Recent Advances on the Matroid Secretary Problem

Michael Dinitz
Weizmann Institute of Science

Abstract

The matroid secretary problem was introduced by Babaioff, Immorlica, and Kleinberg in SODA 2007 as an online problem that was both mathematically interesting and had applications to online auctions. In this column I will introduce and motivate this problem, and give a survey of some of the exciting work that has been done on it over the past 6 years. While we have a much better understanding of matroid secretary now than we did in 2007, the main conjecture is still open: does there exist an $O(1)$-competitive algorithm?

1 Introduction

In this column, I will discuss some of the recent progress that has been made on variants of the *matroid secretary problem*. These problems combine interesting aspects of online algorithms and matroid theory, and can model real-life situations such as online auctions, so have recently received considerable attention from the theory community. As there has already been a great overview of some recent progress on secretary variants and how they relate to auctions and mechanism design [BIKK08], in this survey I will focus only on variants relating to matroids. Matroids have been studied extensively by the algorithms community, particularly with respect to how to optimize over them and over related structures (e.g. Edmonds' famous matroid intersection theorem [Edm70]). Only recently, though, have they been studied in the online setting, which (as always) introduces many complications. Hopefully this column will serve as a reasonably comprehensive survey of our current knowledge of these problems. By necessity I will not go into technical detail and will present no proofs, but I will try to give some intuition behind the algorithms and their analyses.

1.1 Secretary Problems

The classical secretary problem is the following. There is a set E of n elements, each of which has a value given by a function $w : E \to \mathbb{R}^+$. However, this value function is not given up front. The elements appear online in the order given by some bijection $\sigma : [n] \to E$, i.e. the first element that we see is $\sigma(1)$, then $\sigma(2)$, up to $\sigma(n)$, and we do not find out the value $w(u)$ of an element u until we see it. At time t, when $w(\sigma(t))$ is revealed, we must make an irrevocable decision whether or not to stop the process and select element $\sigma(t)$. Once an element is selected the process ends. The goal is to maximize the value of the element that we select, which is essentially equivalent to maximizing the probability that we select the most valuable element. Let $OPT(w)$ denote the most valuable element under value function w, and for a (possibly randomized) algorithm $\mathcal{A}$, let $\mathcal{A}(w, \sigma)$ denote the expected value of the element selected by $\mathcal{A}$ when run with value function w and ordering σ.

It is easy to see that if both w and σ are selected adversarially, then there is essentially nothing that we can do. If $\mathcal{A}$ is deterministic, then the adversary can simply select w and σ to force $\mathcal{A}$ to pick some element at some time (say t), and then immediately afterwards (at time $t+1$) present an element with extremely large value. If $\mathcal{A}$ is randomized, then the best it can do is the trivial algorithm of picking an element uniformly at random, which will pick the most valuable element with probability $1/n$.

So, in order to give nontrivial bounds, we need to weaken the problem a bit. The classical way of doing this is to assume that σ is a *random* ordering, rather than an adversarial ordering. So the adversary can still control the values, but once it has assigned values to the elements they are presented to us in uniformly random order. In this setting, we say that an algorithm $\mathcal{A}$ is α-competitive if $OPT(w)/\,\mathbb{E}[\mathcal{A}(w,\sigma)] \leq \alpha$ for all value functions w, where the expectation is taken over the random choice of σ and the internal randomness of $\mathcal{A}$.

With the extra flexibility afforded to us by forcing σ to be random, we can now give good algorithms. For example, suppose that we allow the first $n/2$ elements to go by without selecting them, and then we set a threshold $\tau = \max_{t \in [n/2]} w(\sigma(t))$ (the value of the best element we saw in the first half). We then select the next element we see with value at least τ. It is easy to see that this algorithm is 4-competitive: with probability $\frac{n/2}{n} \cdot \frac{n/2}{n-1} > 1/4$ the most valuable element is in the second half of the elements and the second most valuable element is in the first half. If this occurs, then τ is equal to the second-highest value, and the highest value is still in the second half, so we will select it. Thus we get the most valuable element with probability at least $1/4$, so the algorithm is 4-competitive. It turns out that a slight variation of this is optimal: if we set the threshold based on the most valuable of the first n/e elements (rather than $n/2$), the algorithm is e-competitive, which is known to be optimal (see e.g. [Dyn63]). This type of *sample-and-price* algorithm is extremely common, and variants of it will be used throughout this survey.

This problem is known as the secretary problem due to the story that is commonly used to describe it: we take the role of a corporate executive, and the elements are candidates for a single secretarial position. The candidates interview in random order, and we find out the value of a candidate upon interviewing him/her. Due to a tight labor market, any candidate we turn away will immediately find a job at a different firm and so will be unavailable, while due to business regulations once a secretary is hired s/he cannot be fired. Thus at the time of an interview we are forced to make an irrevocable decision as to whether or not to hire the candidate. Clearly this is exactly the secretary problem described above. Interestingly, this problem is also known by many other names, which correspond to other equivalent stories. In the sultan's dowry problem [Mos87], we think of a situation where a sultan has granted a commoner a chance to marry one of his n daughters, which are presented sequentially in random order for his consideration. The dowry of the daughter is told to the commoner only when she is presented, and his decision about each daughter is irrevocable. If he tries to maximize the size of the dowry he obtains, this is obviously equivalent to the secretary problem. The secretary problem has also been referred to as the fiancée problem, the marriage problem, the lottery problem, etc. (see [Fer89] for a good historical overview from a probability and statistics point of view).

1.2 Matroid Secretary

While many variants of the secretary problem were considered by the statistics and probability communities, a very basic version escaped consideration until recently. In a paper in SODA 2005, Bobby Kleinberg introduced a version in which instead of choosing a single element we are allowed to choose k elements, and the goal is to maximize the *sum* of their values [Kle05]. This "multiple-choice" secretary problem differed from the previous work in that previous work on selecting k

secretaries either had objective functions depending only on the *ranks* of the elements rather than their actual values (e.g. trying to maximize the probability that the k chosen are exactly the k best [Fre83]), or assumed that the values were drawn from some distribution rather than being assigned adversarially. Kleinberg's algorithm is a clever modification of the basic sample-and-price technique used in the classical version, with a recursive twist. If $k = 1$ then the classical algorithm is used. Otherwise, let m be a random sample from the binomial distribution $B(n, 1/2)$. We recursively apply the algorithm to the first m elements to select up to $\ell = k/2$ elements. We then set the threshold τ to the value of the ℓ'th most valuable element of the first m, and select any of the remaining $n - m$ elements with value larger than τ (until we have selected k elements in total). Kleinberg proved that this algorithm has competitive ratio of at most $\frac{\sqrt{k}}{\sqrt{k}-5}$. Note that this ratio tends toward 1 as k tends to infinity. The motivation for this problem comes from designing strategyproof mechanisms for online auctions, but in this survey we will not discuss the auction-theoretic motivations and results, as a previous survey by Babaioff, Immorlica, Kempe, and Kleinberg examines secretary problems from this point of view [BIKK08].

