Matroids, Secretary Problems, and Online Mechanisms

Moshe Babaioff* Nicole Immorlica† Robert Kleinberg‡

Abstract

We study a generalization of the classical secretary problem which we call the "matroid secretary problem". In this problem, the elements of a matroid are presented to an online algorithm in random order. When an element arrives, the algorithm observes its value and must make an irrevocable decision regarding whether or not to accept it. The accepted elements must form an independent set, and the objective is to maximize the combined value of these elements. This paper presents an $O(\log k)$-competitive algorithm for general matroids (where k is the rank of the matroid), and constant-competitive algorithms for several special cases including graphic matroids, truncated partition matroids, and bounded degree transversal matroids. We leave as an open question the existence of constant-competitive algorithms for general matroids. Our results have applications in welfare-maximizing online mechanism design for domains in which the sets of simultaneously satisfiable agents form a matroid.

1 Introduction

Online mechanism design concerns markets in which agents arrive and depart over time. In this paper we consider the goal of welfare-maximization[1] and seek truthful mechanisms which provide a guaranteed performance with respect to this goal.

As in the offline counterpart, such online mechanism design problems suffer from two key difficulties: computational constraints and incentive constraints. Progress toward proving positive theoretical results in the online setting has been considerably slower than in the offline setting, largely because both of these constraints are considerably more severe in the online setting:

Computational constraints. The model of online computation imposes severe restrictions on the class of allocation rules which can be implemented. Even for the simplest imaginable online mechanism design problem — a single-item auction — the corresponding algorithmic problem (selecting the maximum element in a worse case sequence) has no non-trivial online approximation algorithm due to its online constraints.

Incentive constraints. It is exceedingly rare for an online algorithm to achieve exactly the optimum value of its objective function in the worst case. (At best, one hopes for a constant competitive ratio.) This means that, in general, one cannot design truthful mechanisms using the VCG paradigm. In fact, it has been shown that even for a simple expiring-goods model where constant-competitive allocation rules exist, there is no truthful online mechanism achieving a non-trivial approximation to the optimum social welfare [12].

Recently, it has been observed that these constraints are considerably less severe when one assumes that agents arrive in random order. For example, the algorithmic problem corresponding to the single-item auction then becomes the famous *secretary problem* introduced by Dynkin [4], i.e. the problem of selecting the maximum element in a *randomly-ordered* sequence. This observation leads to a rich interplay between secretary problems and online mechanism design [8, 11]. On one hand, existing algorithms for secretary problems can be transformed into truthful[2] online mechanisms which are constant-competitive for agents with random arrival order. On the other hand, the goal of designing online mechanisms for unit-demand multi-item auctions has led to the formulation and solution of new multiple-choice secretary problems interesting in their own right. Can we extend these techniques and design truthful constant-competitive mechanisms for online domains with richer combinatorial constraints on the feasible allocations?

This question has not been answered to date because of a lack of algorithms for these sorts of generalized secretary problems. Despite the rich literature on generalizations of secretary problems (see [7] for a survey), we are not aware of any previous work on multiple-choice secretary problems in which there is a non-trivial combinatorial structure constraining the sets which may be simultaneously selected. In this paper, we formulate a generalization of the secretary problem, which we call the *matroid secretary problem*, and

*UC Berkeley School of Information. Supported by the National Science Foundation grant number ANI-0331659. moshe@sims.berkeley.edu.

†Microsoft Research. nickle@microsoft.com

‡UC Berkeley Computer Science Division and Cornell University Department of Computer Science. Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship. rdk@cs.cornell.edu

[1] Another commonly studied objective is revenue-maximization.

[2] These mechanisms are truthful under the assumption that agents can lie arbitrarily about their value but only unidirectionally about their arrival time.

design algorithms and truthful mechanisms for this problem and its special cases. In a matroid secretary problem, the elements of a matroid are presented to an online algorithm in random order. As each element arrives and reveals its value, the algorithm must make an irrevocable decision whether to select the element, with the constraint that the set of all selected elements must be an independent set in the underlying matroid. The objective is to maximize the combined value of the selected elements. The case of uniform matroids corresponds to the multiple-choice secretary problem considered in [11]; that paper provides a constant-competitive algorithm for that special case.[3] In this paper, we design an $O(\log k)$-competitive algorithm for general matroids (where k is the rank of the matroid), and give constant-competitive algorithms for several special cases including graphic matroids, transversal matroids of bounded left degree, and their truncations. We leave open the question of whether there is a constant-competitive secretary algorithm for general matroids.

Connecting these results to mechanism design, we define a notion of *matroid domains*, a special case of *single-value domains* [1] which themselves generalize single-minded combinatorial auctions [13, 15]. In matroid domains, there is a set of outcomes, each agent is characterized by a set of *satisfying outcomes* and a value for receiving one of these outcomes, and for every profile of types the sets of agents who can be simultaneously satisfied constitute a matroid. A key example of a matroid domain is the *unit-demand domain* corresponding to transversal matroids, in which there is a finite set of goods for sale and each agent wants to receive one good from a specified subset of the set of all goods. For each of the algorithmic results quoted above, we provide an accompanying truthful mechanism for the corresponding matroid domain.

It is also worth mentioning the thematic connection between our work and recent work on random sampling methods for approximate revenue maximization in truthful offline mechanisms [3, 5, 6, 9]. In those works, the mechanism randomly partitions the set of agents and sets prices for the agents in one piece of the partition based on information derived from the "sample set" consisting of the other pieces of the partition. Similarly, our mechanisms sample a constant fraction of the input without making any decisions and then use information from this sample to guide future decisions. While sampling constitutes an important shared technique between our paper and [3, 5, 6, 9], most of their techniques are not directly applicable to the online setting, because the algorithm must contend with the fact that the second piece of the partition (i.e. the set of elements remaining after the sampling phase of the algorithm) arrives online.

The rest of this paper is organized as follows. Section 2 gives precise definitions of the algorithmic model (the matroid secretary problem) and the strategic model (matroid domains). Section 3 presents a log-competitive algorithm for general matroids as well as a discussion of whether constant-competitive algorithms exist in general. Sections 4 and 5 give constant-competitive algorithms for the transversal and graphic matroids, respectively. Finally, Section 6 defines a *truncation* operation which generalizes the notion of supply constraints to the matroid setting, and notes that if a matroid domain has a constant-competitive secretary algorithm, then every truncation of that domain has a constant-competitive secretary algorithm. Each of the algorithms we describe can be implemented by a truthful mechanism.

