Two Algorithms for the Matroid Secretary Problem

Liri Finkelstein

Two Algorithms for the Matroid Secretary Problem

Research Thesis

Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Management Engineering

Liri Finkelstein

Submitted to the Senate of the Technion - Israel Institute of Technology

Tishrei, 5772 Haifa October 2011

The Research Thesis Was Done Under the Supervision of Dr. Ron Lavi
At the Faculty of Industrial Engineering and Management

I thank Prof. Robert Kleinberg from the Department of
Computer Science at Cornell University for helpful talks.

The Generous Financial Help of the Technion is Gratefully Acknowledged.

Contents

List of Figures

Abstract

The matroid secretary problem is a setting in which elements arrive at a random order, and the algorithm must make an irrevocable decision whether or not to select each element. Not all subsets of elements can be selected, only those subsets that are independent sets of a pre-specified matroid. Each element has a weight, that is revealed when the element arrives, and the goal is to select a subset of elements with maximal weight. The offline problem is optimally solved by a greedy algorithm for any matroid and weight function. An open question is whether constant competitiveness is achievable for all matroids. Several attempts to answer this question during the last decade yielded solutions for some important classes of matroids: uniform, partition, transversal, graphic, and laminar.

I studied laminar matroids - a class that was not solved during the time I worked. I defined cells algorithm and showed it is 4-competitive for uniform matroid and 9.5-competitive for laminar matroids of height 2. I used ideas from the cells algorithm for a new algorithm for general matroids - the circuits algorithm, and showed that it has constant competitive ratios for special cases of matroids that were not studied before. An interesting property of the circuits algorithm is that it uses only terms of general matroids, not relying on properties of a certain class of matroids. Finally, I proved that the circuits and the cells algorithms yield exactly the same output for laminar matroid of general height.

Chapter 1

Introduction

1.1 The secretary problem

The *secretary problem* was introduced in the 1960's, sometimes referred to as the *marriage problem*, and it goes as follows: a known number of candidates are about to be interviewed for a secretary position. They arrive in a randomly-ordered sequence. Whenever a candidate is interviewed, the manager learns her true "value", and then he must make an irrevocable decision as to whether or not to accept her for the position. If he accepts her, he might loose more valuable candidates that are about to arrive later. On rejection, he takes the risk that no better candidate will show later on. An algorithm needs to balance the treatment of those two contradicting options, such that in expectation over all arrival orders, the most valuable candidate will be chosen in a probability that is constant (not dependent on the number of secretaries) and highest as possible. Several variations were studied. Here are some examples: assuming a certain distribution on the possible values, looking for the candidate with highest value below a certain limit, adding capacities to the secretaries. The addition of capacities to the elements brings the problem to the well known knapsack problem. In slightly different phrasing of the problem, which is a special case of the matroid secretary problem, the goal is to maximize the expected value of the chosen secretary. Thus, for example, selecting other candidates with values close to the best, might also be useful.

1.2 Generalizations of the secretary problem

The first generalization to the *multiple-choice secretary problem* was suggested by Kleinberg in [4]. The problem is to select the best k secretaries, or more exactly, find an algorithm that in expectation over all arrival orders, achieves a constant fraction of the best solution (that is, the sum of values of the best k secretaries). The next question, raised by Babaioff et. al in [1], was about general subsets systems. That is, every subset of candidates might be selectable. The value of a subset is the sum of values of the candidates in that subset. The algorithm knows in advance which subsets are selectable, but does not know the candidates' values. The goal is to minimize the worst ratio between the result of OPT and ALG's output, going over all possible weight functions. This concept is called ALG's competitiveness, and it will be defined formally in the following chapter. They showed a lower bound

of $\Omega\left(\frac{log(n)}{loglog(n)}\right)$ on the possible competitiveness of algorithms for general subset systems. This led to the idea of restricting the discussion to matroids, which are subset systems with additional properties (will be defined formally later on), or in other words: *the matroid secretary problem*. In a worst-case analysis, even the restriction to matroids is not enough. Several relaxations were suggested, such as enabling a certain measure of cancellations [2]. The most popular relaxation, which will be used here, is an assumption of equal probability for every order of elements arrival. Babaioff et. al (also in [1]) described an algorithm that is $O\left(log\left(k\right)\right)$-competitive where k is the matroid's rank. They conjectured that there is a *constant competitive* solution for any matroid. That is, there exists an algorithm whose expected output on any matroid and any weight function is at least a constant fraction of the value of the optimal legitimate subset. However, this was not proved nor refuted so far.