Kleinberg's paper was followed two years later by the seminal paper of Babaioff, Immorlica, and Kleinberg that introduced the matroid secretary problem [BIK07]. They made the observation that one could generalize Kleinberg's setting to enforce combinatorial constraints on allowable sets. Let $\mathcal{S} \subseteq 2^E$ be a collection of subsets of the elements. Then what if we only allow our algorithm to pick sets that are in $\mathcal{S}$, and still try to maximize the sum of the values of the chosen elements? In Kleinberg's setting $\mathcal{S}$ is just all subsets of size at most k. Their motivation was, again, auction theoretic: suppose we are selling items, and the elements are bidders who arrive online. Based on their preferences and desires, only certain subsets of the bidders can be simultaneously satisfied. These subsets form $\mathcal{S}$.

It turns out that many interesting auctions can be modeled by forcing $\mathcal{S}$ to be the independent sets of a matroid. While there are many equivalent definitions of matroids, we will use the common one based on independent sets:

Definition 1.1. *A matroid $\mathcal{M} = (E, \mathcal{I})$ is an ordered pair consisting of a finite set E and a collection $\mathcal{I}$ of subsets (called the* independent sets*) of E satisfying the following three conditions:*

1. $\emptyset \in \mathcal{I}$,

2. If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$, and

3. If I_1 and I_2 are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there is an element $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

The matroid secretary problem, as defined by [BIK07], is exactly what it sounds like. For a given matroid $\mathcal{M} = (E, \mathcal{I})$ and value function w, let $OPT(\mathcal{M}, w)$ denote the value of the most valuable set in $\mathcal{I}$ (where the value of a set is the sum of the values of its elements). For any algorithm $\mathcal{A}$ and ordering σ, let $\mathcal{A}(\mathcal{M}, w, \sigma)$ denote the value of the set returned by $\mathcal{A}$ (which must be an independent set). We say that $\mathcal{A}$ is α-competitive for matroid $\mathcal{M}$ if $OPT(\mathcal{M}, w)/\mathbb{E}[\mathcal{A}(\mathcal{M}, w, \sigma)] \leq \alpha$ for any value function $w : E \to \mathbb{R}^+$, where the expectation is taken over the uniformly random choice of σ and the internal randomness of $\mathcal{A}$. Babaioff, Immorlica, and Kleinberg defined this problem, and made the following conjecture:

Conjecture 1.2 ([BIK07])**.** *For every matroid $\mathcal{M}$, there is an algorithm that is $O(1)$-competitive.*

This is the main conjecture motivating the area, and is still essentially wide open. Note that while part of the initial motivation for the matroid secretary problem is from auctions, from a purely online algorithms point of view, matroids are an extremely natural class, since they exemplify the difficulty caused by irrevocable decisions. In particular, suppose that we had the power to later throw out elements that we chose when we initially saw them (maybe a choice to throw an element out is still irrevocable). In this setting, the matroid secretary problem would be trivial. We could simply use the obvious greedy algorithm: maintain an independent set B, and when we see a new element e we set B to be the maximum value independent set contained in $B \cup \{e\}$ (which might require throwing out some elements of B, or not choosing e). It is easy to verify that this will always give the optimum independent set. So studying the matroid secretary problem allows us to study the difficulty caused by the requirement that our choices be irrevocable.

The rest of this survey is organized as follows. First, I will review some basic matroid theory and definitions. I will then discuss in Section 2 the known results on Conjecture 1.2: an $O(\log r)$-competitive algorithm from [BIK07], and an $O(\sqrt{\log r})$-competitive algorithm due to Chakraborty and Lachish [CL12]. Here r denotes the *rank* of the matroid, which is equal to the cardinality of the largest independent set (note that by the third condition of Definition 1.1 all maximal independent sets have the same cardinality). In Section 3, I will go over some progress on general matroids that has been made under slightly different models: $O(1)$-competitive algorithms obtained by weakening the adversary or giving the algorithm more power, and an $O(\log r)$-competitive algorithm for the generalization where the value function is submodular. Finally, in Sections 4 and 5, I will go back to the normal model and give some $O(1)$-competitive algorithms for specific classes of matroids.

1.3 Basic matroid theory

Let $\mathcal{M} = (E, \mathcal{I})$ be a matroid. E is called the *ground set* of $\mathcal{M}$. Any set in $\mathcal{I}$ is called *independent*, and any set that is not independent is called *dependent*. A maximal independent set is called a *basis*, and a minimal dependent set is called a *circuit*. The *rank function* $r : 2^E \to \mathbb{N}$ is the function which maps $A \subseteq E$ to the size of the largest independent set contained in A, i.e. $r(A) = \max_{B \subseteq A : B \in \mathcal{I}} |B|$, and $r(A)$ is called the *rank* of A (which we will denote by r). The rank of the matroid is just $r(E)$, the rank of the set of elements. The span of a set A is the unique maximal superset with the same rank (the span is sometimes referred to as the *closure* [Oxl92]). The *density* of a set A is $|A|/r(A)$, and the density of a matroid is $\gamma(\mathcal{M}) = \max_{A \subseteq E} |A|/r(A)$. We typically define matroids by their independent sets, as in Definition 1.1, but their are equivalent definitions that give axioms for the circuits, the bases, the rank function, the span, etc.

Matroids were originally introduced by Whitney [Whi35] as a way of generalizing notions of independence that had been considered in linear algebra and in graph theory. Slightly more formally, he defined two classes of matroids and showed that they satisfy the matroid axioms.

Definition 1.3. *Let $G = (V, E)$ be an undirected graph. Let $\mathcal{I} = \{A \subseteq E : (V, A) \text{ is acyclic}\}$. Then $(E, \mathcal{I})$ is a* graphic *matroid, also known as the* cycle *matroid of G.*

Definition 1.4. *Let F be a field and let E be a set of elements of a vector space over F. Let $\mathcal{I} = \{A \subseteq E : A \text{ is linearly independent}\}$. Then $(E, \mathcal{I})$ is a* vector *matroid, and is said to be F-*representable*. If there is some field F for which $\mathcal{M} = (E, \mathcal{I})$ is F-representable then we say that $\mathcal{M}$ is* representable.

It is not obvious *a priori* that forests in graphs have anything to do with linearly independent sets in vector spaces, but Whitney was able to see that they are both examples of his more general definition of matroids. Since [Whi35] a rich theory of matroids has been developed (see e.g. [Oxl92] for an overview), and many other classes of matroids have been defined. The next two classes are extremely simple, and can easily be seen to be matroids.

Definition 1.5. *A matroid $\mathcal{M} = (E, \mathcal{I})$ is a* uniform *matroid if there is some integer k so that $I \in \mathcal{I}$ if and only if $|I| \leq k$.*

Definition 1.6. *A matroid $\mathcal{M} = (E, \mathcal{I})$ is a* partition *matroid if there is a partition $E_1, \ldots, E_k$ of E so that $I \in \mathcal{I}$ if and only if $|I \cap E_j| \leq 1$ for all $j \in [k]$.*

A natural class containing the partition matroids are the *laminar* matroids. A laminar family is a collection of sets with the property that if two sets intersect, one is actually a superset of the other. In a laminar matroid, each set in a laminar family is associated with a capacity, and a set is independent if the size of its intersection with each set in the family is at most the capacity:

Definition 1.7. *Let E be a set, let $\mathcal{F}$ be a laminar family over E, and for each $F \in \mathcal{F}$ let c_F be an arbitrary positive integer. Let $\mathcal{I} = \{I \subseteq E : |I \cap F| \leq c_F \text{ for all } F \in \mathcal{F}\}$. Then $(E, \mathcal{I})$ is a matroid, and is called a* laminar *matroid.*

It is easy to see that laminar matroids generalize partition matroids: a partition of E is clearly a laminar family, and we can set each capacity to 1 to recover the partition matroid.