2 Model

The main technical results in this paper address algorithms for the *matroid secretary problem*, a generalization of the classical secretary problem. In the *matroid secretary problem*, there is a matroid[4] with ground set $\mathcal{U}$ and independent sets $\mathcal{I}$, and a weight function assigning a weight $w(i)$ to each element $i \in \mathcal{U}$. We wish to design an algorithm which given the matroid structure[5] $(\mathcal{U}, \mathcal{I})$ (but not the weights $w(i)$) selects online an independent set of (approximately) maximal weight in the following setting. The ground set of the matroid is presented in random order to the online algorithm. The algorithm maintains a set S of selected elements. When an element i arrives, the algorithm learns the weight $w(i)$ of the element. If $S \cup \{i\}$ is an independent set, the algorithm may choose whether to select i (i.e., $S \leftarrow S \cup \{i\}$). Note, the algorithm must decide whether to select an element before the next element arrives, and it is not allowed to later discard selected elements. The goal of the algorithm is to output a final selected set S of maximum weight. If the expected weight of the selected set (over a uniform random ordering of elements) is always within a c factor of the weight of the maximum weight basis for any assignment of weights to the ground set, we say the algorithm is c-competitive or a c-approximation. We call such an algorithm a *matroid secretary algorithm* and the problem it solves is the *matroid secretary problem*. Although we chose not to specify computational efficiency as part of our definition, it is worth noting that all algorithms we present in this paper are polynomial in the succinct representation of the matroid. Note this is a

[3] A key distinction between [8, 11] and ours is that their mechanisms are *temporally* strategyproof in the sense that agents can not gain by overstating their arrival times and/or understating their departure times. In our setting, agents depart immediately after arriving and hence our mechanisms are trivially temporally strategyproof.

[4] A *matroid* $(\mathcal{U}, \mathcal{I})$ is constructed from a ground set $\mathcal{U} \neq \emptyset$ and a nonempty family of subsets of $\mathcal{U}$, called the *independent* subsets of $\mathcal{U}$, such that if $B \in \mathcal{I}$ and $A \subseteq B$ then $A \in \mathcal{I}$ ($\mathcal{I}$ is hereditary). Additionally, if $A, B \in \mathcal{I}$ and $|A| < |B|$, then there is some element $x \in B \setminus A$ such that $A \cup \{x\} \in \mathcal{I}$ (exchange property).

[5] As noted throughout the text, many of our algorithms work in a setting where the matroid structure is revealed online as well.

generalization of the *classical secretary problem* introduced by Dynkin [4] in which the independent sets are precisely the sets of singletons (i.e. the uniform matroid of rank 1). Throughout this paper we will make the simplying assumption of *well-behaved inputs*, meaning that elements of the matroid have distinct values unless their value is 0. The assumption is essentially without loss of generality, because any algorithm which achieves competitive ratio c on well-behaved inputs can be easily transformed into an algorithm which is $(1+\varepsilon)c$-competitive on *all* inputs (the proof is deferred to the journal version).

The matroid secretary problem can be used to design online mechanisms for *matroid preference domains*, a special case of *single-value preference domains*. In *single-value preference domains*, there is a set $\mathcal{U}$ of n agents with preferences over a set Ω of possible outcomes. The preference domain is *single-valued* if each agent i has a *value* $v_i \in \Re_+$ and a *satisfying set* $A_i \subseteq \Omega$ of outcomes such that agent i obtains value v_i from outcomes in A_i and 0 from outcomes in $\Omega \setminus A_i$. We will occasionally use $x_i : \Omega \to \{0,1\}$ as an indicator function for the set A_i (i.e., $x_i(\omega) = 1$ if and only if $\omega \in A_i$). We assume the value v_i of an agent is private information known only to him[6]. A set of agents $S \subseteq \mathcal{U}$ is *independent* if there is an outcome $\omega \in \Omega$ that satisfies exactly the agents in S (i.e., $x_i(\omega) = 1$ if and only if $i \in S$). A single-value preference domain is a *matroid domain* if for any profile of types the family of independent sets of agents form a matroid over the set $\mathcal{U}$ of all agents. Similarly, given a matroid, it is possible to define the corresponding matroid domain where the agents are the ground set of the matroid, the satisfying outcomes for an agent are the independent sets which include the corresponding element, and the value v_i of an agent i is the weight $w(i)$ of the corresponding element i in the matroid.

Matroid domains are of particular economic interest. For example, transversal matroids correspond to preference domains in which agents have unit demand, are indifferent between a subset of goods, and are unsatisfied by the remaining goods as might be the case when allocating condos in a complex to agents with preferences of the sort "I want to live on the first floor" or "I want a south-facing window". Some examples of matroid domains include:

Domain 1: Selling k Identical Items There are n agents, and k identical items. Each agent wants to buy a single item, and agent i has a value of v_i if he gets an item. An outcome is a set S of at most k agents (winners), such that each agent in S gets an item. The underlying matroid of the preference domain is then the *uniform matroid* of rank k.

Domain 2: Unit-Demand Domain There are n agents, and a set M of m non-identical items. Each agent i has a set $T_i \subseteq M$ of desired items, that is, agent i has a value of v_i if he gets an item $j \in T_i$. An outcome is one-to-one matching of agents to items. The underlying matroid of the preference domain is then the *transversal matroid*[7].

Domain 3: Gammoids In the gammoid matroid domain, the agents correspond to sources in a graph, all of which wish to be routed to a common sink. A set S of sources is independent if there exist internally vertex-disjoint paths routing each source in S to the sink.

Domain 4: Graphical Matroids In the graphical matroid domain the agents are the edges of an undirected graph $G = (V, E)$. A set of edges is independent if it does not contain a cycle in the graph.

Domain 5: Truncated Partition Matroids There are n agents, and a set M of m non-identical items. Each agent wants to buy exactly one of the items in M (and is not satisfied by any other element of M), but the seller is allowed to sell only $k \leq m$ items. The underlying matroid of the preference domain is then a *truncated partition matroid* of rank k.