1.3 Motivation for the matroid secretary problem

The motivation for research of the matroid secretary problem goes far beyond the original secretary problem, spreading to several fields. Here are some examples:

- *Graph theory* - An undirected graph is given in advance. The weights of the edges are revealed one by one (in a randomly-ordered sequence), and the goal is to find a maximal weight spanning tree, subject to the secretary problem constraints. That is, an edge can be selected only upon its arrival and not later on. This problem is the matroid secretary problem on the class of *graphic matroid*. Full definition and examples of this and other classes of matroids appear in the following chapter.

- *Transversal theory* - A bipartite graph is given in advance. The right nodes have no weight. The left nodes have weights, but those weights are revealed online one by one. The goal is to create a maximum-weight matching. When a left node's weight is revealed, it can be selected to the matching, but not later on. The actual matching to right nodes is done after all information has been revealed. This problem is the matroid secretary problem on the class of *transversal matroid*. Full definition and examples appear in the following chapter.

- *Online auctions* - An online auction with a single value domains is a game in which players arrive one after the other. On arrival, a player announces his bid for winning the auction, and then the organizer must make an irrevocable decision as to whether or not the player is one of the winners. Thinking of the players as the ground set of a matroid, and limiting the possible winners sets to form independent sets of a matroid, one gets the obvious extension of the matroid secretary problem to online auctions mechanism design. The main difference between the two is the addition of possible incentives in the last.

1.4 Previous results

Constant competitive solutions for the matroid secretary problem as described above where given for certain classes of matroids:

1. *Uniform* - e-competitive [7]. When the matroid's rank k is large, Kleinberg in [4] gives a better ratio of $1 - \Theta\left(\frac{1}{\sqrt{k}}\right)$. Those two algorithms are fully listed in chapter 3, where I suggest a new algorithm and show how it works on the uniform case.

2. *Transversal* - Pal and Korula showed 8-competitive [5], which is an improvement to a result of $4d$-competitiveness for bipartite graphs with the maximal degree of a left vertex is d by Babaioff et. al [1].

3. *Graphic* - $2e$ in [5].

4. *Laminar* - $\frac{16{,}000}{3}$-competitive in [10]. In 2011, Im and Wang have shown a constant-competitive algorithm for laminar matroids. A significant part of my thesis concerns laminar matroids, as both works were done simultaneously, not knowing of each other. More about this algorithm at the end of chapter 3.

Another result of [1] is a reduction from a matroid M to any *truncation* of M (operations on matroids are explained later in the thesis). If M is c-competitive, then any truncation of M has an algorithm which is $max\,(13c, 400)$-competitive.

1.5 Results in this thesis

My research goal was to deepen the understanding of the matroid secretary problem by exploring laminar matroids - a class of matroids that was not addressed until my work has started. As seen in figure 2.2, laminar matroids can be viewed as a generalization of partition matroids, and as a special case of gammoids. In this thesis I show cells algorithm, which is 4-competitive for uniform matroids and 9.5-competitive for laminar matroids of height 2. Generalization of ideas from the cells algorithm give rise to a new algorithm for general matroids - the circuits algorithm. The circuits algorithm is constant competitive for some special cases of matroids that were not studied before. An interesting property of the circuits algorithm is that it uses only terms of general matroids, not relying on properties of a certain class of matroids. Finally, I prove that the cells and circuits algorithm share the same output for laminar matroids of any height.

1.6 Outline of the rest of the thesis

- Chapter 2 - Preliminary background on the secretary problem and matroids.
- Chapter 3 - The cells algorithm for uniform matroid (4-competitive) and laminar matroid of height 2 (9.5-competitive).
- Chapter 4 - The circuits algorithm for general matroid.
- Chapter 5 - Proof that the cells algorithm and the circuits algorithm yield the same output for laminar matroids.
- Chapter 6 - Conclusion and suggestions for future research.

