_j are the elements of $B_j^{U\setminus F}$ that satisfy condition (i). We will show that such an element satisfies condition (ii) if it is not in the closure of the union of $B^F_{Bad(i)}$ and **only** all the elements from $B_i^{U\setminus F}$ that were previously selected. The reason this happens is that, for every $j' > i$, such that $Block(j') > Block(i)$ all the element selected from $B_{j'}^{U\setminus F}$, satisfy condition (i) and hence are in $\text{Cl}\left(B^F_{Bad(i)}\right)$, and for every $j' < i$, such that $Block(j') < Block(i)$ the condition (ii) ensures, for every $j \in Block(j')$, that each element selected from $B_j^{U\setminus F}$ will not prevent the selection of any element from S_i because $S_i \subseteq \text{Cl}\left(B^F_{Good(i)}\right) \subset \text{Cl}\left(B^F_{Bad(j)}\right)$.

Thus, when restricted to the elements of $\bigcup_{j\in Block(i)} S_j$, the Gap Algorithm can be viewed as if it was executing the Simple Algorithm with input $J = Block(i) \cup Bad(i)$ and the elements revealed are those of $S = B^F_{Bad(i)} \cup \bigcup_{j\in Block(i)} S_j$, which are revealed in an arbitrary order, except that the elements of $B^F_{Bad(j)}$ are revealed first. Thus, using Lemma 12, it is straight forward to see that at least $\text{uncov}\left(B^F_{Bad(i)} \cup \bigcup_{j\in Block(i)\setminus\{i'\}} S_j, S_{i'}\right)$ are selected from $S_{i'}$, for every $i' \in Block(i)$. We shall show, that this term, is at least $\text{uncov}\left(B^F_{Bad(j)} \cup B^{U\setminus F}_{Block(i)\setminus\{i\}}, S_{i'}\right)$, which in turn is at least $\text{uncov}\left(B^F_{Bad(j)} \cup B^{U\setminus F}_{Block(i)\setminus\{i\}}, B^{U\setminus F}_{i'}\right) - \text{loss}\left(B^F_{Good(i)}, B_i^{U\setminus F}\right)$. In Section V, we show that with high probability, by using only the elements of F, we can approximate $\text{uncov}\left(B^F_{Bad(j)} \cup B^{U\setminus F}_{Block(i)\setminus\{i\}}, B^{U\setminus F}_{i'}\right)$ and upper bound $\text{loss}\left(B^F_{Good(i)}, B_i^{U\setminus F}\right)$. In Section VI, we use the result of Section V to show that, if there is no element with a very high value, then either the Simple Algorithm or the Gap Algorithm will achieve the required competitive-ratio and we can choose the proper option using only the elements of F.

Algorithm 2 Gap Algorithm

Input: a critical tuple $(Block, Good, Bad)$

1) $P \longleftarrow \emptyset$
2) immediately after each element $e \in U \setminus F$ is revealed do
   a) $\ell \longleftarrow \log val(e)$
   b) if $Block(\ell) \neq \emptyset$ do
      i) if $e \in \text{Cl}\left(B^F_{Good(\ell)}\right)$, do
         A) if $e \notin \text{Cl}\left(P \cup B^F_{Bad(\ell)}\right)$, do $P \longleftarrow P \cup \{e\}$

Output: P

Lemma 17: Let i be such that $Block(i) \neq \emptyset$ and P as it was in any stage in an arbitrary execution of the Gap Algorithm. If $e \in B^{U\setminus F}_{Block(i)} \cap \text{Cl}\left(B^F_{Good(i)}\right)$, then $e \notin \text{Cl}\left(P \cup B^F_{Bad(i)}\right)$ if and only if $e \notin \text{Cl}\left(\left(P \cap B^{U\setminus F}_{Block(i)}\right) \cup B^F_{Bad(i)}\right)$.

Proof: Let $e \in B^{U\setminus F}_{Block(i)} \cap \text{Cl}\left(B^F_{Good(i)}\right)$. We note that the "only if" condition trivially holds and hence we only prove the "if" condition. Let $P^* = P \cap B^{U\setminus F}_{Block(i)}$. Assume that $e \in \text{Cl}\left(P \cup B^F_{Bad(i)}\right)$. Let C be a minimal subset of $P \setminus (P^* \cup \text{Cl}\left(B^F_{Bad(i)}\right))$ such that $e \in \text{Cl}\left(C \cup B^F_{Bad(i)} \cup P^*\right)$. We shall show that $C = \emptyset$ and

hence $e \in \text{Cl}\left(\left(P \cap B^{U\setminus F}_{Block(i)}\right) \cup B^F_{Bad(i)}\right)$. Hence the result then follows.

Let e' be the latest element C added to P and let $j = \log val(e')$. According to construction, the elements of C were selected by the Gap Algorithm and hence $Block(j) \neq \emptyset$. Also, by construction, $Block(i) \neq Block(j)$, since otherwise $e' \in P^* = P \cap B^{U\setminus F}_{Block(i)}$.