A slightly less simple class are the transversal matroids, which are based on graphs but are about matchings rather than forests.

Definition 1.8. *Let $G = (L, R, E)$ be a bipartite graph, and let $\mathcal{I} = \{A \subseteq L : \exists \text{ an injective function } f_A : A \to R\}$. Then $(L, \mathcal{I})$ is a matroid, and is called a* transversal *matroid.*

In other words, a set A of left nodes is independent if and only if there is a matching that saturates A. It is less obvious that transversal matroids are indeed matroids, but it is not very hard to see [Oxl92, Theorem 1.6.2]. A *degree-d transversal matroid* is a transversal matroid in which all left nodes have degree at most d.

The final class of matroids we will consider may seem somewhat strange, but have been well studied by matroid theorists and play a key role in many structural results about matroids.

Definition 1.9. *A matroid is* regular *if it is F-representable for every field F.*

Matroids, like graphs, have a rich theory of minors. As in graphs, the two minoring operations are deletion and contraction. Let $A \subseteq E$. Then the deletion of A from $\mathcal{M}$, denoted by $\mathcal{M} \setminus A$, is the matroid on $E \setminus A$ in which a subset of $E \setminus A$ is independent if and only if it is independent in $\mathcal{M}$. An equivalent operation is *restriction*, where the restriction of $\mathcal{M}$ to A (denoted by $\mathcal{M}|A$) is just $\mathcal{M} \setminus (E \setminus A)$. An operation that is dual to deletion is *contraction*: the contraction of A from $\mathcal{M}$, denoted by $\mathcal{M}/A$, is the matroid on $E \setminus A$ in which a set $I \subseteq E \setminus A$ is independent if and only if there is a basis B of $\mathcal{M}|A$ such that $I \cup B$ is independent in $\mathcal{M}$. In fact, the choice of basis does not matter: let B be an arbitrary basis for $\mathcal{M}|A$. Then a set I is independent in $\mathcal{M}/A$ if and only if $I \cup B$ is independent in $\mathcal{M}$ (see [Oxl92, Proposition 3.1.8]). A matroid $\mathcal{M}'$ is a *minor* of $\mathcal{M}$ if it can be obtained from $\mathcal{M}$ by a series of minoring operations. The interested reader should take a look at the excellent (albeit slightly old) survey by Paul Seymour on matroid minors [Sey95].

2 Progress on General Matroids

In this section I will give a brief overview of the progress that has been made so far on Conjecture 1.2. In particular, I will discuss two algorithms for the matroid secretary problem, neither of which is $O(1)$-competitive. Recall that r is the rank of the given matroid.

Threshold Price Algorithm: This algorithm was the first algorithm designed for the matroid secretary problem, and is due to [BIK07]. It is another variation of the sample-and-price paradigm. They call the algorithm the *Threshold Price Algorithm*, and it works as follows. As in the classical setting, we first observe the first $n/2$ elements without selecting any of them. In the classical setting we would now set a threshold to $\tau = \max_{t \in [n/2]} w(\sigma(t))$, but instead we choose a value j uniformly at random from $\{0, 1, \dots, \log r\}$ and set $\tau = \max_{t \in [n/2]} w(\sigma(t))/2^j$. We initialize our chosen set B to the empty set, and then at time $t \in \{n/2+1, \dots, n\}$ we add the element $\sigma(t)$ to B if $w(\sigma(t)) \geq \tau$ and $B \cup \{\sigma(t)\} \in \mathcal{I}$.

Babaioff, Immorlica, and Kleinberg proved that this algorithm is $O(\log r)$-competitive. At a high level, their argument works as follows. Let OPT be $e_1, e_2, \dots, e_r$ with $w(e_1) \geq w(e_2) \geq \dots w(e_r)$. Without loss of generality we can assume that $w(e_r) \geq w(e_1)/2r$, since elements with smaller value can contribute total value at most 1/2 of OPT. Fix $i \in [r]$, and consider the number of elements in B (the set returned by the algorithm) with value at least $w(e_i)/2$. With probability $1/(2\log r)$, the most valuable element e_1 is in the first half of the elements and j is chosen to be the smallest integer so that $w(e_1)/2^j$ is at most $w(e_i)$ (and thus at least $w(e_i)/2$). By definition there are i elements of OPT of value at least $w(e_i)$, and in expectation $i/2$ of them are in the second half. So the second half contains an independent set of size $i/2$ made up of elements more valuable than threshold. It is easy to see that this fact and the matroid axioms (in particular, the third axiom) guarantee that in this case the algorithm will return an independent set of size at least $i/2$, all elements of which have value at least $w(e_i)/2$. Since this event holds with probability $1/(2\log r)$ for each $i \in [r]$, by linearity of expectations we get that the algorithm returns a set that in expectation has value at least $\Omega(1/\log r)$ times the value of OPT.

Theorem 2.1 ([BIK07])**.** *The Threshold Price Algorithm is $O(\log r)$-competitive*

Improvement to $O(\sqrt{\log r})$: After the publication of [BIK07], many papers were published on the matroid secretary problem and variants (some of which I will discuss later in this survey). But progress on the actual problem stalled, until a recent paper of Chakraborty and Lachish from SODA 2012 that improved the competitive ratio to $O(\sqrt{\log r})$ [CL12]. Their algorithm is extremely complex, and we will only give a very high level overview. To begin, with probability 1/2 they simply run the classical secretary algorithm. If there is an element that contributes at least a $1/\sqrt{\log r}$ fraction of OPT, then this gives the theorem. Otherwise, every element has value at most $OPT/\sqrt{\log r}$. Under this assumption, they begin as always by letting about half of the elements go by to create a sample. They then use this sample to construct a good estimate of the value of OPT, and using this estimate they partition the elements (past and future) into buckets: an element is in bucket i if its value is $OPT/2^i$ (they assume without loss of generality that OPT is a power of 2 and that all values are powers of 2). Since no element contributes much to OPT, any bucket that contributes significantly to OPT must have reasonably large rank. Thus for every bucket the sample will with high probability be a very good representation of the bucket, so they can estimate quantities such as the rank of the bucket.

With this ability, they first check whether there is a set of buckets M that are extremely nice. Let L_i denote the elements in bucket i, and for any set of buckets A let $L_A = \cup_{i \in A} L_i$. For a set of buckets M to be good, the main requirement is that $\sum_{i \in M} (rank(L_M) - rank(L_{M \setminus \{i\}})) \cdot \frac{OPT}{2^i} \geq \frac{OPT}{\sqrt{\log r}}$. Intuitively, this means that for an average bucket $i \in M$, a lot of the rank of the buckets in M comes from bucket i. So we can just add arbitrary elements from buckets in M (while maintaining global independence) with the guarantee that we will always be able to choose many elements from the average bucket, regardless of what happens in the other buckets.