Chapter 2

Preliminaries

2.1 Secretary problems

2.1.1 Standard solution scheme for the secretary problem

Algorithms for the secretary problem are usually composed of two phases: sample and selection. The idea is to look at a certain fraction of the candidates without choosing anyone, and then select a candidate that complies with certain criteria - usually a candidate whose value passes a threshold which the algorithm computes out of the information known so far or just during the sample. The standard proof shows a lower bound on the percentage of realizations in which the optimal candidate was selected. It computes such a bound according to sufficient conditions for the candidate's selection. The probability of the realization of those conditions gives the competitive ratio. Both concepts are used in this thesis: the algorithms are composed of sample and selection phases, and their analyses follow the idea of summing over the probabilities of selecting each member of the optimal base.

2.1.2 4-competitive algorithm for the secretary problem

Algorithm

Sample $\frac{n}{2}$ candidates without performing a selection, and afterwards select a candidate if her value is higher than all the values of candidates seen so far.

Analysis

Denote the most valuable candidate by z_1 and the second most valuable by z_2. When $z_2 \in Sample$, no other candidate except z_1 can be selected. Therefore, a lower bound on the probability of selecting z_1, which is the optimal offline solution, is: $Pr\left[z_1 \notin Sample, z_2 \in Sample\right] = Pr\left[z_1 \notin Sample\right] \cdot Pr\left[z_2 \in Sample | z_1 \notin Sample\right] \geq \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$.

Enhanced algorithm is e-competitive

When sampling $\frac{n}{e}$ instead of $\frac{n}{2}$ the algorithm is *e-competitive* ($e \approx 2.718$). This ratio is achieved using other analyses [3, 11].

2.2 Matroids

2.2.1 Basic definitions

Definition 2.2.1 *A* subset system $(\mathcal{E}, \mathcal{I})$ *is a set of elements* $\mathcal{E}$ *and a set of subsets of elements* $\mathcal{I} \subseteq P(\mathcal{E})$ *(where* $P(\mathcal{E})$ *is the power set of* $\mathcal{E}$*). A* matroid $M(\mathcal{E}, \mathcal{I})$ *is a subset system with additional properties.*

Definition 2.2.2 *A subset system* $M(\mathcal{E}, \mathcal{I})$ *is a* matroid *if it satisfies the following conditions*

1. $\emptyset \in \mathcal{I}$
2. $\mathcal{I}$ *is closed under inclusion:* $(T \in \mathcal{I}) \wedge (S \subseteq T) \rightarrow (S \in \mathcal{I})$
3. *If* $T, S \in \mathcal{I}$ *and* $|T| < |S|$*, then there is an element* $e \in (S \setminus T)$ *such that* $T \cup \{e\} \in \mathcal{I}$*.*

Matroid notations

- $\mathcal{E}$ is called the *ground set*, and each $e \in \mathcal{E}$ is called an *element*.
- Each $I \in \mathcal{I}$ is called an *independent set*.
- $n = |\mathcal{E}|$ - The number of elements in the matroid.
- $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$ - The elements (not ordered in any way).
- $T \in \mathcal{I}$ is called a *base* of the matroid if T is a maximal independent set with respect to inclusion. The collection of all bases of a matroid is denoted by $\mathcal{B}$. It can be shown that all bases of a matroid have the same cardinality. This cardinality is called the matroid *rank* and is denoted by k. (more on the basics of matroids theory can be found in [8, 9])
- $C \in P(\mathcal{E}) \setminus I$ is called a *circuit* of the matroid if C is a minimal dependent set with respect to inclusion. The circuits of a matroid can be of different cardinality. The collection of all circuits of a matroid is denoted by $\mathcal{C}$.
- Throughout this work, assume all elements have *different* and positive weights. This assumption can be relaxed easily, as suggested in [7].Also assume that the matroids are *normal*: $\forall e \in \mathcal{E}$: $\{e\} \in \mathcal{I} \wedge \exists C \in \mathcal{C}$ *such that* $e \in C$.