Now assume on the other hand that $Block(j) > Block(i)$. By Items 4 and 5 of Definition 14, this implies that $Block(j) \subseteq Good(j) \subseteq Bad(i)$. Since e' was selected by the Gap Algorithm, by Step 2(b)i, this implies that $e' \in \text{Cl}\left(B^F_{Good(j)}\right) \subseteq \text{Cl}\left(B^F_{Bad(i)}\right)$. This contradicts the choice of $e' \in C \subseteq P \setminus (P^* \cup \text{Cl}\left(B^F_{Bad(i)}\right))$.

Assume on the other hand that $Block(j) < Block(i)$. By Items 4 and 5 of Definition 14, this implies that $Block(i) \cup Bad(i) \subseteq Good(i) \subseteq Bad(j)$ and hence $e \in B^{U\setminus F}_{Block(i)} \cap \text{Cl}\left(B^F_{Good(i)}\right) \subseteq \text{Cl}\left(B^F_{Bad(j)}\right)$ and, using Item 5 of the definition of a critical tuple, $B^F_{Bad(i)} \cup P^* \subseteq B^F_{Bad(i)} \cup \text{Cl}\left(B^F_{Good(i)}\right) \subseteq \text{Cl}\left(B^F_{Bad(j)}\right)$ since, by Step 2(b)i, every element in P^* is in $\text{Cl}\left(B^F_{Good(i)}\right)$. With out loss of generality, we assume that e' was the last element in C that was added to P, otherwise we initially select e' to be so.

Since e' was the latest element C added to P, by Item 6 of Proposition 4, $e' \in \text{Cl}\left(\{e\} \cup \left(C \cup B^F_{Bad(i)} \cup P^*\right) \setminus \{e'\}\right)$. Since $e \in \text{Cl}\left(B^F_{Bad(j)}\right)$ and $B^F_{Bad(i)} \cup P^* \subseteq \text{Cl}\left(B^F_{Bad(j)}\right)$, we see that $e' \in \text{Cl}\left(\left(C \cup B^F_{Bad(j)}\right) \setminus \{e'\}\right)$. Therefore, e' did not satisfy the condition in Step 2(b)iA, when it was considered and hence could not have been selected to be in P. This contradicts the fact that $e' \in C \subseteq P$. ∎

Theorem 18: Given a critical tuple $(Block, Good, Bad)$ as input, Algorithm 2 returns an independent set of elements $P \subseteq U \setminus F$ such that $\text{OPT}(P)$ is at least $\sum_{j:Block(j)\neq\emptyset} 2^j \cdot \text{uncov}\left(B^F_{Bad(j)} \cup B^{U\setminus F}_{Block(j)\setminus\{j\}}, B^{U\setminus F}_j\right)$ minus $\sum_{j:Block(j)\neq\emptyset} 2^j \cdot \text{loss}\left(B^F_{Good(j)}, B^{U\setminus F}_j\right)$.

Proof: Step 2(b)iA implies that P is always an independent set. Consider j such that $Block(j) \neq \emptyset$. For every $i \in Block(j)$, we let $S_i = B^{U\setminus F}_i \cap \text{Cl}\left(B^F_{Good(i)}\right)$.

We note that, by definition, for every $i \in Block(j)$, every element in S_i satisfies the condition in Step 2(b)i. Consequently, by Lemma 17, the Gap Algorithm processes the elements in $\bigcup_{i\in Block(j)} S_i$, as if it was the Simple Algorithm in the following setting: the input is a set $J = Block(j) \cup Bad(j)$ and the elements revealed are those of $S = B^F_{Bad(j)} \cup \bigcup_{i\in Block(j)} S_i$, which are revealed in an arbitrary order, except that the elements of $B^F_{Bad(j)}$ are revealed first. Thus, by Lemma 12, $\text{rank}\left(B^P_j\right)$ is at least $\text{uncov}\left(B^F_{Bad(j)} \cup \bigcup_{i\in Block(j)\setminus\{j\}} S_i, S_j\right)$.

Let $B^{U\setminus F}_j$, $R_1 = B^F_{Bad(j)} \cup \bigcup_{i\in Block(j)\setminus\{j\}} S_i$ and $R_2 = B^F_{Bad(j)} \cup B^{U\setminus F}_{Block(j)\setminus\{j\}}$. Then, $\text{rank}\left(B^P_j\right) \geq \text{uncov}(R_1, S_j)$. We next show that $\text{uncov}(R_1, S_j)$ is at least $\text{uncov}(R_2, S_j)$ and afterwards that $\text{uncov}(R_2, S_j)$ is at least $\text{uncov}\left(R_2, B^{U\setminus F}_j\right) - \text{loss}\left(B^F_{Good(j)}, B^{U\setminus F}_j\right)$. By Observation 8, this implies the result.

We let $R' = R_1$, which trivially implies that $\text{uncov}(R', S_j) = \text{uncov}(R_1, S_j)$, and then we start adding the elements of $R_2 \setminus R_1$ to R'. Since $R_1 \subseteq R_2$ at the end of the process $R' = R_1$. We note that during the process the value $\text{uncov}(R', S_j) = \text{rank}(R' \cup S_j) - \text{rank}(R')$ never increases since when $\text{rank}(R' \cup S_j)$ increases by one so does $\text{rank}(R')$. Thus, indeed $\text{uncov}(R_1, S_j)$ is at least $\text{uncov}(R_2, S_j)$.