If such a set of buckets M does not exist, then they use a different algorithm, which they call the *protection algorithm*. This algorithm is based on a different intuition. Suppose that we had a magic oracle that could build a basis for any bucket, including both past and future elements. Then a natural approach upon seeing an element e would be to use this oracle to build a basis for all buckets containing more valuable elements, and then test whether e is in the span of those bases and the already selected elements. If it is not, then we select e. This is a natural algorithm since it prevents valuable elements from being blocked by less valuable elements. Suppose that we are considering an element e, and let S denote the previously selected elements of smaller value. When we considered the elements of S, we only selected them if they were not in the span of the bases of the more valuable buckets, including the bucket i containing e. So the union of S and any independent set of bucket i (including e itself) must be independent. Thus e cannot be blocked by lower value elements that just happened to be earlier in the order.

Obviously we do not have such a magic oracle, since we do not know what bucket an element will be in until it appears. But what Chakraborty and Lachish show is that if there is no nice set of buckets, then two things happen: this protection algorithm would give good value if we had such an oracle, and we can construct an oracle that "approximately" constructs bases out of only elements that were in the sample. The definition of approximately constructing a basis is not straightforward, but intuitively means that it constructs an independent set that is almost as large as the basis. This turns out to be enough for the algorithm to be $O(\sqrt{\log r})$-competitive.

Theorem 2.2 ([CL12]). *There is an $O(\sqrt{\log r})$-competitive algorithm for the matroid secretary problem.*

3 Alternative Models

Since [BIK07], much of the work on the matroid secretary problem on general matroids falls into two categories: getting stronger upper bounds by weakening the adversary or allowing the algorithm extra power, or making the setting even more general while still proving comparable upper bounds to [BIK07]. In this section, I will discuss some examples of both of these.

3.1 Weakening the Adversary

One extremely interesting line of work was initiated by José Soto [Sot11], who weakened the adversary by forcing the value function w to *also* be random. In particular, he assumed that values are assigned as follows. The adversary chooses a set of n real values, but then the value function is a random bijection between the elements of the matroid and these values. So the adversary essentially has the power to choose whatever terrible set of values will make an algorithm bad, but cannot choose how these values are assigned to the elements. More formally, let $L \subset \mathbb{R}^+$ be any multiset of values with $|L| = n$, and let $\mathcal{M} = (E, \mathcal{I})$ be a matroid with $|E| = n$. Let $w : E \to L$

be a random bijection (drawn uniformly from the set of bijections), and let $\sigma : [n] \to E$ also be a uniformly random bijection. Then an algorithm $\mathcal{A}$ in this setting is α competitive for $\mathcal{M}$ if $\mathbb{E}_w[OPT(w, \mathcal{M})] / \mathbb{E}_{w,\sigma}[\mathcal{A}(\mathcal{M}, w, \sigma)] \leq \alpha$ (note that here the optimum value is a random variable depending on w). Soto calls this the *random assignment model*. Note that it is clearly a generalization of models where the values are chosen iid from some distribution (known or unknown to the algorithm). His main result was that in this model it is in fact possible to give $O(1)$-competitive algorithms for all matroids.

Theorem 3.1 ([Sot11])**.** *For every matroid $\mathcal{M}$, there is an algorithm which is $(2e^2/(e-1))$-competitive in the random assignment model.*

Soto's Algorithm: Soto's algorithm uses two main insights. First, he uses tools from matroid theory to decompose $\mathcal{M}$ into a set of *uniformly dense* matroids. A matroid $\mathcal{M}$ is uniformly dense if $\gamma(\mathcal{M}) = |E|/r(E)$, i.e. if $|E|/r(E) = \max_{A \subseteq E} |A|/r(A)$. Soto noticed that a tool from matroid theory known as the *principal sequence* easily gives a partition of $\mathcal{M}$ into uniformly dense minors of $\mathcal{M}$, which we will call $\mathcal{M}_1 = (E_1, \mathcal{I}_1), \ldots, \mathcal{M}_k = (E_k, \mathcal{I}_k)$. These minors are called the *principal minors* of $\mathcal{M}$, and have the nice property that the union of one independent set from each is independent in $\mathcal{M}$. So if $I_j \in \mathcal{I}_j$ for all $j \in [k]$ then $\cup_{j \in [k]} I_j \in \mathcal{I}$. Moreover, Soto proved that respecting this decomposition does not lose too much of the value. If $OPT(\mathcal{M}_i, w_i)$ denotes the value of the most valuable independent set in $\mathcal{M}_i$ under value function w_i (where w_i is just the restriction of w to $\mathcal{I}_i$), then $\sum_{j \in [k]} OPT(\mathcal{M}_i, w_i) \geq (1 - 1/e) OPT(\mathcal{M}, w)$. Note that this is true for all value functions w, so this step does not depend on the random assignment assumption.

With this decomposition in hand, it only remains to design a $2e$-competitive algorithm for uniformly dense matroids. Running such an algorithm on each principal minor in parallel would give an algorithm that returns an independent set with total value (in expectation) that is at least a $(1 - 1/e)/2e$ fraction of the expected value of the optimum. Thus the competitive ratio is $2e/(1 - 1/e) = 2e^2/(e-1)$, as claimed.

The algorithm for uniformly dense matroids is simple, but its analysis depends heavily on the random assignment assumption. Let r denote the rank of the uniformly dense matroid. We first sample half of the elements, and let T be the most valuable r elements seen so far. Now when we see a new element e, we test whether it is one of the most valuable r elements seen so far, and if so we add it to T (and remove the least valuable element in T, so T is always the most valuable r elements seen so far). If e does get added to T, then if the element that is removed from T is from the sample and we can add e to our output set while maintaining independence, then we select e. To analyze this algorithm note that since the set T has nothing to do with the matroid, but is instead just about the weights, we can think of T being an (adversarial) set of weights and then randomly assigning elements to these weights. It turns out that when looked at from this perspective, it becomes possible to prove (thanks to the uniform density of the matroid) that every one of the top r values will be chosen with reasonable probability.

Adversarial Order: While the algorithm for uniformly dense matroids is interesting and important, probably the major contribution of [Sot11] was showing that the decomposition into the principal minors essentially preserves the value for *every* value function. This gives us a powerful tool, since it means that we can change the setting and still give $O(1)$-competitive algorithms for general matroids as long as we can do it for uniformly dense matroids. This was realized by Oveis Gharan and Vondrák, who looked at the version of the matroid secretary in which the value function is still random (exactly as in Soto's random assignment model), but where the ordering σ is *adversarial*. In other words, their setting is the inverse of the classical setting, in which the ordering is random but the values are adversarial. They designed an $O(1)$-competitive algorithm for uniformly dense matroids in this model (the random-assignment adversarial-order model), which when combined with Soto's decomposition immediately yields the following theorem:

Theorem 3.2 ([GV11])**.** *For every matroid $\mathcal{M}$, there is an $O(1)$-competitive algorithm in the random-assignment adversarial-order model.*

Free Order Model: Theorem 3.2 essentially implies that as long as values are assigned at random, the problem is reasonably easy. But what if we go back to the classical setting of adversarial assignment? Is there some way of weakening the adversary slightly that would make the problem easy? This problem was very recently considered by Jaillet, Soto, and Zenklusen [JSZ13], who designed a constant-competitive algorithm in what they call the *free order* model. In this model the value assignment is completely adversarial, but instead of the ordering σ being random or adversarial, it is instead under the control of the algorithm. The algorithm is allowed to select the next element to consider after making a decision about the previous element. In other words, at any point in time t, the algorithm is allowed to select $\sigma(t)$ based on the structure of the matroid and the values $w(\sigma(1)), w(\sigma(2)), \ldots, w(\sigma(t-1))$ seen so far. But as before, values are revealed online and decisions are irrevocable.