2.2.2 The greedy algorithm

For any subset system $(\mathcal{E}, \mathcal{I})$, the problem of finding maximum-weight subset is defined as follows: For a function $W : \mathcal{E} \rightarrow \Re_+$, find a subset $S \in \mathcal{I}$ that maximizes $\sum_{e \in S} W(e)$. For a matroid, there is a simple greedy algorithm that finds the maximum-weight subset:

1. Order the elements by weight: $W(e_1) \geq W(e_2) \geq \ldots \geq W(e_n)$.
2. Initialize $S \leftarrow \emptyset$.

3. For each element $e \in \mathcal{E}$ in descending order of weight, if $S \cup \{e\} \in \mathcal{I}$. then $S \leftarrow S \cup \{e\}$

It turns out that a matroid can actually be defined using this property. If the greedy algorithm finds the maximum-weight subset of a subset system $(\mathcal{E}, \mathcal{I})$ for any weight function, then the subset system is a matroid [6].

2.3 Definition of the matroid secretary problem

Babioff et al. [1] defined the matroid secretary problem as follows: a matroid's structure $M(\mathcal{E}, \mathcal{I})$ is known in advance, but the elements' weight function W is not known. The elements of $\mathcal{E}$ arrive at a randomly-ordered sequence. On arrival, an element reveals its weight, and then the algorithm must make an irrevocable decision as to whether or not to accept it. At the end, the algorithm returns the set of accepted elements, which must be in $\mathcal{I}$, and should be of maximum-weight. As said before, the offline problem is optimally solved by the greedy algorithm for any matroid and weight function. However in the online case it is impossible for an algorithm to guarantee the output of the maximum-weight subset, not knowing W in advance. Assume an algorithm ALG, that works on the online problem of a certain matroid $M(\mathcal{E}, \mathcal{I})$, and outputs $S(\mathcal{E}, \mathcal{I}, W)$ for a specific weight function W (revealed one element at a time). The goal would be to minimize the *competitive ratio*, which is the worst ratio between the result of the best offline algorithm (in the matroid case, this is the OPT), and the result of ALG, going over all possible weight functions. The competitive ratio of ALG (that outputs a set S) is: $max_{possible\ W\ functions}\left(\frac{\sum_{e \in GREEDY(\mathcal{E},\mathcal{I},W)} w(e)}{E_{arrival\ orders\ and\ ALG's\ randomizations} \sum_{e \in S(\mathcal{E},\mathcal{I},W)} w(e)}\right)$. It is as if we choose an algorithm and then an adversary tries to fail it with the worst possible weight function. The adversary cannot change the matroid structure (which subsets are independent), and cannot change the random arrival order (the adversary's output is also an expectation over arrival orders). ALG is a constant competitive algorithm for the matroid secretary problem if it is constant competitive for each and every matroid.

2.3.1 Classes of matroids

Uniform

Definition 2.3.1 *$M(\mathcal{E}, \mathcal{I})$ is a* uniform matroid *with parameters n and k, written sometimes as Uni(n,k), if $|\mathcal{E}| = n$ and $\mathcal{I} = \{T \subseteq P(E) : |T| \leq k\}$.*

The classic secretary problem domain can be interpreted as a uniform matroid with $k = 1$. The ground set is the set of candidates, and the independent sets are all the singletons. The problem of selecting k secretaries instead of one can also be described as selecting the best independent set of a uniform matroid. This problem is also known as "the multiple-choice secretary problem" [4].

Partition

Definition 2.3.2 *A matroid $M(\mathcal{E},\mathcal{I})$ is a* partition matroid *if there is a partition of $\mathcal{E}$ into d sets $E_1, E_2, \ldots, E_d$ with associated integers $k_1, k_2, \ldots, k_d$, and a subset $A \subseteq \mathcal{E}$ is independent if and only if $\forall i \in \{1, \ldots, d\} : |A \cap E_i| \leq k_i$. When $d = 1$ it is a* uniform matroid.