By definition, $\text{uncov}(R_2, S_j)$ is equal to $\text{uncov}\left(R_2, B^{U\setminus F}_j\right) - \left(\text{rank}\left(R_2 \cup B^{U\setminus F}_j\right) - \text{rank}(R_2 \cup S_j)\right)$. Finally, $S_j = B^{U\setminus F}_j \cap \text{Cl}\left(B^F_{Good(i)}\right)$, we see that $\text{rank}\left(R_2 \cup B^{U\setminus F}_j\right) - \text{rank}(R_2 \cup S_j)$ is at most $\text{rank}\left(B^{U\setminus F}_j \setminus \text{Cl}\left(B^F_{Good(i)}\right)\right)$, which is equal to $\text{loss}\left(B^F_{Good(i)}, B^{U\setminus F}_j\right)$. Therefore, indeed $\text{uncov}(R_2, S_j)$ is at least $\text{uncov}\left(R_2, B^{U\setminus F}_j\right) - \text{loss}\left(B^F_{Good(i)}, B^{U\setminus F}_j\right)$. ∎

V. Prediction

In this section, we prove that for a specific subset of the integers, which we denote by $Super$ and later, with constant probability, for every $K, K^* \subseteq Super$, where $K^* < K$ and $\min\{\text{rank}(B_i) \mid i \in K\} \geq \text{rank}(B_{\min K})^{\frac{6}{7}}$, and for every $j \in K$ we have that (i) $\text{uncov}\left(B^F_{K^*} \cup B^{U\setminus F}_{K\setminus\{j\}}, B^{U\setminus F}_j\right)$ is approximately $\text{uncov}\left(B^F_{K^*\cup K\setminus\{j\}}, B^F_j\right)$; and (ii) $\text{loss}\left(B^F_K, B^{U\setminus F}_j\right)$ is bounded above by approximately $\text{uncov}\left(B^F_{K\setminus\{j\}}, B^F_j\right)$. This result enables us at Preprocessing stage of the Main Algorithm to chose whether to select elements using the Simple Algorithm or the Gap Algorithm, and to compute the input to the chosen algorithm.

In Subsection V-B, we use the Talagrand inequality for the unquantified version of (i), in Subsection V-A, we use Martingales and Azuma's inequality fur the unquantified version of (ii) and in Subsection V-C, we define the set $Super$ and use the Union Bound together with the results in the previous sections to prove the main result of this section.

A. Upper Bounding loss

Theorem 19: Let $K \subset \mathbb{Z}$, be finite and non-empty and $k \in K$ then, $\text{loss}\left(B^F_K, B^{U\setminus F}_k\right)$ does not exceed $\text{uncov}\left(B^F_{K\setminus\{k\}}, B^F_k\right) + \text{rank}(B_k)^{\frac{2}{3}}$, with probability greater than $1 - e^{-\text{rank}(B_k)^{\frac{1}{3}}}$.

Proof: We fix $S = B^F_{K\setminus\{k\}}$ and let $m = \text{rank}\,(B_k)$. We initially let both H^F and $H^{U\setminus F}$ be empty sets. Then, we repeat the following $4m$ times: if there exists an element in $B_k \setminus (H^F \cup H^{U\setminus F})$ that is not in $\text{Cl}\left(S \cup H^F\right)$, then we pick such an element arbitrarily, if it is in F, then we add it to H^F and otherwise we add it to $H^{U\setminus F}$.

We observe that every time an element is added to H^F it is independent of $\text{Cl}\left(S \cup H^F\right)$ and hence it increases by one the quantity $\text{uncov}\left(S, H^F\right) = \text{rank}\left(S \cup H^F\right) - \text{rank}\,(S)$. Thus, if after $4m$ repetitions there are no elements in $B_k \setminus \text{Cl}\left(S \cup H^F\right)$, then the preceding quantity cannot be increased further by adding elements from $B_k \setminus (H^F \cup H^{U\setminus F})$ to H^F and therefore $\text{uncov}\left(S, H^F\right) = |H^F|$. Since, in this case every element in $B^F_k \setminus H^F$ is in $\text{Cl}\left(S \cup H^F\right)$ and $H^F \subseteq B^F_k$, we see that $\text{Cl}\left(S \cup B^F_k\right) = \text{Cl}\left(S \cup H^F\right)$. Therefore, $\text{rank}\left(S \cup B^F_k\right) = \text{rank}\left(S \cup H^F\right)$. Hence, by the definition of uncov, $\text{uncov}\left(S, B^F_k\right) = \text{uncov}\left(S, H^F\right) = |H^F|$.