In the online setting, agents arrive in a random order. When an agent arrives, he announces a value v_i'. The mechanism then must commit to either choosing an outcome $\omega \in A_i$ or an outcome $\omega \notin A_i$ as well as a price p_i. After all agents have arrived, the mechanism chooses a final outcome ω which satisfies all prior commitments. Alternatively, one could describe the mechanism as choosing online an independent set S of agents and a price p_i for each agent $i \in \mathcal{U}$. Our goal is to design mechanisms which are *truthful* (each agent maximizes his *utility* $v_i \cdot x_i(\omega) - p_i$ by announcing his true value v_i for any declaration of the others and any ordering of agents) and *maximize the social welfare* (the final outcome ω is the one which maximizes the sum $\sum_{i \in N} v_i \cdot x_i(\omega)$). Our final mechanisms will fall short of this goal and instead approximately maximize the social welfare. We say a mechanism is *c-competitive* or a *c-approximation* if for any profile of types the expected social value of the selected outcome is within a factor of c of the maximum social welfare. Typically (and in all cases derived in this paper), a c-competitive algorithm for the matroid secretary problem implies a c-competitive online mechanism for the corresponding matroid preference domain.

[6]The domain described here is called the *known* single-value domain as the set A_i of desired outcomes for agent i is assumed to be public knowledge (creating a single-parameter domain). It is also possible to define the *unknown* single-value domain in which the set A_i is private knowledge, but coming from publicly known family of possible sets. We consider the unknown setting in Section 4.

[7]A distinction can be made based on whether the set T_i of desired items for agent i is private knowledge. The algorithms we develop for this domain are truthful regardless of this distinction. See Section 4 for further discussion.

3 An Algorithm for Matroid Domains

Our work focuses on matroid domains. Intuitively, matroid domains are more likely to be tractable than general set systems since in the offline setting a simple greedy algorithm selects the welfare-maximizing solution (i.e. the maximum weight basis). In fact, our hypothesis is that *any* matroid domain has a constant-competitive algorithm.

QUESTION 1. *Is there an algorithm which is constant competitive for every matroid domain?*

It can be shown (see Appendix A) that various intuitively natural algorithms fail to be constant-competitive. For example, it is impossible to achieve a constant competitive ratio using any algorithm which observes elements until it reaches a (possibly random) stopping time τ, sets a threshold value at time τ, and selects every subsequent element which is independent of the previous selections and exceeds the threshold value. Another intuitively natural algorithm observes a constant fraction of the elements (the "sample") without making any selections. Afterward, it keeps track of an independent set which is initially a maximum-value independent subset of the sample. Whenever it is possible to improve the value of the independent set by incorporating the current element (and possibly swapping out one of the elements of the sample) the algorithm selects the current element and incorporates it into the independent set. This algorithm, too, fails to be constant-competitive.

We do not yet know how to settle Question 1, but we present a series of results supporting an affirmative answer which are interesting in their own right. First, we observe that the assumption of matroid domains is essential. Namely, for some hereditary set systems, there is no constant-competitive algorithm. (In a matroid, choosing one suboptimal element can exclude at most one element of the maximum-weight basis from being selected in the future. In general set systems, this property does not hold; a single early mistake can exclude a large number of elements of the optimum from being selected afterward.) Next we show that there is an algorithm which is logarithmically competitive for any matroid domain. The following sections present improved algorithms with constant competitive ratio for specific matroid domains of particular economic or combinatorial interest.

We remark that there are alternative assumptions which also generalize the multiple choice secretary problem of [11] to matroids: one could assume that the set of n numerical values are assigned to the matroid elements using a random one-to-one correspondence (the "random assignment" model) but that the elements are presented in an adversarial order, or that both the assignment of values and the ordering of the elements in the input are random (the "random order and random assignment" model). Question 1 appears to be non-trivial in all of these cases.

3.1 Lower bound for general set systems

For an integer n with $k = \lfloor \ln(n) \rfloor$, let $(\mathcal{U}, \mathcal{I})$ be the set system defined as follows. The set $\mathcal{U}$ consists of n elements partitioned into $m = \lceil n/k \rceil$ subsets $S_1, \ldots, S_m$, each having k or $k-1$ elements. A set $A \subseteq \mathcal{U}$ belongs to $\mathcal{I}$ if and only if it is contained in one of the pieces of the partition, S_i. Suppose that we assign independent random values in $\{0, 1\}$ to the elements of $\mathcal{U}$ such that for each x, $w(x) = 1$ with probability $1/k$, 0 with probability $1 - 1/k$.

THEOREM 3.1. *The expected value of the maximum-weight set in $\mathcal{I}$ is $\Omega(\log n / \log\log n)$. For any randomized online algorithm to select a set in $\mathcal{I}$, the expected value of the set selected when the elements are presented to the algorithm in random order is less than 2.*

Proof. Suppose the algorithm makes its first selection at time t, and that it chooses an element $x \in S_i$. All future selections must be elements of S_i. Let T_i be the subset of S_i consisting of elements which have not yet been observed at time t. The values of the elements of T_i are independent of all the information observed up to time t; there are less than k elements in T_i and each of them has expected value $1/k$, so the expected combined value of all remaining elements which the element could potentially select after time t is less than 1. The only element selected up to time t is x, whose value is at most 1. This proves that the expected value of the set selected by the algorithm is less than 2.

The proof that the expected value of the maximum-weight set in $\mathcal{I}$ is $\Omega(\log n / \log\log n)$ is similar to a standard balls-in-bins calculation [14]; we include it here to make the exposition self-contained. Let $j = \lfloor k/(2\ln(k)) \rfloor$, and let E_i denote the event that at least j elements of S_i have value 1. The probability of E_i is at least $(1/k)^j \geq (1/\ln n)^{\ln n / 2 \ln\ln n} = 1/\sqrt{n}$. Since the events E_i are independent for $i = 1, 2, \ldots, m$, the probability that none of them occur is at most $(1 - 1/\sqrt{n})^m = o(1)$. If at least one event E_i occurs, then the maximum-weight element of $\mathcal{I}$ has weight at least $j = \Omega(\log n / \log\log n)$. □

3.2 Logarithmically competitive algorithm for matroid domains

The above result demonstrates that without imposing any structure on the preference domain, it is impossible to achieve constant competitive algorithms. Unfortunately, we do not know how to prove that constant-competitive algorithms exist for general matroid domains. However, it is not hard to see that the following simple algorithm is $O(\log k)$-competitive for any matroid domain where k is the rank of the matroid.

Threshold Price Algorithm

1. Observe $s = \lceil n/2 \rceil$ elements without picking any element, and let $S \subset \mathcal{U}$ be the set of observed elements (S is called the *sample*).

2. Let $l^* \in S$ be the element of S with maximum weight: $l^* = \text{argmax}_{l \in S}(w(l))$. Pick a random number j between 0 and $\lceil \log k \rceil$. The *threshold price* will be the weight of l^*, $w(l^*)$, divided by 2^j.