Suppose we begin by sampling half of the elements to get a set A, i.e. we simply select $\sigma(1), \ldots, \sigma(n/2)$ uniformly at random. Let OPT_A denote the optimum solution for A. Define an element $e \notin A$ to be *good* if it can be used to improve OPT_A: either $OPT_A \cup \{e\}$ is independent, or there is some $f \in OPT_A$ with $w(e) > w(f)$ such that $OPT_A \cup \{e\} \setminus \{f\}$ is independent. A natural approach to the secretary problem would be to first construct A and OPT_A, and then select elements if they are good and can be selected while maintaining independence of the selected set. This approach immediately runs into problems in the classical setting: we might select low-value elements that "block" high-value elements that appear later. But by allowing ourselves to set the order in which the values of elements are revealed, we can try to avoid this.

The algorithm from [JSZ13] first samples A as described. Let $A = \{a_1, a_2, \ldots, a_m\}$ in decreasing order of value (where $m = n/2$), and let $A_i = \{a_1, \ldots, a_i\}$. Let $B = E \setminus A$ denote the matroid elements not in A. We divide the rest of the elements into groups by letting B_i be the elements in B that are in $span(A_i) \setminus span(A_{i-1})$. We let B_{m+1} denote the elements in B that are not in $span(A)$. Note that after sampling A, we know each B_i just from the structure of the matroid and the values of the elements in A. So we set the order so that after considering A we consider elements from B_1, then elements from B_2, etc., continuing through B_{m+1}. The order inside of each group is arbitrary. When considering an element of B_i, we select it if it has value larger than $w(a_i)$ (so swapping it in for a_i gives us extra value) and if we can select it while maintaining independence.

The analysis of this algorithm proves that every element in OPT is selected with probability at least 1/9. For any element $e \in OPT$, Soto shows that there exists some value $i \in [m]$ so that with constant probability $e \in span(A_i)$ (and thus will be considered in some phase $i' \leq i$) and simultaneously e is not in the span of the elements of B that have value at least $w(a_i)$. If this happens, and if e is not in A, then the algorithm will select e. To see this, let i' be the phase at which e is considered. Since e is in both $span(A_i)$ and in OPT, it must be the case that $w(e) > w(a_i)$. So the only reason the algorithm would not add e is if selecting it would make the selected set dependent. But all of the elements considered before e must have value at least $w(a_{i'}) \geq w(a_i)$

in order to be selected, and since e is not in the span of the elements of B with value larger than $w(a_i)$, adding it cannot make the selected set dependent. Thus we have the following theorem.

Theorem 3.3 ([JSZ13])**.** *For every matroid, there is a 9-competitive algorithm in the free order model.*

3.2 Submodular Matroid Secretary

In this extension, the value function w is no longer simply a function from elements to values. Instead $w : 2^E \to \mathbb{R}^+$ is a function from *sets* of elements to values. The only requirements are that w is nonnegative and submodular: $w(S) + w(T) \geq w(S \cup T) + w(S \cap T)$ for all $S, T \subseteq E$. Clearly this generalizes the standard matroid secretary problem, since linear functions (the normal setting) are submodular. Submodular functions are an extremely interesting class of functions, and a significant amount of work has been done on offline optimization of submodular functions. In the online secretary setting there are a few results, the main one being an $O(\log r)$-competitive algorithm for general matroids due to Gupta, Roth, Schoenebeck, and Talwar [GRST10]. As with the original matroid secretary problem, certain classes of matroids are known to admit $O(1)$-competitive algorithms: uniform matroids [BHZ10, GRST10, FNS11], partition matroids [GRST10, FNS11], and transversal and laminar matroids [MTW13]. Some of these results depend on the additional assumption that the value function is monotone, but we will focus on the [GRST10] result for general matroids which does not make this assumption.

The $O(\log r)$-competitive algorithm for general matroids is a subtle modification of the threshold price algorithm described in Section 2. As in the threshold price algorithm, the first half of the elements are sampled and a value j is chosen uniformly at random from $\{0, 1, \ldots, \log r\}$. The threshold τ is then set to $\frac{2}{5} \cdot \max_{t \in [n/2]} w(\sigma(t))/2^j$. Whereas in the threshold price algorithm we kept track of one set B, we will now keep track of two sets B_1 and B_2. We flip a fair coin at the beginning to decide whether to select the elements of B_1 or the elements of B_2, but we will keep track internally of the elements of both while only selecting the elements of the appropriate set. When we see an element $\sigma(t)$ for $t > n/2$, we check whether the marginal value it adds to B_1 is at least τ, i.e. whether $w(B_1 \cup \{\sigma(t)\}) - w(B_1) \geq \tau$. If it is (and adding it to B_1 maintains the independence of B_1), then we add $\sigma(t)$ to B_1. If not, then if adding it to B_2 maintains the independent of B_2 and the marginal value it adds to B_2 is at least τ, we add it to B_2. Otherwise we simply discard $\sigma(t)$.

To analyze this algorithm, let C^* denote the optimal set. Let $C^*_\tau \subseteq C^*$ denote the set obtained by adding the elements of C^* greedily in order of decreasing marginal value, and adding an element to C^*_τ if when it was added in this ordering it had marginal value at least τ. The first claim in the analysis of [GRST10], and the heart of their argument, is that $w(B_1) + w(B_2) \geq |C^*_\tau| \cdot \tau/10$. This means that by randomly choosing either B_1 or B_2 we get at least half of this value in expectation. They then show that if we sum this value over the possible choices of τ the total value is at least $\Omega(OPT)$. Thus in expectation the random choice of τ in the algorithm achieves value at least $\Omega(OPT/\log r)$.

4 Simple Matroid Classes

Along with proving Theorem 2.1, Babaioff, Immorlica, and Kleinberg also designed $O(1)$-competitive algorithms for some specific classes of matroids. In particular, they provided $O(1)$-competitive algorithms for uniform, partition, graphic, and bounded-degree transversal matroids. In this section, I will present a unification of their results due to Babaioff, Dinitz, Gupta, Immorlica, and Talwar [BDG$^+$09]. The following theorem will be the main goal of this section:

Theorem 4.1 ([BIK07])**.** *Any matroid that is uniform, partition, or graphic admits an $O(1)$-competitive algorithm. Any degree-d transversal matroid admits an $O(d)$-competitive algorithm.*

While [BIK07] presented totally different algorithms for each of the above classes, Babaioff et al. [BDG$^+$09] unified their arguments (and slightly improved their bounds) by showing that all of these classes can be reduced to partition matroids. For a value function $w : E \to \mathbb{R}^+$ and a matroid $\mathcal{M} = (E, \mathcal{I})$, recall that $OPT(\mathcal{M}, w)$ denotes the value of the max-value basis of $\mathcal{M}$.