We also observe that every time an element is added to $H^{U\setminus F}$ it may increase by one the quantity $\text{loss}\left(S \cup H^F, H^{U\setminus F}\right) = \text{rank}\left(H^{U\setminus F} \setminus \text{Cl}\left(S \cup H^F\right)\right)$. We note that if after $4m$ repetitions there are no elements in $B_k \setminus \text{Cl}\left(S \cup H^F\right)$, then the preceding quantity cannot be increased further by adding elements from $B_k \setminus (H^F \cup H^{U\setminus F})$ to $H^{U\setminus F}$ and therefore $\text{loss}\left(S \cup H^F, H^{U\setminus F}\right) \leq |H^{U\setminus F}|$. Since, in this case every element in $B^{U\setminus F}_k \setminus H^{U\setminus F}$ is in $\text{Cl}\left(S \cup H^F\right)$ and $H^{U\setminus F} \subseteq B^{U\setminus F}_k$, we see that $\text{loss}\left(S \cup H^F, H^{U\setminus F}\right) = \text{loss}\left(S \cup H^F, B^{U\setminus F}_k\right)$. We note that $S \cup B^F_k = B^F_K$ and we already proved $\text{Cl}\left(S \cup B^F_k\right) = \text{Cl}\left(S \cup H^F\right)$. Thus, $\text{loss}\left(B^F_K, B^{U\setminus F}_k\right) = \text{loss}\left(S \cup H^F, H^{U\setminus F}\right) \leq |H^{U\setminus F}|$.

Next we show that, $||H^F| - |H^{U\setminus F}|| \leq m^{\frac{2}{3}}$, with probability at least $1 - e^{-2m^{\frac{1}{3}}}$, and afterwards we show that, with probability at least $1 - e^{-\frac{m}{2}}$, after $4m$ repetitions, there are no elements in $B_k \setminus \text{Cl}\left(S \cup H^F\right)$. By the union bound, this implies the theorem.

We define the variables Z_i so that $Z_0 = 0$ and (i) $Z_i = Z_{i-1} - 1$ if in the i'th repetition an element was added to H^F; (ii) $Z_i = Z_{i-1} + 1$ if in the i'th repetition an element was added to $H^{U\setminus F}$; and (iii) $Z_i = Z_{i-1}$ if nothing happened in the i'th repetition.

We note that, for every $i > 0$, either $Z_i = Z_{i-1}$ or Z_i is distributed uniformly over $\{Z_{i-1} - 1, Z_{i-1} + 1\}$ and hence $E(Z_i \mid Z_{i-1}) = Z_{i-1}$, where $E()$ denotes the expected value. Consequently, we have a martingale. Thus, by Azuma's inequality, $Z_{4m} > 4m^{\frac{2}{3}}$ with probability less than $e^{-m^{\frac{2}{3}}}$. Since, $Z_{4m} = |H^{U\setminus F}| - |H^F|$ we have proved the first inequality. We now proceed to the second.

We define the variables X_i so that $X_i = 1$ if in the ith repetition the element processed was in F and otherwise $X_i = 0$. By definition, for every $i \in [4m]$, if $Z_i = Z_{i-1} - 1$, then $X_i = 1$. If $|H^F| = m$ after $4m$ repetitions, then $\text{rank}\left(S \cup H^F\right) = \text{rank}\,(S) + \text{rank}\,(B_k)$, which can only happen if $B_k \subseteq \text{Cl}\left(S \cup H^F\right)$. This implies that $B_k \setminus \text{Cl}\left(S \cup H^F\right)$ is empty. So $B_k \setminus \text{Cl}\left(S \cup H^F\right)$ is not empty after $4m$ only if $\sum_{i=1}^{4m} X_i < m$. By Observation 1, for every $i \in [4m]$, X_i is independently distributed uniformly over $\{0, 1\}$. By the Chernoff inequality, with probability at least $1 - e^{-\frac{m}{2}}$, $\sum_{i=1}^{4m} X_i \geq m$. ∎

B. Talagrand based concentrations

This subsection is very similar to one that appears in [8], we include it for the sake of completeness. The following definition is an adaptation of the Lipschitz condition to our setting.

Definition 20: **[Lipschitz]** Let $f : U \longrightarrow \mathbb{N}$. If $|f(S_1) - f(S_2)| \leq 1$ for every $S_1, S_2 \subseteq U$ such that $|(S_1 \setminus S_2) \cup (S_2 \setminus S_1)| = 1$, then f is **Lipschitz**.

Definition 21: **[Definition 3, Section 7.7 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 $h(y) \geq s$.

Observation 22: For every finite $K \subset \mathbb{Z}$, the $rank$ function over subsets of B_K is Lipschitz and f-certifiable with $f(s) = \text{rank}\,(B_K)$, for all s.

Proof: The $rank$ function is Lipschitz, by the definition of the $rank$ function (Definition 3). By Item 2 of Proposition 4, for every $S \subseteq R \subseteq B_K$, we have that $\text{rank}\,(S) \leq \text{rank}\,(R) \leq \text{rank}\,(B_K)$. Thus, the $rank$ function over subsets of B_K is f-certifiable with $f(s) = \text{rank}\,(B_K)$. ∎

The succeeding theorem is a direct result of Theorem 7.7.1 from [1].

Theorem 23: If h is Lipschitz and f certifiable, then for x selected uniformly from Ω and all b, t, $Pr[h(x) \leq b - t\sqrt{f(b)}] \cdot Pr[h(x) \geq b] \leq e^{-t^2/4}$.

Lemma 24: Let $t \geq 2$, $K, K^* \subseteq \mathbb{Z}$, where $K^* \cap K = \emptyset$, and $j \in K$ then, $\left|\text{uncov}\left(B^F_{K^*} \cup B^{U\setminus F}_{K\setminus\{j\}}, B^{U\setminus F}_j\right) - \text{uncov}\left(B^F_{K^*\cup K\setminus\{j\}}, B^F_j\right)\right| \leq 4t\sqrt{\text{rank}\,(B_{K^*\cup K})}$, with probability at most $e^{4-\frac{t^2}{4}}$.