3. Initialize the set of selected elements B to be the empty set.

4. Let l_t be the element in $\mathcal{U} \setminus S$ observed at time $t = s+1, \ldots, n$. If $w(l_t) \geq w(l^*)/2^j$ and $l_t \cup B$ is an independent set, then select l_t (i.e., $B := B \cup l_t$).

THEOREM 3.2. *The threshold price algorithm is $32\lceil \log k \rceil$-competitive for any matroid domain where k is the rank of the matroid.*

Proof. Let B^* denote the max-weight basis of the matroid, consisting of elements $x_1, ..., x_k$ with values $v_1, ..., v_k$, such that $v_1 \geq v_2 \geq ... \geq v_k$. Let q be such that $v_q \geq v_1/k$ and either $q = k$ or $v_{q+1} < v_1/k$. Note that the elements of B^* whose value is less than v_1/k sum up to less than v_1, so they contribute less than half the value of B^*; hence $v_1 + v_2 + \ldots + v_q$ is more than half the value of B^*.

For any set $A \subset \mathcal{U}$, let $n_i(A)$ denote the number of elements of A whose value is at least v_i and let $m_i(A)$ denote the number of elements of A whose value is at least $v_i/2$. The sum of the q largest values of elements of B^* is then

$$\left[\sum_{i=1}^{q-1} (v_{i+1} - v_i) n_i(B^*) \right] + v_q n_q(B^*).$$

Let B be the independent set output by the threshold price algorithm. The value of B is then at least

$$(1/2) \cdot \left[\sum_{i=1}^{q-1} (v_{i+1} - v_i) m_i(B) \right] + (1/2) \cdot v_q m_q(B).$$

Thus, in order to prove that the threshold price algorithm is $32(\log k)$-competitive, it suffices to prove that the expected value of $m_i(B)$ is within a $8(\log k)$ factor of $n_i(B^*)$ for all $i \in 1, ..., q$. (Recall we lost a factor of two by comparing the value of B with $v_1 + \ldots + v_q$ instead $v_1 + \ldots + v_k$.) The case $i = 1$ is a special case. With probability $1/4$ the sample does not contain the maximum-weight element but does contain the element with the second-highest weight. Conditional on this event, with probability $1/\log(k)$ this second-highest weight becomes the threshold price, in which case the algorithm is guaranteed to select the element with weight v_1. Thus $\mathbb{E}(m_1(B)) \geq 1/(4 \log k)$ whereas $n_1(B^*) = 1$.

Assume from now on that $i > 1$, and note that $n_i(B^*) = i$ by definition. We condition on the event E that the sample contains the maximum-weight element in the matroid and that j, the threshold-setting parameter in step 2 of the algorithm, is such that $w(l^*)/2^j$ is the maximum element less than or equal to v_i in the set $\{w(l^*), w(l^*)/2, \ldots, w(l^*)/2^{\lceil \log k \rceil}\}$. Under the assumption that $v_q \geq v_1/k$, for every i such a j exists, and the algorithm has a $1/\log(k)$ probability of selecting this j. Thus event E has probability $1/(2 \log k)$. Given event E, there is an independent set A of size at least i each of whose values exceeds the threshold price (namely, $\{x_1, \ldots, x_i\}$). In expectation, at least $(i-1)/2 \geq i/4$ elements of A appear in the second half of the input. By the exchange property, this implies that the expected value of $|B|$ conditioned on event E is at least $i/4$.[8] As every element of B has value exceeding the threshold price which is at least $v_i/2$, $m_i(B) = |B|$ conditioned on event E. Removing the conditioning, we see that the expected value of $m_i(B)$ is at least $(1/8 \log k)$ times $n_i(B^*)$ for each i. □

REMARK 1. *The threshold price algorithm, as stated, takes the rank of the matroid as an input. However, at the cost of a constant factor in the competitive ratio, the algorithm can be modified so that it does not need to know the value of k at the start (and thus the entire matroid structure may be revealed online). Instead, it can estimate k to be twice the rank of the matroid induced on the sampled elements. This is never more than twice the true rank of the matroid, and with constant probability is it at least equal to the true rank of the matroid. A formal proof of this observation is deferred to the full version.*

4 The Unit-Demand Domain

In this section we consider the unit-demand domain. In this domain there are n agents, and a set M of m non-identical items. Each agent i has a set $T_i \subseteq M$ of desired items and receives a value of v_i for any item $j \in T_i$. We assume that there is a constant d such that $|T_i| \leq d$, for all i, that is, each agent desires one of at most d items. An outcome is mapping of agents to items, such that each agent is matched to at most one item. In this case A_i are all the outcomes in which i is matched to an item in T_i. We assume the values v_i are private information. If the sets T_i are private information as well, we say the domain is an *unknown single-value* (USV) domain.

The unit-demand domain is a matroid domain in which independent sets of agents form a *transversal matroid* of bounded left-degree. The matroid elements of a *transversal matroid* correspond to vertices on the left side (L) of a bipartite graph $G = (L, R, E)$ (thus the size of the ground

[8]The exchange property states that if A and B are two independent sets and A has more elements than B, then there exists an element in A which is not in B and when added to B still gives an independent set.

set is $|L| = n$). A set $S \subseteq L$ is *independent* if there is a perfect matching of S to nodes in R. The unit-demand domain is a transversal matroid domain in which L is the set of agents, R is the set of items, and there is an edge from $l \in L$ to $r \in R$ if $r \in T_l$. The bound on the number of items an agent desires translates to a bound of d on the maximal degree of any node in L. The value of an agent l corresponds to the weight of the node $l \in L$ and is denoted by $w(l)$.

We first present a $4d$-approximation algorithm to the matroid secretary problem for transversal matroids with left-degree at most d; we later show that this algorithm also creates a truthful online mechanism for the USV unit-demand domain. The algorithm is as follows.

Price Sampling Algorithm:

- Observe $s = \lceil n/2 \rceil$ elements without picking any element, and let $S \subset L$ be the set of observed elements (S is called the sample). For a right node $r \in R$, let $l_s^*(r) \in S$ be the sampled left node with maximal weight that is a neighbor of r. Let the price of $r \in R$ be $w(l_s^*(r))$.
- At time $t = s+1, \ldots, n$ we observe element $l \in L$ with weight $w(l)$. Let $R^*(l)$ be the set of unmatched neighbors of l with price lower than $w(l)$. If $R^*(l)$ is not empty, match l to the node with the lowest price in $R^*(l)$.