Definition 4.2. *A matroid $\mathcal{M} = (E, \mathcal{I})$ has the α-*partition property *if we can randomly construct a partition matroid $\mathcal{M}' = (E', \mathcal{I}')$ with $E' \subseteq E$ such that for every value function w,*

- $\mathbb{E}[OPT(\mathcal{M}', w)] \geq 1/\alpha \times OPT(\mathcal{M}, w)$*, and*
- $\mathcal{I}' \subseteq \mathcal{I}$

This property allows us to reduce matroids to partition matroids: if a matroid has the α-partition property, then the first thing we do is (randomly) construct the partition matroid guaranteed by the property. We then use an $O(1)$-competitive algorithm for the partition matroid. The second part of the property guarantees that any independent set in the partition matroid that we might return is also independent in the original matroid, and the first part of the property guarantees that being competitive with the partition matroid is enough to be competitive with the original matroid. Note that there is a trivial e-competitive algorithm for partition matroids: we can simply run the classical secretary algorithm on each part of the partition. Combining this with Definition 4.2 gives us the following theorem.

Theorem 4.3. *If a matroid has the α-partition property, then there is an $e\alpha$-competitive algorithm for it.*

Thus it just remains to show that uniform, graphic, and constant-degree transversal matroids have the α-partition property for constant α. Uniform matroids are simple: we just create r parts of the partition (where r is the rank of the matroid) and for each element choose a distinct part to put it in uniformly at random. It is straightforward to argue that this construction satisfies the $e/(e-1)$-partition property. Transversal matroids are similar, and just require using the graph structure in the obvious way. For every right node in the bipartite graph we create a part of the partition, and then for each left node (i.e. element of the matroid) we assign it to the part of one of its neighbors uniformly at random. Clearly each element of the optimal solution gets assigned to the right node that it is matched with in the optimal solution with probability $1/d$, and this is enough to imply the d-partition property.

In order to illustrate one example of the partition property, we will show it for graphic matroids, which were shown by [BDG$^+$09] to have the 3-partition property. This was later improved by [KP09] who showed that they have the 2-partition property. Both of these results give algorithms with competitive ratio better than the original 16 proved by [BIK07].

Theorem 4.4 ([KP09])**.** *Graphic matroids have the 2-partition property, and thus admit $2e$-competitive algorithms.*

Proof. Let $G = (V, E)$ be the graph defining the graphic matroid $\mathcal{M}$. Arbitrarily order the vertices $v_1, v_2, \ldots, v_n$ of V. Let G_0 be the directed graph obtained by orienting every edge from the vertex with smaller index to the vertex with larger index, and let G_1 be the graph obtained by orienting every edge in the opposite direction. We create a bin for every vertex, then uniformly at random choose $X \in \{0, 1\}$. We then create the partition matroid $\mathcal{M}'$ by adding edge e to the bin corresponding to its tail in G_X. Clearly G_X does not contain any directed cycles, so since we pick at most one outgoing edge from each vertex, any set independent in $\mathcal{M}'$ is also independent in $\mathcal{M}$. To analyze the value, consider the optimum basis in the original matroid $\mathcal{M}$ (which corresponds to a forest F in G). By rooting each tree arbitrarily and directing every edge up towards the root, we can associate a value with every vertex (the value of the edge leaving it) so that the value of the basis is equal to the sum of the values of the vertices. Then in $\mathcal{M}'$, every edge in F has probability $1/2$ of being in the bin corresponding to its tail in F, in which case the value of the vertex in $\mathcal{M}'$ is at least as large as its value in $\mathcal{M}$. Thus in expectation $OPT(\mathcal{M}', w) \geq \frac{1}{2} OPT(\mathcal{M}, w)$. $\square$

5 Advanced Matroid Classes

In this section, we will discuss three classes of matroids that were not considered by Babaioff, Immorlica, and Kleinberg: transversal matroids (without the degree bound from the previous section), laminar matroids, and regular matroids. The algorithms and analyses for these classes are significantly more complex than for the simple classes of the last section, and do not fit into the partition property framework. On the other hand, they do not stray too far from the [BDG$^+$09] framework: both the transversal and the laminar algorithms can be seen as a reduction to a partition matroid that is only defined *after* the sample rather than at the beginning of the algorithm. This means that the reduction can take into account the knowledge of the values gained by the sample, which is crucial for both of them.

5.1 Transversal Matroids

The first class of matroids that was shown to admit $O(1)$-competitive algorithms (other than those from [BIK07]) were transversal matroids. $O(1)$-competitive algorithms for transversal matroids were given first by Dmitrov and Plaxton [DP08], who designed an algorithm that was later generalized by Korula and Pál [KP09] to hold for slightly more general settings (online vertex-at-a-time weighted matching, in particular). In order to define this algorithm, we will first define the obvious offline greedy algorithm. Recall that in a transversal matroid the elements of the matroid are the left vertices of a bipartite graph $G = (L, R, E)$, so each left vertex has some value (which we do not find out until they appear online). We first arbitrarily order the right vertices. The greedy algorithm, when given a collection of left vertices with their values (and edges), considers them in nonincreasing order of value. When considering left vertex u, it examines its neighborhood $N(u)$ of right vertices and finds the highest ranked neighbor that has not already been matched (if such a vertex exists). It then matches u to this vertex. It is easy to see that this greedy matching always matches left nodes with at least $1/2$ as much value as the optimal matching.

The online algorithm works as follows. We first choose a random value k from the binomial distribution $B(n, 1/2)$, and let the first k elements $\sigma(1), \ldots, \sigma(k)$ go by without selecting anything. As always, this set will be our sample. We call this set of elements A. Now when we see a new left vertex u, we first run the greedy algorithm on $A \cup \{u\}$, and let $g(u)$ denote the right-vertex that u

was matched to by the greedy algorithm (if it is indeed matched). Note that this is equivalent to running the greedy algorithm on just u and the elements of A with value larger than u. So $g(u)$ depends only on u and on elements with larger weight. If $g(u)$ is currently unmatched, then we select u and mark $g(u)$ as matched. Otherwise we do not select u.

This algorithm trivially returns an independent set, since we only ever select vertices that can be matched without modifying the previous matching. It is not too hard to see (at least intuitively) why this algorithm returns sets with large value. First, let $M_1 \subseteq A$ denote the sampled nodes that are matched when we run the greedy algorithm on A. Then since the greedy algorithm is a 2-approximation to the maximum matching, and about half of the elements are in the sample, the value of M_1 is at least $OPT/4$ in expectation. Now let $M_2 = \{u \notin A : g(u) \text{ exists}\}$ denote the left vertices not in A for which $g(u)$ exists. Since every vertex is equally likely to be in A as not to be in A, every element contributes equally to M_1 and to M_2 (we can think of determining whether $g(u)$ exists as happening before determining whether or not u was sampled), and thus M_2 has expected value at least $OPT/4$.