Transversal

Definition 2.3.3 *$M(\mathcal{E},\mathcal{I})$ is called a* transversal matroid *if it is possible to build an undirected bipartite graph $G(L, R, Edges)$ as follows:*

- *There is a one-to-one and onto function $f : \mathcal{E} \to L$.*
- *$U \in \mathcal{I}$ if and only if there is a matching from the vertices $\{l \in L | \exists u \in U$ such that $l = f(u)\}$ to R. A "matching" means that each of the relevant left nodes is attached by an edge to a different right node.*

A transversal matroid might have multiple bipartite graph representations. Moreover, within the same representation, there might be several types of matching for a certain independent set. A bipartite representation of a transversal matroid is shown in example 2.3.10, on the upper right side of figure 2.3.

Graphic

Definition 2.3.4 *$M(\mathcal{E},\mathcal{I})$ is called a* graphic matroid *if it is possible to build an undirected graph $G(V, E)$ as follows:*

- *The edges are the elements. That means there is a one-to-one and onto function $f : \mathcal{E} \to E$.*
- *$U \in \mathcal{I}$ if and only if the edges $\{e \in E | e = f(u \in U)\}$ form a forest in the graph (the set of edges U contains no cycles).*

A graphic representation of a graphic matroid is shown in example 2.3.10, on the lower left side of figure 2.3.

Laminar

Definition 2.3.5 *A family L of sets is said to be* laminar *if for any two sets $A, B \in L$, either $A \subseteq B$ or $B \subseteq A$ or $A \cap B = \emptyset$. This property represents a tree structure.*

Definition 2.3.6 *$M(\mathcal{E},\mathcal{I})$ is a* laminar matroid *if there is a laminar family of subsets of $\mathcal{E}$, and each subset S in the family has an integer value k_S such that a subset $A \subseteq \mathcal{E}$ is independent if and only if $|A \cap S| \leq k_S$ for each S in the family.*

It is easier to look at a Laminar matroid in its tree form, where the elements of $\mathcal{E}$ are the leaves, and inner nodes v contain the $k(v)$ values, and act as constraints. A set of leaves S is independent if and

only if it does not break any constraint of the tree. That is, for any inner node v, not more than $k(v)$ of its descendants are in S. It is immediately seen that the description of elements and independent sets as stated above satisfies the conditions of definition 2.2.2, or that the greedy algorithm finds a maximum-weight independent set of leaves. A matroid $M(\mathcal{E}, \mathcal{I})$ is said to be Laminar if it is representable in a tree as described above. Note that, given a laminar matroid $M(\mathcal{E}, \mathcal{I})$, its tree representation is not unique. A tree representation of a laminar matroid is shown in example 2.3.10, on the upper left side of figure 2.3, and in example 2.3.11, figure 2.4.

Gammoid

Definition 2.3.7 *The matroid $M(\mathcal{E}, \mathcal{I})$ is a* gammoid *if it is possible to build a directed graph G with two distinguished sets of vertices S and T (G usually has more vertices) as follows:*

- *There is a one-to-one and onto function $f : \mathcal{E} \to S$.*
- *$U \in \mathcal{I}$ if and only if there are $|U|$ vertex-disjoint paths from the U vertices $\{s \in S | s = f(u \in U)\}$ into T. (note that the $|U|$ vertex-disjoint paths end at $|U|$ different nodes in T). When U are all the vertices in the graph, the gammoid is a* strict gammoid.

It is immediate to see that any transversal matroid is a gammoid. The bipartite graph representation of the transversal matroid is just the same graph which shows the matroid is a gammoid. Any laminar matroid is a gammoid as well. Using the laminar tree form, multiply each inner node according to its label. For example, see figure 2.1.