THEOREM 4.1. *For any transversal matroid with bounded left degree d, the above algorithm is a $4d$-approximation.*

Proof. Let OPT be a maximum weight matching in the graph, with weight $w(OPT) = \sum_{l \in OPT} w(l)$. For a right node $r \in R$, let $w(m(r))$ be the weight of the element that is matched to r in OPT (0 if no node is matched). Note that $w(OPT) = \sum_{r \in R} w(m(r))$. Additionally, for each right node $r \in R$, let $h(r)$ be the neighbor of r with maximal weight (w.l.o.g. r has neighbors). Let $H = \{h(r) | r \in R\}$ and let $w(H) = \sum_{l \in H} w(l)$.

CLAIM 1. $w(OPT) \leq d \cdot w(H)$.

Proof.

$$\begin{aligned} w(OPT) &= \sum_{r \in R} w(m(r)) \\ &\leq \sum_{r \in R} w(h(r)) \\ &\leq d \cdot \sum_{l \in H} w(l) \\ &= d \cdot w(H) \end{aligned}$$

where $\sum_{r \in R} w(h(r)) \leq d \cdot \sum_{l \in H} w(l)$ as each $l \in H$ appears at most d times in $\sum_{r \in R} w(h(r))$. □

CLAIM 2. $w(H) \leq 4 \cdot E[w(ALG)]$, *where ALG denotes the set of elements selected by the price sampling algorithm.*

Proof. For each $l \in H$ we prove that with probability at least $1/4$, l is matched by ALG. This implies that the expected weight of ALG is $E[w(ALG)] \geq \sum_{l \in H} 1/4 \cdot w(l) = 1/4 \cdot w(H)$.

For $l \in H$, let $r \in R$ be a right node with $h(r) = l$ (l is the highest weight neighbor of r). We show that with probability at least $1/4$, l comes after the sample and the only element that can be matched to r is $l = h(r)$. If r has only one neighbor, this clearly holds (with probability $1/2$, l comes after the sample). Otherwise, let $s(r)$ be the neighbor of r with second to maximal weight.

Let A and B be the events that $h(r)$ was not sampled and $s(r)$ was sampled, respectively. Then $Pr(A \wedge B) = Pr(B) \cdot Pr(A|B) = \frac{s}{n} \cdot \frac{n-s}{n-1} = \frac{\lceil n/2 \rceil}{n} \cdot \frac{n - \lceil n/2 \rceil}{n-1} > 1/4$. Thus with probability at least $1/4$, $h(r)$ was not sampled and $s(r)$ was sampled. This implies that with probability at least $1/4$, $l_s^*(r) = s(r)$ and when $h(r)$ arrives after the sample, r is unmatched. This means that $r \in R^*(h(r))$ and thus $R^*(h(r))$ is not empty, therefore when $l = h(r)$ arrives after the sample, it will be matched by the algorithm. □

By the two claims we derive that $w(OPT) \leq d \cdot w(H) \leq 4d \cdot E[w(ALG)]$ which concludes the proof of the theorem. □

We next observe that the above algorithm creates a truthful mechanism for the unit demand matroid domain (even without the bounded degree assumption), in the USV case in which the set of desired items is private information.

THEOREM 4.2. *For any unit demand matroid domain with a bound of d on the number of items an agent desires, the above is a truthful mechanism for the USV model, and it achieves $4d$-approximation to the social welfare.*

Proof. We need to show that the mechanism is truthful, both with respect to the value and with respect to the set of desired items. Clearly, any agent in the sample has no incentive to lie, as his utility is 0 for any declaration. An agent that is not in the sample is facing the following problem: given prices for each item, pick a desired item that has minimal price and is not taken yet, and pay its price. Clearly being truthful about this item will maximize the agent's utility (he answers a demand query). □

Note that the mechanism need not solicit the value and desired items of agents not in the sample. Such agents can simply be presented with item prices and be allowed to pick the item which maximizes their utility.

5 Graphic Matroids

In this section we consider *graphic matroids*. In a *graphic matroid*, a matroid element corresponds to an edge $e \in E$

of an undirected graph $G(V, E)$. A set of edges $S \subseteq E$ is *independent* if it does not contain a cycle. We denote $|E| = n$.

We next present a 16-approximation algorithm for this family of matroids based on a modification of the algorithm for transversal matroids. Given a graph $G = (V, E)$, we create a bipartite graph $G' = (L, R, E')$ by mapping each edge $e = (v, u) \in E$, to a left node which we denote by $n_{vu} \in L$. We map each node $u \in V$, to a right node $u \in R$. This creates a one-to-one mapping from edges and nodes in G to left nodes and right nodes in G', respectively. For each node $n_{uv} \in L$, there are two edges $(n_{vu}, v) \in E'$ and $(n_{vu}, u) \in E'$. G' is a bipartite graph with left degree 2.

Any tree T in G corresponds to a matching in the G': each edge $e = (v, u) \in T$ is matched to a node as follows. We pick a node r and look at the tree as rooted at r. Assume that for an edge $e = (v, u) \in T$, u is closer to r then v, then we match the edge to v. On the other hand, note that a matching in G' might correspond to a cycle in G: given a triangle, fix an orientation and match each edge to its left node. This implies that if we ran the algorithm for the transversal matroids on G', we might get a matching in G' that corresponds to cycle in G.

To overcome this problem we modify the algorithm for transversal matroids as follows: if an edge $(v, u) \in E$, that corresponds to a left node $n_{vu} \in L$ will close a cycle in G (when added to the already matched edges in G), it cannot be matched (even if it beats the price of one of its endpoints).

THEOREM 5.1. *For any graphic matroid the algorithm is a* 16*-approximation. Moreover, the algorithm with the defined payments creates a truthful mechanism.*

The proof is analogous to the proof for unit-demand domains in Section 4, except that there is an additional step required to bound the amount of value lost due to disallowing cycles. The details are deferred to the full version.

6 Truncations of matroids

Truncation is an operation which decreases the rank of a matroid by throwing away all independent sets whose cardinality exceeds a specified limit. In a matroid domain, this corresponds to a sort of *limited-supply* assumption: we modify the outcome set by eliminating all outcomes which satisfy more than a specified number of agents. A motivating example is a *truncated partition matroid*, i.e. the matroid whose ground set is a set $\mathcal{U}$ partitioned into subsets $\mathcal{U}_1, \ldots, \mathcal{U}_k$ and whose independent sets are all the subsets which have at most r elements and intersect each piece of the partition in at most one element. Truncated partition matroids correspond to the following natural selection problem: a department has the opportunity to hire up to r new faculty members, subject to the constraint that no two new hires should belong to the same subfield.