Let $u \in M_2$, and suppose that $g(u) = v$. We say that u is *good* if $g(u') \neq v$ for all $u' \in M_2$ with $u' \neq u$. Let M_3 denote all of the good elements of M_2. We do not lose much value when restricting to M_3, since it can be proved that in expectation, for any right node v, the total value of nodes u with $g(u) = v$ is only $2w(e)$, where e is the most valuable element with $g(e) = v$. On the other hand, v has probability $1/2$ of only one element u existing with $g(u) = v$. Putting this all together, we get that in expectation M_3 has value at least least $1/4$ of M_2 and thus at least $1/16$ of OPT. Finally, it is easy to see that any element of M_3 will be selected by the algorithm.

5.2 Laminar Matroids

In SODA 2011, Im and Wang [IW11] showed that laminar matroids admit $O(1)$-competitive algorithms. In particular the gave an algorithm which has competitive ratio at most $16000/3$. While their algorithm is reasonably simple, their analysis is quite complex. Recently Jaillet, Soto, and Zenklusen [JSZ13] gave a different algorithm, which is also quite simple and has competitive ratio at most $3e\sqrt{3}$. Even more recently, a simple "simulated greedy" algorithm inspired by the transversal matroid algorithm was shown by Ma, Tang, and Wang to be 9.6-competitive [MTW13]. In the rest of this section, I will describe the simple algorithm of [JSZ13] and give a sketch of the analysis that shows the competitive ratio is at most $27e/2$ (the improvement to $3e\sqrt{3}$ requires a more subtle analysis).

The first thing the algorithm does is rename the matroid: it is easy to see (say from representing the laminar family as a tree/forest) that there is a way of ordering the elements $e_1, \dots, e_n$ so that every $F \in \mathcal{F}$ is exactly an interval $e_i, \dots, e_j$ for some i and j. With this renaming of the elements, their algorithm begins as most matroid secretary algorithms do: they choose a threshold k from the binomial distribution $B(n, 2/3)$ and let the first k elements go by as a sample A (their modification of the probability in the binomial distribution from $1/2$ to $2/3$ is just to optimize the constant achieved). They then construct the optimum solution $OPT_A = \{e_{i_1}, e_{i_2}, \dots, e_{i_p}\}$ of the sample, where $1 \leq i_1 \leq i_2 \leq \dots \leq i_p \leq n$. Let $P_j = \{e_k : k \in \{i_{j-1}, \dots, i_j\}$, so the P_j's are the intervals between consecutive elements of OPT_A. They split these intervals into two sets: let $\mathcal{P}_{odd} = \{P_j : j \text{ is odd}\}$, and let $\mathcal{P}_{even} = \{P_j : j \text{ is even}\}$. Let $\mathcal{P} = \mathcal{P}_{odd}$ with probability $1/2$, and otherwise let $\mathcal{P} = P_{even}$. They then run the classical secretary algorithm on every $P_j \in \mathcal{P}$.

Note that this algorithm is similar to the reduction to partition matroids that we saw in Section 4, since running the classical secretary algorithm on each set in $\mathcal{P}$ is exactly the same as running the partition matroid algorithm on the partition matroid defined by $\mathcal{P}$. But there is one crucial difference: the partition matroid is only defined *after* the sampling. This ends up making it simple to prove that the set selected is actually independent. Suppose that our algorithm selects elements e_i and e_j with $i < j$. Then by construction, $e_i \in P_{i'}$ and $e_j \in P_{j'}$ for some $j' \geq i' + 2$. Thus OPT_A contains at least two elements e_a and e_b with $i < a < b < j$. So for any set $F \in \mathcal{F}$, if our algorithm returns a set I containing b_F elements of F, then $|OPT_A \cap F| \geq b_F$, and so $c_F \geq b_F$. Since this is true for any set $F \in \mathcal{F}$, this implies that I is in fact independent.

To see that this algorithm gives good value, consider some element f_i in the optimal solution, let f_j be the element of OPT with largest index less than i, and let f_k be the element of OPT with smallest index larger than i. Then with probability $(\frac{2}{3})^2 \cdot \frac{1}{3} \cdot \frac{1}{2} = \frac{2}{27}$ the following event happens: $f_j, f_k \in A$ and $f_i \notin A$ and the coin flip to determine $\mathcal{P}$ is such that the set P_ℓ containing f_i is in $\mathcal{P}$. If all of this happens, then by running the classical secretary algorithm on P_ℓ, with probability at least $1/e$ we get an element at least as valuable as f_i. Now linearity of expectations implies that the total value of our algorithm is in expectation at least $\frac{2}{27e}$ times the value of OPT.

5.3 Regular Matroids

When looking through the literature on the matroid secretary problem, it is hard not to notice one missing class of matroids: representable matroids (i.e. vector matroids). These matroids were one of the original motivations for Whitney's definition of matroids (along with graphic matroids). So it is somewhat surprising that until very recently, essentially no natural class of representable matroids was known to admit an $O(1)$-competitive algorithm (many classes such as graphic matroids are in fact representable, but they are more naturally considered as graphs rather than through linear algebra). Given this seemingly difficult class of matroids, a very natural place to start is the regular matroids. Recall that a matroid is regular if it is F-representable for every field F. Intuitively, regular matroids should be the "easiest" class of representable matroids, since they are so heavily restricted. And indeed, Dinitz and Kortsarz recently designed a good algorithm for regular matroids which does not extend to general representable matroids.

Theorem 5.1 ([DK13]). *There is a $9e$-competitive algorithm for the matroid secretary problem on regular matroids.*

The techniques used in [DK13] are more matroid-theoretic than algorithmic, and depend heavily on a famous theorem of Paul Seymour known as the *regular matroid decomposition theorem* [Sey80]. Informally, this theorem states that any regular matroid can be decomposed into a set of graphic matroids, cographic matroids, and copies of a special matroid known as R_{10} (cographic matroids are the duals of graphic matroids under the standard notion of matroid duality, where a set is independent in the dual if and only if its complement contains a basis of the primal). The notion of decomposition used by Seymour is too complex for this survey, but interestingly enough, $O(1)$-competitive algorithms for these classes were already known: we have discussed graphic matroids, cographic matroids were shown to have $O(1)$-competitive algorithms by Soto [Sot11], and any constant size matroid is trivial. So a natural approach to regular matroids is to use the existing algorithms for these three classes as black boxes, and show that they respect Seymour's decomposition. Unfortunately, it turns out that they do not, and so the bulk of [DK13] is not designing new online algorithms, but rather showing two offline constructions. First, we show that Seymour's decomposition theorem can be modified to give a decomposition with some extra properties that turn out to be useful. Second, we give a way of modifying all of the matroids in the decomposition so that the appropriate algorithm can be run on each of them while guaranteeing global independence and only a small loss in value.

It is beyond the scope of this survey to go into the details of [DK13], but it is worth noting that their use of the regular matroid decomposition theorem and Soto's use of the principal sequence [Sot11] represent essentially the only uses of matroid theory beyond the basic definitions. The vast majority of the results that have been presented in this survey, including the results on general matroids and the results on all other specific matroid classes, simply use the definition of matroids (or the subclass of matroids). Yet there is a deep and rich literature in structural matroid theory, including a deep theory of matroid minors which was useful in [DK13], that has remained largely untapped. If we hope to make progress on more advanced classes of matroids, it seems natural to start using some of this theory. We suggest that the interested reader look at Oxley's book [Oxl92] for a good general overview, and a survey of Seymour [Sey95] for an introduction to matroid minor theory.