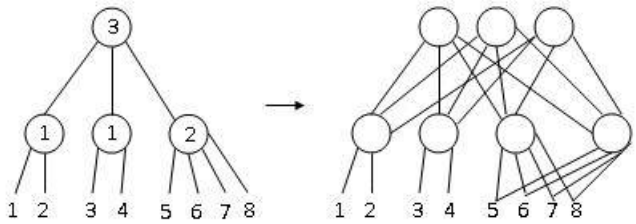

Figure 2.1: Any laminar matroid is also a gammoid

Binary

Definition 2.3.8 *If E is any finite subset of a vector space V, then it is possible to define a matroid M on the ground set E as follows: a set of vectors is in $\mathcal{I}$ if and only if they are linearly independent. This matroid is called a* vector matroid. *A matrix with a set of columns in a field gives rise to a matroid whose ground set is the matrix columns. Such a matroid is called* column matroid. *Those two definitions are almost equivalent and are used interchangeably.*

Definition 2.3.9 binary matroid *is a vector matroid that can be represented over a field of two elements.*

A vector representation of a binary matroid is shown in example 2.3.10, on the lower right side of figure 2.3.

Relationships between various classes of matroids

Figure 2.2 shows the different classes of matroids and the relations between them according to [8] (some of it is in page 214, and some I deduced from differenet theorems). Notes:

- The diagram shows inclusion relationships between the different classes. The relative size of the figures does not indicate relative cardinality.
- Constant-competitive algorithms were found for the colored classes (details of previous results are listed later on).

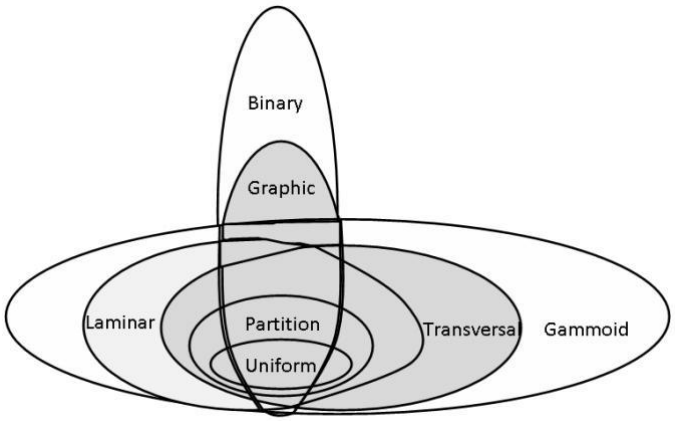

Figure 2.2: Relationships between various classes of matroids

2.3.2 Examples of matroids

Example 2.3.10 *The matroid in figure 2.3 is both laminar, graphic, transversal, binary and a gammoid. It is not uniform, nor partition. The graph which shows the matroid is transversal also proves that it is a gammoid.*

The matroid description:

$\mathcal{E} = \{1, 2, 3, 4, 5\}$

$\mathcal{I} = \{\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 5\}, \{4, 5\}\}$

The bases of this matroid are: $\mathcal{B} = \{s \in \mathcal{I} : |s| = 2\}$.

The circuits are $\mathcal{C} = \{\{1, 2\}, \{3, 4\}, \{1, 3, 5\}, \{1, 4, 5\}, \{2, 3, 5\}, \{2, 4, 5\}\}$.

Example 2.3.11 *The matroid in figure 2.4 is laminar, therefore it is also a gammoid. However, it is not graphic nor transversal (proofs below). Since the matroid is not transversal, it is also not uniform nor partition. This matroid is not binary because it is a gammoid which is not graphic (see figure 2.2).*

The matroid description:

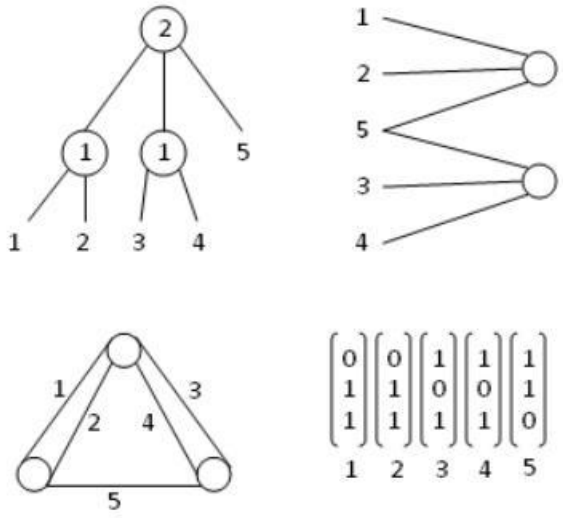

Figure 2.3: A matroid that is laminar, graphic, transversal, binary and gammoid

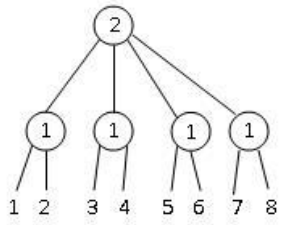

Figure 2.4: Laminar matroid that is not graphic, nor transversal

$\mathcal{E} = \{1, 2, 3, 4, 5, 6, 7, 8\}$

The bases of a matroid entail all its independent sets.