DEFINITION 1. *Let $\mathfrak{M} = (\mathcal{U}, \mathcal{I})$ be a matroid of rank k, and let r be a number less than or equal to k. The truncation $\tau_r(\mathfrak{M}) = (\mathcal{U}, \tau_r(\mathcal{I}))$ is the matroid whose collection of independent sets consists of all sets in $\mathcal{I}$ with at most r elements. If $\mathbb{D}$ is a matroid domain with outcome set Ω, then $\tau_r(\mathbb{D})$ is the domain obtained by deleting all outcomes $\omega \in \Omega$ which satisfy more than r agents.*

The following theorem provides a general reduction from the secretary problem for a matroid $\mathfrak{M}$ to the secretary problem for any truncation of $\mathfrak{M}$, at the cost of a constant factor in the competitive ratio. Since we know, for example, that partition matroids have an e-competitive algorithm (run an independent copy of the original secretary algorithm on each piece of the partition) this implies in particular that truncated partition matroids have a constant-competitive algorithm.

THEOREM 6.1. *Let $\mathfrak{M} = (\mathcal{U}, \mathcal{I})$ be a matroid. If* ALG *is a c-competitive algorithm for the $\mathfrak{M}$-secretary problem, then there is another algorithm $\tau_r(\mathsf{ALG})$ for the $\tau_r(\mathfrak{M})$-secretary problem whose competitive ratio is at most $\max(13c, 400)$. If $\mathbb{D}$ is a matroid domain and there is a truthful mechanism for $\mathbb{D}$ with approximation ratio c and allocation rule* ALG*, then there is a truthful mechanism for $\tau_r(\mathbb{D})$ with approximation ratio at most $\max(13c, 400)$ and with alloction rule $\tau_r(\mathsf{ALG})$.*

Proof. Without loss of generality, we may assume that ALG never selects an element whose value is 0. (Otherwise we may modify ALG, without changing its competitive ratio, by throwing away elements with zero value.) We will prove the theorem by reducing the $\tau_r(\mathfrak{M})$-secretary problem to the $\mathfrak{M}$-secretary problem using Karger's matroid sampling theorem [10]. This reduction will be achieved using an online procedure for mapping any input instance to a modified instance in which each element's value is either unchanged or reduced to 0. With constant probability, the modified input instance will satisfy three properties:

1. The value of its maximum-weight basis is at least a constant fraction of the value of the original maximum-weight basis of $\tau_r(\mathfrak{M})$.
2. The number of elements with nonzero modified value is at most r.
3. Conditional on the modified values of all elements, the order in which they appear in the modified input instance is random.

The second property (along with our assumption that ALG never selects an element of value 0) ensures that the set of elements selected by ALG will be independent in $\tau_r(\mathfrak{M})$. The first and third properties (along with our assumption that ALG is c-competitive) ensure that the expected value of the elements in this set is a constant fraction of the value of the maximum-weight basis of $\tau_r(\mathfrak{M})$.

We begin by recalling the following pair of definitions from [10].

DEFINITION 2. *If T is an independent set in a matroid $\mathfrak{M}$, and x is any element of the ground set of $\mathfrak{M}$, we say x* improves *T if x belongs to the maximum-weight independent subset of $T \cup \{x\}$.*

DEFINITION 3. *If X is a set and $p \in [0,1]$, then $X(p)$ denotes the random subset of X obtained by sampling each element independently with probability p.*

The following theorem is proved by Karger in [10]:

THEOREM 6.2. *Let $\mathfrak{M}$ be a matroid with ground set $\mathcal{U}$ in which each element has been assigned a real-valued weight. If $p > 0$ and T is a maximum-weight independent subset of $\mathcal{U}(p)$ then for any number $\varepsilon > 0$, the probability that more than $(1+\varepsilon)r/p$ elements improve T is at most* $\exp\left(-\varepsilon^2 r/2(1+\varepsilon)\right)$.

Finally, we shall need the following form of the Chernoff bound, which can be found in [14].

THEOREM 6.3. *Let $X_1, X_2, \ldots, X_m$ be independent Bernoulli random variables. Then for $X = \sum_{i=1}^{m} X_i$, $\mu = E[X]$, and $0 < \varepsilon \leq 1$,*

$$\Pr[X < (1-\varepsilon)\mu] < \exp(-\mu\varepsilon^2/2).$$

Let u be a sample from the binomial distribution $B(n, 3/4)$. The subset $U \subseteq \mathcal{U}$ consisting of the first u elements observed in the input has the same distribution as the random subset $\mathcal{U}(3/4)$. (Both subsets have cardinality distributed as $B(n,3/4)$, and both of them are uniformly-distributed conditional on their cardinality.) Let B be the maximum-weight independent subset of U in $\tau_r(\mathfrak{M})$, and let A denote the set of all elements of $\mathcal{U} \setminus B$ that improve B in $\tau_r(\mathfrak{M})$. Define the modified value of an element $x \in \mathcal{U}$ to be:

$$\hat{w}(x) = \begin{cases} w(x) & \text{if } x \in A \\ 0 & \text{otherwise} \end{cases}.$$

Let $\mathsf{OPT}_r(\mathfrak{M})$ denote the maximum-weight basis of $\tau_r(\mathfrak{M})$. The elements of $\mathsf{OPT}_r(\mathfrak{M})$ improve all independent sets [10]; this fact is an easy consequence of the matroid axioms. Consequently $\mathsf{OPT}_r(\mathfrak{M}) \subseteq A \cup B$, and the elements of $\mathsf{OPT}_r(\mathfrak{M})$ are randomly assigned to either A or B, independently, with probabilities $1/4$ and $3/4$, respectively. This implies the following:

1. The probability that $|A \cap \mathsf{OPT}_r| < r/6$ is at most $\exp(-r/72)$. (By Theorem 6.3 with $\mu = \frac{r}{4}, \varepsilon = \frac{1}{3}$.)
2. The probability that $|B \cap \mathsf{OPT}_r| < r/2$ is at most $\exp(-r/24)$. (By Theorem 6.3 with $\mu = \frac{3r}{4}, \varepsilon = \frac{1}{3}$.)
3. The probability that $|A \cup B| > 3r/2$ is at most $\exp(-r/144)$. (By Theorem 6.2 with $\varepsilon = \frac{1}{8}$.)