6 Discussion and Directions

In this column, I have tried to present a reasonably comprehensive survey of the current results on the matroid secretary problem and its variants. While optimization over matroids is reasonably well understood in the offline setting, the online setting has proven to be much more difficult. Yet we have had a fair amount of success so far: an $O(\sqrt{\log r})$-competitive algorithm for the matroid secretary problem, and $O(1)$-competitive algorithms for many special classes such as partition, uniform, graphic, transversal, laminar, and regular matroids. In addition, we know that much of the difficulty comes from the precise adversarial value, random ordering assumptions: if the values are assigned randomly to the elements then there is an $O(1)$-competitive algorithm for general matroids, and if the assignment is still adversarial but the algorithm gets to control the order (rather than it being random) then there is also an $O(1)$-competitive algorithm for general matroids. So the classical assumptions, of random order and adversarial values, seem together to be the cause of the difficulty of the problem.

There are still many interesting open problems related to the matroid secretary problem. Obviously the big one is Conjecture 1.2: is there an $O(1)$-competitive algorithm for all matroids? Resolving this one way or the other would be extremely interesting. Assuming that this conjecture is either false or that we cannot make progress on it, designing $O(1)$-competitive algorithms for specific classes is also extremely interesting. For example, we still have a very poor understanding of representable matroids: there is an $O(1)$-competitive algorithm for regular matroids, but this algorithm does not seem to generalize even to binary matroids (i.e. $GF(2)$-representable matroids). Can we design a good algorithm for binary matroids? Ternary matroids? All representable matroids? In general, the algorithms that we know for specific classes have not used much of the structural matroid theory that has been developed by mathematicians over the last half-century – maybe we can make progress by using results from this literature (such as Seymour's regular matroid decomposition theorem)?

Another set of interesting problems are related to the differences between the current algorithms. For example, many of the known algorithms work by reducing to partition matroids, either implicitly or explicitly. The simple classes discussed in Section 4 do this explicitly before any elements have even been seen through the α-partition property, while the algorithms for laminar and transversal matroids use the values revealed in the sample to design the partition matroid. Is this use of the sample actually necessary? More specifically, do transversal and laminar matroids satisfy the α-partition property for some constant α? This is interesting, since it is still not known whether *all* matroids have the α-partition property for constant α. If they do, then this would resolve Conjecture 1.2 in the affirmative. More likely they do not, but at this point we do not know a single class of matroids that provably do not. Constructing such a class might be a good first step towards proving Conjecture 1.2 false.

Finally, one issue that I have glossed over throughout this survey is the question of when the algorithm receives knowledge of the matroid. We have generally been assuming that the algorithm receives knowledge of the matroid upfront, before any elements are revealed. This is what lets us, for example, apply Seymour's regular matroid decomposition theorem or apply the decomposition into the principal minors of the matroid. But what if the structure of the matroid is revealed online as the weights are? Maybe we cannot query whether a set is independent until we have seen all elements of that set. Some of the algorithms discussed above can easily be modified to work in this setting, but for some we do not know how to do this. Figuring out whether foreknowledge of the matroid is actually necessary is also an interesting open problem.

References

[BDG$^+$09] Moshe Babaioff, Michael Dinitz, Anupam Gupta, Nicole Immorlica, and Kunal Talwar. Secretary problems: weights and discounts. In *Proc. SODA*, pages 1245–1254, 2009.

[BHZ10] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Morteza Zadimoghaddam. Submodular secretary problem and extensions. In *Proc. APPROX/RANDOM*, pages 39–52, 2010.

[BIK07] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proc. SODA*, pages 434–443, 2007.

[BIKK08] Moshe Babaioff, Nicole Immorlica, David Kempe, and Robert Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exch.*, 7(2):7:1–7:11, June 2008.

[CL12] Sourav Chakraborty and Oded Lachish. Improved competitive ratio for the matroid secretary problem. In *Proc. SODA*, pages 1702–1712, 2012.

[DK13] Michael Dinitz and Guy Kortsarz. Matroid secretary for regular and decomposable matroids. In *Proc. SODA*, 2013.

[DP08] Nedialko B. Dimitrov and C. Greg Plaxton. Competitive weighted matching in transversal matroids. In *Proc. ICALP*, pages 397–408, 2008.

[Dyn63] E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Soviet Math. Dokl*, 4, 1963.

[Edm70] Jack Edmonds. Submodular functions, matroids, and certain polyhedra. In *Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf.)*, pages 69–87. Gordon and Breach, 1970.

[Fer89] Thomas S. Ferguson. Who solved the secretary problem? *Statistical Science*, 4(3):pp. 282–289, 1989.

[FNS11] Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. Improved competitive ratios for submodular secretary problems. In *Proc. APPROX/RANDOM*, pages 218–229, 2011.

[Fre83] Freeman. The secretary problem and its extensions: A review. *International Statistical Review*, 51(2):pp. 189–206, 1983.

[GRST10] Anupam Gupta, Aaron Roth, Grant Schoenebeck, and Kunal Talwar. Constrained non-monotone submodular maximization: offline and secretary algorithms. In *Proc. WINE*, pages 246–257, 2010.

[GV11] Shayan Oveis Gharan and Jan Vondrák. On variants of the matroid secretary problem. In *Proc. ESA*, pages 335–346, 2011.

[IW11] Sungjin Im and Yajun Wang. Secretary problems: laminar matroid and interval scheduling. In *Proc. SODA*, pages 1265–1274, 2011.

[JSZ13] Patrick Jaillet, José Soto, and Rico Zenklusen. Advances on matroid secretary problems: Free order model and laminar case. In *Proc. IPCO*, pages 254–265, 2013.

[Kle05] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proc. SODA*, pages 630–631, 2005.

[KP09] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *Proc. ICALP*, pages 508–520, 2009.

[Mos87] Frederick Mosteller. *Fifty Challenging Problems In Probability With Solutions.* Dover Publications, 1987.

[MTW13] Tengyu Ma, Bo Tang, and Yajun Wang. The Simulated Greedy Algorithm for Several Submodular Matroid Secretary Problems. In *Proc. STACS*, pages 478–489, 2013.

[Oxl92] J.G. Oxley. *Matroid theory.* Oxford Graduate Texts in Mathematics Series. Oxford University Press, Incorporated, 1992.

[Sey80] P. D. Seymour. Decomposition of regular matroids. *J. Combin. Theory Ser. B*, 28(3):305–359, 1980.

[Sey95] P. D. Seymour. Matroid minors. In R. L. Graham, M. Grötschel, and L. Lovász, editors, *Handbook of combinatorics (vol. 1)*, pages 527–550. MIT Press, Cambridge, MA, USA, 1995.

[Sot11] José A. Soto. Matroid secretary problem in the random assignment model. In *Proc. SODA*, pages 1275–1284, 2011.

[Whi35] Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):pp. 509–533, 1935.