Proof: Let $S \in \{B^F_{K^*\cup K}, B^F_{K^*\cup K\setminus\{j\}}, B^F_{K^*} \cup B^{U\setminus F}_K, B^F_{K^*} \cup B^{U\setminus F}_{K\setminus\{j\}}\}$. By Observation 22, the $rank$ function is Lipschitz and $rank$-certifiable.

Clearly, since F and $U \setminus F$ are both distributed uniformly, with probability at least $\frac{1}{2}$, we have that $\text{rank}\,(S) \geq \text{med}\,(\text{rank}\,(S))$, Hence, by taking $b = \text{med}\,(\text{rank}\,(S)) + t\sqrt{\text{rank}\,(B_{K^*\cup K})}$, by Theorem 23, we get that $\text{rank}\,(S) - \text{med}\,(\text{rank}\,(S)) \geq t\sqrt{\text{rank}\,(B_{K^*\cup K})}$, with probability at most $2e^{\frac{t^2}{4}}$. In a similar manner, by taking $b = \text{med}\,(\text{rank}\,(S))$, we get that $\text{med}\,(\text{rank}\,(S)) - \text{rank}\,(S) \geq t\sqrt{\text{rank}\,(B_{K^*\cup K})}$, with probability at most $2e^{\frac{-t^2}{4}}$. Thus, by the union bound, $|\text{rank}\,(S) - \text{med}\,(\text{rank}\,(S))| \geq t\sqrt{\text{rank}\,(B_{K^*\cup K})}$, with probability at most $4e^{-\frac{t^2}{4}}$.

We note that, since F and $U \setminus F$ are identically distributed and $K \cap K^* = \emptyset$, we have that

$\text{med}\left(\text{rank}\left(B^F_{K^*\cup K}\right)\right) = \text{med}\left(\text{rank}\left(B^F_{K^*}\cup B^{U\setminus F}_K\right)\right)$ and $\text{med}\left(\text{rank}\left(B^F_{K^*\cup K\setminus\{j\}}\right)\right) = \text{med}\left(\text{rank}\left(B^F_{K^*}\cup B^{U\setminus F}_{K\setminus\{j\}}\right)\right)$. Consequently, the result follows, by the union bound and the definition of uncov (Definition 11). ∎

C. Union bound

Definition 25: **[Super]** We define $Super = \left\{i \middle| \text{rank}\left(B_i\right) \geq \max\left\{1, \sqrt{\frac{\text{LOPT}(U)}{2^{i+12}}}\right\}\right\}$.

Theorem 26: If $\max\{val(e) \mid e \in U\} < 2^{-64}\cdot\text{OPT}\left(U\right)$ then, with probability at least $\frac{3}{4}$, for every $K, K^* \subseteq Super$, where $K^*\cap K = \emptyset$ and $\min\{\text{rank}\left(B_K\right) \mid i \in K\} \geq \text{rank}\left(B_{\min K}\right)^{\frac{6}{7}}$, and every $k\in K$, the following hold:

1) $\left|\text{uncov}\left(B^F_{K^*}\cup B^{U\setminus F}_{K\setminus\{k\}}, B^{U\setminus F}_k\right) - \text{uncov}\left(B^F_{K^*\cup K\setminus\{k\}}, B^F_k\right)\right| \leq 4\cdot\text{rank}\left(B_{K^*\cup K}\right)^{\frac{2}{3}}$
2) $\text{loss}\left(B^F_K, B^{U\setminus F}_k\right) \leq \text{uncov}\left(B^F_{K\setminus\{k\}}, B^F_k\right) + \text{rank}\left(B_k\right)^{\frac{2}{3}}$.

The proof is a combination of the Union Bound and Theorem 19 and Lemma 24. It is omitted do to lack of space.

VI. Structural Theorem

In this section F is used after all its elements have been revealed and hence it is treated as fixed. The goal of this section is to show that if the maximum value of the elements of the Matroid is sufficiently small, then an input that guarantees the claimed competitive-ratiocan be found either for the Simple Algorithm or for the Gap Algorithm. The search for such an input is restricted to a subset of the follow set of indices.

Definition 27: **[Valuable]** We define $Valuable = \left\{j \middle| \text{rank}\left(B^F_j\right) \geq \max\left\{1, \sqrt{\frac{\text{LOPT}(F)}{2^{j+8}}}\right\}\right\}$.

We note that this definition is very similar to the definition of $Super$. This enables us to prove later that, with high probability, $Valuable \subseteq Super$, and hence the concentration results apply to the bucket indices in $Valuable$.

Proposition 28: $|Valuable| \leq 2\cdot\log\text{rank}\left(F\right)+8$

Proof: By definition, $\max Valuable \leq \log\text{LOPT}\left(F\right)$. By the definition of $Valuable$, $\sqrt{\frac{\text{LOPT}(F)}{2^{\min Valuable+8}}} \leq \text{rank}\left(F\right)$ and hence $\min Valuable \geq \log\text{LOPT}\left(F\right) - 2\cdot\log\text{rank}\left(F\right) - 8$. Since $Valuable$ contains only integers, the result follows. ∎

The rest of definitions are used in order to show that if we cannot find an input for the Simple Algorithm that guarantees the claimed competitive-ratio, then we can find an input to the Gap Algorithm that guarantees the claimed competitive-ratio. The first two definitions are used in order to restrict the set of buckets used to one that has properties that are essential for the rest of the result.