Assume $r > 144$ — otherwise we can simply run Dynkin's secretary algorithm to select the maximum-weight element of $\mathcal{U}$ with probability at least $1/e$, and this algorithm will have a competitive ratio of at most $144e$, which is less than 400. Given that $r > 144$, it means that the probability that any of the events (1)-(3) occur is at most $\exp(-2) + \exp(-6) + \exp(-1) < 0.51$. Let $\mathcal{E}$ denote the event that none of (1)-(3) occur. Note that $\mathcal{E}$ implies $|B| \geq r/2$ and $|A \cup B| \leq 3r/2$, hence $|A| \leq r$. For any integer a, conditional on the event $|A \cap \mathsf{OPT}_r| = a$, the set $|A \cap \mathsf{OPT}_r|$ is uniformly distributed over all a-element subsets of OPT_r and therefore $\mathbb{E}(w(A \cap \mathsf{OPT}_r) \mid a) = (a/r)w(\mathsf{OPT}_r)$. Hence the expected value of the set $A \cap \mathsf{OPT}_r$, conditional on $\mathcal{E}$, is at least $w(\mathsf{OPT}_r)/6$.

Suppose that the elements of $\mathfrak{M}$ are presented to ALG with their modified values $\hat{w}(x)$, in random order. Let T denote the set of elements selected by ALG and let R denote the maximum-weight independent subset of A. (Note that both of these sets consist of elements whose modified value is equal to their original value, so $w(T) = \hat{w}(T)$ and $w(R) = \hat{w}(R)$.) Since ALG is c-competitive, we have $\mathbb{E}(w(T) \mid A) \geq w(R)/c$. We also have $\mathbb{E}(w(R) \mid \mathcal{E}) \geq \mathbb{E}(w(A \cap \mathsf{OPT}_r) \mid \mathcal{E}) \geq w(\mathsf{OPT}_r)/6$. Also, $\mathcal{E}$ has probability at least 0.49, and conditional on $\mathcal{E}$, ALG selects at most r elements. Combining these observations, we find that the expected combined value of the first r elements selected by ALG is at least $\left(\frac{0.49}{6c}\right) w(\mathsf{OPT}_r)$, which is at least $w(\mathsf{OPT}_r)/13c$.

It remains to show that there is an online algorithm (the *simulation algorithm*) which can observe the elements of $\mathfrak{M}$ (with their original values) in random order, and can present these elements to ALG with their modified values, also in random order, subject to the constraint that every element x with $\hat{w}(x) > 0$ is presented to ALG before the simulation algorithm observes any of the elements of $\mathfrak{M}$ following x. (This timing constraint is necessary because if ALG decides to select an element, we want to be able to select it before observing any of the subsequent elements in the input.) We accomplish this simulation using a random shuffling trick. Let $z_1, z_2, \ldots, z_n$ denote a sequence of independent Bernoulli random variables, each with expected value $3/4$. Let u denote the number of i such that $z_i = 1$. The simulation algorithm observes the first u elements of the input without selecting anything, and it places these elements into a set U. It also computes B, the maximum-weight independent subset of U in $\tau_r(\mathfrak{M})$. Now it presents a sequence of elements x_t (for $t = 1, 2, \ldots, n$) to ALG as follows: if $z_t = 1$ then it samples x_t uniformly at random from U, deletes this element from U, and presents it to ALG with value $\hat{w}(x_t) = 0$. If $z_t = 0$ it observes the next element x in its own input sequence, computes whether this element improves B in $\tau_r(\mathfrak{M})$, and presents x to ALG with value $\hat{w}(x) = w(x)$ if x improves B, $\hat{w}(x) = 0$ otherwise. It is an

exercise to show that the modified values $\hat{w}(\cdot)$ defined in this paragraph are identical with those defined earlier, and that conditional on the modified value function $\hat{w}$, the elements of $\mathfrak{M}$ are presented to ALG in a random order.

If M is a truthful mechanism for domain $\mathbb{D}$, with allocation rule ALG, then ALG must be monotone (an agent cannot go from winning to losing by increasing her bid). The reduction from $\tau_r(\mathsf{ALG})$ to ALG preserves monotonicity, so $\tau_r(\mathsf{ALG})$ will also be monotone. Then we can design a truthful mechanism $\tau_r(M)$ using allocation rule $\tau_r(\mathsf{ALG})$; the price charged to agent i is 0 if the outcome does not satisfy i, otherwise it is the minimum value that i would have to bid in order to get a satifying outcome.

□

References

[1] Moshe Babaioff, Ron Lavi, and Elan Pavlov. Mechanism design for single-value domains. In *Proceedings of the 20th National Conference on Artifi cial Intelligence (AAAI 2005)*, pages 241–247, 2005.

[2] Moshe Babaioff and William E. Walsh. Incentive-compatible, budget-balanced, yet highly efficient auctions for supply chain formation. *Decision Support Systems*, 39(1):123–149, 2005.

[3] Maria-Florina Balcan, Avrim Blum, Jason D. Hartline, and Yishay Mansour. Mechanism design via machine learning. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*, pages 605–614, 2005.

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

[5] Uriel Feige, Abraham Flaxman, Jason D. Hartline, and Robert D. Kleinberg. On the competitive ratio of the random sampling auction. In *WINE*, pages 878–886, 2005.

[6] Amos Fiat, Andrew V. Goldberg, Jason D. Hartline, and Anna R. Karlin. Competitive generalized auctions. In *STOC*, pages 72–81, 2002.

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

[8] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and David C. Parkes. Adaptive limited-supply online auctions. In *Proc. 5th ACM conference on Electronic commerce*, pages 71–80. ACM Press, 2004.

[9] Jason D. Hartline and Robert McGrew. From optimal limited to unlimited supply auctions. In *ACM Conference on Electronic Commerce*, pages 175–182, 2005.

[10] David Karger. Random sampling and greedy sparsification for matroid optimization problems. *Mathematical Programming*, 82:41–81, 1998.

[11] R. Kleinberg. A multiple-choice secretary problem with applications to online auctions. In *Proc. 16th ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, pages 630–631, 2005.

[12] Ron Lavi and Noam Nisan. Online ascending auctions for gradually expiring items. In *SODA*, pages 1146–1155, 2005.

[13] Daniel J. Lehmann, Liadan O'Callaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49(5):577–602, 2002.