In this case, every set of 2 elments that are not siblings is a base.

The circuits in this case are a union of two groups:

$parent - circuits = \{\{1, 2\}, \{3, 4\}, \{5, 6\}, \{7, 8\}\}$

root-circuits - every set of 3 elments, such that each has a different parent, is a circuit.

Note that not every set of 3 elements form a circuit. A circuit must satisfy the property that removing any one of the elements leaves an independent set.

Claim 2.3.12 *The matroid in figure 2.4 is not graphic.*

Proof: Each pair of siblings must be parallel edges in the graph. Call such a pair of edges a thick edge. So there are 4 thick edges $e_{12}, e_{34}, e_{56}, e_{78}$. A trio of thick edges must form a circuit because the root label in the laminar matroid was 2. Assume a triangle of e_{12}, e_{34}, e_{56}. There is no way to add e_{78} to the graph and maintain the dependency structure of the laminar matroid. If e_{78} adds a new vertex to the graph, then there will be a tree of 3 edges, in contradiction with the fact that every 3 elements are dependent. Otherwise, e_{78} has to be parallel to another thick edge. This option creates dependencies that are not part of the original matroid. □

Claim 2.3.13 *The matroid in figure 2.4 is not transversal.*

Proof: Each pair of siblings must connect to a single right node. Two different pairs cannot share the same right node because that would erase the independent sets of one element from each of the two pairs. The result is a bipartite graph with 4 right nodes, allowing an independent set of size 4, in contradiction to the laminar matroid with root constraint 2. □

This very simple laminar example motivated me to study the class of laminar matroids, as a case which was not solved prior to my work. As can be seen in figure 2.2, laminar and transversal are both gammoids, so I chose to focus on the laminar class in hope to come up with new insights that might be extendable to gammoids (and could have been harder to see in transversal matroids).

2.3.3 Operations on matroids

Given a matroid M, there are several basic operations that can be used in order to create a smaller matroid M', that shares some of the structure and properties of M. Here are two of them:

Truncation

Definition 2.3.14 *Consider a matroid $M(\mathcal{E},\mathcal{I})$. The* truncation of M at j *is a matroid $M_j(\mathcal{E},\mathcal{I}_j)$ where $\mathcal{I}_j = \{T \in \mathcal{I} : |T| \leq j\}$.*

Restriction

Definition 2.3.15 *Consider a matroid $M(\mathcal{E},\mathcal{I})$. The* restriction *of M to $\mathcal{E}' \subseteq \mathcal{E}$ is a matroid $M'(\mathcal{E}',\mathcal{I}')$ where $\mathcal{I}' = \{T \in \mathcal{I} | T \subseteq \mathcal{E}'\}$.*

In a laminar matroid any subtree is a restriction matroid of the whole tree. The following claims will be useful later in this thesis.

Claim 2.3.16 *Consider a matroid $M(\mathcal{E},\mathcal{I})$ and denote the maximum-weight independent set in M by S. Let $M'(\mathcal{E}',\mathcal{I}')$ be a restriction matroid of M where $\mathcal{E}' = \mathcal{E} \setminus \{x\}$. Let S' be the maximum-weight independent set in M'. Then $S \setminus \{x\} \subseteq S'$.*