Definition 29: **[Strong sequence]** A sequence H of integers $h_1, h_2, \ldots, h_k$ is a **strong sequence** for $K\subset\mathbb{Z}$, if

1) $h_1, h_2, \ldots, h_k \in K$,
2) H is strictly monotonically decreasing,
3) $\text{LOPT}\left(B^F_H\right) \geq \frac{1}{18}\cdot\text{LOPT}\left(B^F_K\right)$ and
4) for every $j\in[k-1]$, $0 < \text{rank}\left(B^F_{h_j}\right) \leq \frac{1}{32}\cdot\text{rank}\left(B^F_{h_{j+1}}\right)$.

Lemma 30: For every $K\subset\mathbb{Z}$, there exists a strong sequence for K.

Proof: Define, $w:\mathbb{Z}\longrightarrow\mathbb{N}$ as follows: for every $i\in\mathbb{Z}$, $w(i) = \text{rank}\left(B^F_i\right)$ if $i\in K$, and otherwise $w(i)=0$. Let $m = \sum_{j\in\mathbb{Z}} w(j)\cdot 2^j = \text{LOPT}\left(B^F_K\right)$.

Let us construct a sequence of integers $\ell_1, \ell_2, \ldots, \ell_{k'}$ as follows: ℓ_1 is maximum so that $w(\ell_1) > 0$ and for every $j\in[k'-1]$, inductively, ℓ_{j+1} is the maximum integer such that $\ell_{j+1} < \ell_j$ and $w(\ell_j) \leq \frac{1}{2}\cdot w(\ell_{j+1})$. We note that, $0 < w(\ell_i) \leq \frac{1}{2}\cdot w(\ell_{i+1})$, for every $i\in[k'-1]$. We also note that $\sum_{i\in[k']} w(\ell_i)\cdot 2^{\ell_i} > \frac{m}{3}$, because for every $i\in[k']$, the sum of $2^j\cdot w(j)$ over all integers j less than ℓ_i but greater ℓ_{i+1}, when ℓ_{i+1} exists, is at most $2\cdot 2^{\ell_i}\cdot w(\ell_i)$. Thus, $\sum_{i\in[k']} w(\ell_i)\cdot 2^{\ell_i} \geq m - \frac{2m}{3} = \frac{m}{3}$.

By the pigeon hole principle, there exists $q\in[6]$ such that the sum of $w(\ell_{q+6i})\cdot 2^{\ell_{q+6i}}$ over all $i\geq 1$ such that $q+6i\leq k'$, is at least $\frac{m}{18}$. We denote this sequence by $h_1, h_2, \ldots, h_k$. We note that according to construction, $0 < w(h_i) \leq \frac{1}{32}\cdot w(h_{i+1})$, for every $i\in[k-1]$. Thus, this sequence satisfies **Items 3 and 4** of the definition of a strong sequence. By construction the sequence $h_1, h_2, \ldots, h_k$ is strictly monotonically increasing and all its elements are in K and hence it satisfies **Items 1 and 2** of the definition of a strong sequence. ∎

The next observation is a direct result of the definition of $uncov$ (Definition 11), the definition of a strong sequence and Item 4 of Proposition 4.

Observation 31: Let H be a strong sequence for a set of integers, $H'\subset H$ and $j\in H'$ such that $j < \min H'$, then $\text{uncov}\left(B^F_{H'}, B^F_j\right) \geq \frac{31}{32}\cdot\text{rank}\left(B^F_j\right)$.

Definition 32: **[Partition]** For every $H\subset\mathbb{Z}$, we let *Partition*(H) to be an arbitrarily fixed partition of H into sub-sequences $H_1, H_2, \ldots, H_g$ such that:

1) $g \leq \max\left\{1, 8\cdot\log\log\text{rank}\left(B^F_K\right)\right\}$
2) for every $i\in[g-1]$, $H_i > H_{i+1}$,
3) for every $i\in[g]$, $\text{rank}\left(B^F_{\max H_i}\right) \geq \text{rank}\left(B^F_{\min H_i}\right)^{\frac{8}{9}}$.

The next observation follows from Item 4 of Definition 29 and Item 3 of Definition 32.

Lemma 33: For every, $K\subset\mathbb{Z}$ and sequence H of integers $h_1, h_2, \ldots, h_k$ that is a strong sequence for K, Partition(H) is well defined.

The proof is based on the definition of a strong sequence. It is omitted due to lack of space.

Definition 34: **[M_H]** For every $K\subseteq H\subset\mathbb{Z}$ be let $M_H\left(K\right) = \{i\in H \mid i > K\}$. We omit the subscript when

clear from context.

Observation 35: Let H be a strong sequence for a set of integers. For every $H' \in \text{Partition}(H)$ and $\ell \in H'$, we have $2 \cdot \text{rank}\left(B_\ell^F\right)^{\frac{9}{8}} \geq \text{rank}\left(B_{M_H(H') \cup H'}^F\right)$.