[14] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[15] Ahuva Mu'alem and Noam Nisan. Truthful approximation mechanisms for restricted combinatorial auctions. In *AAAI/IAAI*, pages 379–384, 2002.

A Some counterexamples

As evidence that Question 1 is non-trivial, we show two natural generalizations of the secretary algorithm can not be constant-competitive for general matroids.

A.1 Single-threshold algorithms

A single-threshold algorithm is any algorithm which has the following format. It observes the input without making any selections until it reaches a stopping time τ.[9] At time τ it computes a threshold value v which may depend on the portion of the input observed thus far. After time τ it selects every element x such that the value of x is at least v, and x is independent of the elements previously selected.

Let m be a large positive integer. Let $\mathfrak{M} = (\mathcal{U}, \mathcal{I})$ be a partition matroid consisting of n elements partitioned into $k = \lceil n/C \rceil$ subsets $S_1, S_2, \ldots, S_k$, each having size C or $C-1$. A set $A \subseteq \mathcal{U}$ belongs to $\mathcal{I}$ if $|A \cap S_i| \leq 1$ for all i. Assign values to the elements of $\mathcal{U}$ as follows. In S_i there is a single element (the *good element*) of value $1/i$ and all other elements (the *bad elements*) have value $1/Ci$. The value of the maximum-weight basis of $\mathfrak{M}$ is $H_k = \Omega(\ln(n/C))$.

Suppose we are given a single-threshold algorithm, and we run this algorithm on the input specified in the preceding paragraph. Let $\mathcal{E}(v)$ denote the event that the algorithm sets threshold value v. We will show that the expected value of the set selected by the algorithm, conditioned on event $\mathcal{E}(v)$, is at most $\left(\frac{2}{\sqrt{C}} + \frac{1}{C}\right) H_k + \ln C + 1$. First, the combined value of the bad elements selected by the algorithm is at most $\frac{1}{C} H_k$. Second, call a good element *safe* if its value is between v and Cv; otherwise call the good element *unsafe*. The combined value of the safe elements is at most $\sum_{1/v \leq i \leq 1/Cv} 1/i \leq 1 + \ln(\frac{1/v}{1/Cv}) = \ln C + 1$.

Finally we must bound the expected value of the good unsafe elements selected by the algorithm. Let S_i be any piece of the partition whose good element, x_i, is unsafe. Let $\mathcal{E}_i^1$ be the event that x_i is among the final $\lfloor \sqrt{C} \rfloor$ elements of S_i in the random ordering of the input. Note that $\Pr(\mathcal{E}_i^1) \leq \frac{1}{\sqrt{C}}$. If $\mathcal{E}_i^1$ does not occur, then either there are fewer than $\sqrt{C}$ elements of S_i which remain unobserved at time τ — in which case the algorithm has already failed to select x_i — or there are at least $\sqrt{C}$ elements of S_i which are

[9] Note we allow the possibility that the algorithm uses information about the elements observed up to time τ in deciding whether it has reached τ.

unobserved at time τ. In this latter case, let $\mathcal{E}_i^2$ be the event that x_i is the next element of S_i observed after τ. Note that $\Pr(\mathcal{E}_i^2|\overline{\mathcal{E}_i^1}) \leq 1/\sqrt{C}$. If neither $\mathcal{E}_i^2$ nor $\mathcal{E}_i^1$ occurs, then x_i will not be selected by the algorithm. This is because our assumption that x_i is unsafe implies that either $v > 1/i$, in which case the algorithm can not select x_i, or that $v < 1/Ci$, in which case the first element of S_i observed after time τ is selected instead of x_i. We conclude that the probability of selecting the unsafe element x_i is bounded above by $\Pr(\mathcal{E}_i^1 \vee \mathcal{E}_i^2)$, which is at most $\frac{2}{\sqrt{C}}$.

We have proven that for every constant C, every single-threshold algorithm has competitive ratio $\Omega(\sqrt{C})$ on the instance specified above. Hence there is no constant-competitive single-threshold algorithm.

A.2 The greedy algorithm

The natural generalization of the greedy algorithm for the matroid secretary problem is the following. The algorithm observes a constant fraction of the elements (the "sample") without making any selections. Afterward, it keeps track of an independent set which is initially a maximum-value independent subset of the sample. Whenever it is possible to improve the value of the independent set by incorporating the current element (and possibly swapping out one of the elements of the sample) the algorithm selects the current element and incorporates it into the independent set.

This algorithm is not constant-competitive, even when the matroid is a graphic matroid, and next we illustrate a counterexample: we present a family of graphs with size parametrized by m ($n = 2m + 1$), such that for any sufficiently large graph in the family (for large m), the algorithm is not C-competitive for the constant $C < 1$.

Let $G = (V, E)$ be a graph with vertex set $V = \{u, v, w_1, w_2, \ldots, w_m\}$ and edge set $E = \{(u, v)\} \cup \{(u, w_i), (v, w_i) \,|\, i = 1, 2, \ldots, m\}$. Assume we are given an arbitrarily small positive constant $\varepsilon > 0$. Assign weights to the elements of E by specifying that $w(u, v) = m + 1$, that $\varepsilon < w(u, w_i) < 2\varepsilon$ for each i, and that $2\varepsilon < w(v, w_i) < 3\varepsilon$ for each i. For the pair of edges (u, w_i) and (v, w_i), we call (u, w_i) the *light* edge of the pair, and (v, w_i) the matching *heavy* edge of the pair. If the algorithm does not select edge $e^* = (u, v)$ then its competitive ratio is at least $1/3\varepsilon$, since it can select at most $m + 1$ other edges and each of them has value less than 3ε.

To prove the claim we show that the probability of selecting e^* is arbitrarily small (tends to zero when m tends to infinity). The idea is to show that conditional on e^* not in the sample, the following event has high probability and ensures that e^* is not picked by the algorithm. The event is that the $n^{2/3}$ elements coming after the sample do not include e^* and include a pair of matching edges that are picked by the algorithm. To show that this event is very likely to happen we use the "Birthday Paradox" and show that with high probability the $n^{2/3}$ elements coming after the sample are going to include a pair of light and heavy edges, coming in this order, and no light edge that has higher weight than the light edge of maximum weight with matching heavy edge in the sample is observed (as this light edge is one of the $n^{1/6}$ maximum-weight light edges w.h.p.). This ensures that a pair of matching edges is picked by the algorithm, preventing e^* to be selected.