At this stage we are preparing the ground for proving that if a sufficient input for the Simple Algorithm does not exist, then we can find sufficient critical tuple for the Gap Algorithm. The idea is to construct a hierarchical decomposition of the buckets whose indices are in a partition of a strongsequence. The construction starts with the sets of the partition. During the construction process each set gets one of three treatments: if a set is negligible i.e., does not have significant value, then it is ignored, this is also the case if a set is burnt i.e., has a very high uncover value; if a set can contribute significantly to the success of the Gap Algorithm i.e., is useful, then it will be used; and if a set is none of the above, then it is splittable, as we show later, and will indeed be split. The process we just described is captured further on by the definition of a critical-tree.

Definition 36: **[useful]** A subset K^* of $K \subseteq H$ is **useful** for K if the following hold:

1) $\text{LOPT}\left(B_{K^*}^F\right) > \frac{1}{32} \cdot \text{LOPT}\left(B_K^F\right)$
2) sum of every $j \in K^*$ of 2^j times $\text{uncov}\left(B_{M(K^*) \cup K^* \setminus \{j\}}^F, B_j^F\right)$ minus 2^j times $\text{uncov}\left(B_{M(K) \cup K \setminus \{j\}}^F, B_j^F\right)$ is at least $\frac{\text{LOPT}\left(B_{K^*}^F\right)}{2^{11} \cdot \log\log \text{rank}(F)}$.

A set that has a useful subset is **useful**.

Definition 37: **[splittable]** $K \subseteq H$ is **splittable** if it has a partition $\{K_1, K_2\}$ such that

1) $K_1 > K_2$ and
2) $\text{LOPT}\left(B_{K_i}^F\right) > \frac{1}{32} \cdot \text{LOPT}\left(B_K^F\right)$, for every $i \in [2]$.

Definition 38: **[negligible]** A set $K \subseteq H$ is **negligible** if $\text{LOPT}\left(B_K^F\right) < \frac{\text{LOPT}\left(B_H^F\right)}{64 \cdot \log \text{rank}(F)}$.

Definition 39: **[burnt]** A set $K \subseteq H$ is **burnt** if

$$\sum_{j \in K} 2^j \cdot \text{uncov}\left(B_{M(K) \cup K \setminus \{j\}}^F, B_j^F\right) > \frac{3}{4} \cdot \text{LOPT}\left(B_K^F\right).$$

Definition 40: **[critical-tree]** A critical-tree for $H \subset \mathbb{Z}$ is a rooted tree whose vertices are subsets of H and that satisfies the following:

1) the root of the tree is H,
2) the children of the root are $\text{Partition}(H)$,
3) every leaf is either negligible, useful or burnt, and
4) every internal vertex K is splittable and neither useful, nor negligible nor burnt. It also has a set of two children that is a partition of K as described in the definition of splittable.

We next prove that the construction of a critical-tree actually works and results in a bounded depth tree; after we prove that a proper critical tuple can be extracted from a critical-tree.

Lemma 41: Assume that there exists a critical-tree $\mathcal{T}$, whose root is labeled by H. If $\text{rank}(F) \geq 2^8$, then the depth of $\mathcal{T}$ does not exceed $2^8 \cdot \log\log \text{rank}(F)$.

The proof is based on the definition of splittable. It is omitted due to lack of space.

Lemma 42: Let H be a strong sequence for a set of integers, then there exists a critical-tree $\mathcal{T}$, whose root is labeled by H.

We construct $\mathcal{T}$ as follows: we set the root to be H and its children to be $\text{Partition}(H)$; then, as long as there is a leaf K in the tree that is splittable but not useful, negligible or burnt we add two children to K the sets K_1 and K_2, where $\{K_1, K_2\}$ form a partition of K as in the definition of splittable; if there are no such leaves, we stop.

The rest of the proof follows from Lemma 41 and the definitions of burnt, negligible splittable and useful. It is not given due to lack of space.

Theorem 43: Let $H^* \subseteq Valuable$, H be a strong sequence for H^* and assume that $\text{rank}(F) \geq 2^8$. If for every $H_i \in \text{Partition}(H)$, $\sum_{j \in H_i} 2^j \cdot \text{uncov}\left(B_{H_i \setminus \{j\}}^F, B_j^F\right)$ is bounded above by $\frac{\text{LOPT}\left(B_{H^*}^F\right)}{2^{11} \cdot \log\log \text{rank}(F)}$, then there exists a critical tuple $(Block, Good, Bad)$ such that, for every $i \in \mathbb{Z}$,

1) $Good(i)$, $Bad(i)$ and $Block(i)$ are subsets of H,
2) $2 \cdot \text{rank}\left(B_i^F\right)^{\frac{9}{8}} \geq \text{rank}\left(B_{Good(i)}^F\right)$, and
3) sum over every $j \in \mathbb{Z}$ of 2^j times $\text{uncov}\left(B_{Bad(j) \cup Block(j) \setminus \{j\}}^F, B_j^F\right)$ minus 2^j times $\text{uncov}\left(B_{Good(j) \setminus \{j\}}^F, B_j^F\right)$ is at least $\frac{\text{LOPT}\left(B_{H^*}^F\right)}{2^{17} \cdot \log\log \text{rank}(F)}$.

Due to lack of space we only give the beginning of the proof.

Proof: Let $\mathcal{T}$ be a critical-tree with a root labeled by H. We note that, by Lemma 42, such a critical-tree exists. Let $\mathcal{Q}$ be the family of all the leaves in $\mathcal{T}$. Let $\mathcal{Q}_{useful}$ be the family of all the useful sets in $\mathcal{Q}$. Define $\mathcal{Q}_{burnt}$ and $\mathcal{Q}_{negligible}$ in the same manner.

We construct a tuple $(Block, Good, Bad)$ as follows: for each $K \in \mathcal{Q}_{useful}$, we pick an arbitrary useful $K^* \subset K$ and then, for each $i \in K^*$, we let $Block(i) = K^*$, $Bad(i) = M_H(K)$ and $Good(i) = K \cup M_H(K)$. For every i such that $Block(i)$ is not defined in the previous process, we let $Block(i) = Good(i) = Bad(i) = \emptyset$.

The rest of the proof follows from the construction $(Block, Good, Bad)$ and the definitions of burnt, negligible, splittable, useful and critical-tree. ■

VII. Main Result

The following lemma enables the combination of the concentration result and the structural.

Lemma 44: If the event of Theorem 26 holds and $\max\{val(e) \mid e \in U\} < 2^{-64} \cdot \text{LOPT}(U)$, then

1) for every $j \in Super$, $\text{rank}(F) \geq \text{rank}\left(B_j^F\right) \geq \frac{3}{8} \cdot \text{rank}(B_j) \geq 2^{14}$.
2) $Valuable \subseteq Super$, and

3) $\text{LOPT}\left(B^F_{Valuable}\right) \geq \frac{3}{4} \cdot \text{LOPT}(F) \geq \frac{1}{4} \cdot \text{LOPT}(U)$.

The proof is omitted due to lack of space. The following lemma is used in order to bound the influence of the deviation in the concentration results.

Lemma 45: Let $K \subseteq Super$. If $\max\{val(e) \mid e \in U\} < 2^{-64} \cdot \text{LOPT}(U)$ and $\max K \leq \log \text{LOPT}(F) - 64 \cdot \log\log \text{rank}(F)$, then $8 \cdot \sum_{i \in K} 2^i \cdot \text{rank}(B_i)^{\frac{3}{4}} \leq \frac{\text{LOPT}(U)}{2^{20} \cdot \log\log \text{rank}(F)}$.

The proof is omitted due to lack of space.

Theorem 46: The Main Algorithm is Order-Oblivious, Known-Cardinality and has a competitive-ratio of $O(\log\log \text{rank}(U))$.

We note that the Main Algorithm is Known-Cardinality, since (i) the computation in Gathering stage is independent of the matroid elements; (ii) the computation in the Preprocessing stage; and (iii) the Selection stage uses only elements of the matroid that have been revealed. We also note that the Main Algorithm is Order-Oblivious, because the analysis depends on the elements in the sets F and $U \setminus F$ but not on their order.

The proof is a combination of the main theorems in this paper and the other two results in this section. It is omitted due to lack of space.

ACKNOWLEDGMENT

We would like to thank Ilan Newman for useful discussions and Trevor Fenner, Sven Helmer, George Loizou, Ilan Newman and Ron Peretz for helping to edit and verify. We would also like to thank the anonymous reviewers and in particular reviewer C for their suggested simplifications to proofs.

REFERENCES

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

[2] Pablo D Azar, Robert Kleinberg, and S Matthew Weinberg. Prophet inequalities with limited information. In *Proceedings of the 45th symposium on Theory of Computing*, pages 123–136. ACM, 2013.

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

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

[5] Siddharth Barman, Seeun Umboh, Shuchi Chawla, and David L. Malec. Secretary problems with convex costs. In *ICALP (1)*, pages 75–87, 2012.

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

[7] Niv Buchbinder, Kamal Jain, and Mohit Singh. Secretary problems via linear programming. *Mathematics of Operations Research*, 2013.

[8] Sourav Chakraborty and Oded Lachish. Improved competitive ratio for the matroid secretary problem. In *SODA*, pages 1702–1712, 2012.

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

[10] Michael Dinitz. Recent advances on the matroid secretary problem. *ACM SIGACT News*, 44(2):126–142, 2013.

[11] Michael Dinitz and Guy Kortsarz. Matroid secretary for regular and decomposable matroids. *CoRR*, abs/1207.5146, 2012.

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

[13] M. Feldman, J. Naor, and R. Schwartz. Improved competitive ratios for submodular secretary problems. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 218–229, 2011.

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

[15] Shayan Oveis Gharan and Jan Vondrák. On variants of the matroid secretary problem. In *ESA*, pages 335–346, 2011.

[16] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: offline and secretary algorithms. In *WINE*, pages 246–257, 2010.

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

[18] Patrick Jaillet, José A. Soto, and Rico Zenklusen. Advances on matroid secretary problems: Free order model and laminar case. *CoRR*, abs/1207.1333, 2012.

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

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

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

[22] James G Oxley. *Matroid theory*, volume 3. Oxford university press, 2006.

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