Proof: Recall that the maximum-weight independent set of any matroid is unique under the assumption of different weights (it is easy to prove using the greedy algorithm). Observe the execution of the greedy algorithm for finding the maximum-weight independent set on M and M' concurrently. Denote the algorithms by Alg and Alg' respectively. The algorithms start from the heaviest element and descend in elements weight. Until x, Alg and Alg' made the same selections. If $x \notin S$, then Alg and Alg' make the same decisions later too. Assume $x \in S$. Alg selects x and Alg' does not, and both algorithms go on, making the same selections until the next difference, call it y. It is impossible that $y \in S$ but $y \notin S'$ because up to now $S' \subseteq S$. If $y \in S$, then any subset of $S \in \mathcal{I}$ is also independent. Moreover, if such a set does not contain x, then it is an independent set of M' too. Specifically, $S' \cup \{y\} \subseteq S$ is an independent set. So y will be selected by Alg'. That is, if $y \in S$, then $y \in S'$. Therefore, $y \in S'$ but $y \notin S$. I argue that both algorithms continue making the same selections until the end. Assume

by contradiction, a third difference, call it z. Denote by T the elements that Alg and Alg' select until z, excluding $\{x, y, z\}$. $T \cup \{x\} \in \mathcal{I}$, $T \cup \{y\} \in \mathcal{I}$, $T \cup \{x\} \cup \{y\} \notin \mathcal{I}$.

If $z \in S$ but $z \notin S'$, then $T \cup \{x\} \cup \{z\} \in \mathcal{I}$ and $T \cup \{y\} \cup \{z\} \notin \mathcal{I}$. This cannot be true because in that case, there is no way to "complete" $T \cup \{y\} \in \mathcal{I}$ into an independent set of size $|T| + 2$ using elements from $T \cup \{x\} \cup \{z\} \in \mathcal{I}$ (recall the definition of a matroid, elaborated before). The reason is that both $T \cup \{y\} \cup \{x\} \notin \mathcal{I}$ and $T \cup \{y\} \cup \{z\} \notin \mathcal{I}$. If $z \in S'$ but $z \notin S$, then $T \cup \{y\} \cup \{z\} \in \mathcal{I}$ and $T \cup \{x\} \cup \{z\} \notin \mathcal{I}$. This is impossible from the same reason: there is no way to "complete" $T \cup \{x\} \in \mathcal{I}$ into an independent set of size $|T| + 2$ using elements from $T \cup \{y\} \cup \{z\} \in \mathcal{I}$.

□

Claim 2.3.17 *Let S be the maximum-weight independent set in $M(\mathcal{E}, \mathcal{I})$, let $\mathcal{E}' \subseteq \mathcal{E}$, and denote $S' = S \cap \mathcal{E}'$. Consider the restriction matroid $M(\mathcal{E}', \mathcal{I}')$ and denote by $S_{restriction}$ the maximum-weight independent set in $M(\mathcal{E}', \mathcal{I}')$. Then $S' \subseteq S_{restriction}$.*

Proof: Immediate from repetitive usage of claim 2.3.16. □

Chapter 3

Cells Algorithm

3.1 Cells algorithm for uniform matroid

3.1.1 Cells algorithm for uniform matroid

The idea is to sample some of the elements and set the heaviest weights seen as thresholds for k cells. Afterwards the algorithm fills up the cells with elements heavier than the thresholds. Note that when $k = 1$ this is almost exactly the classic secretary algorithm. Reasons for the algorithm's details, like the binomial toss of the sample size (instead of the fixed sample size in the classic secretary algorithm), are explained in the analysis.

1. `sample` - Sample the first $x \sim Bin\left(n, \frac{1}{2}\right)$ elements and order them from heavy to light: $e_1, e_2, \ldots, e_x$ without taking anyone.

2. `Cells construction` - Construct k (the matroid's rank) cells: $c_1, c_2, \ldots, c_k$. Use the sample to create the thresholds for the cells: $c^1 = w\left(e_1\right), c^2 = w\left(e_2\right), \ldots, c^x = w\left(e_x\right)$. If $x < k$, pad with zeros the remaining lightest thresholds. A cell whose threshold is zero will be called a *zero-cell*.

3. `Selection` - At start, all cells are *available* and $S \leftarrow \emptyset$. For each incoming element e after the sample, if there exists an available cell c_j such that $c^j < w\left(e\right)$, then select e to the output S. Associate e with the available cell whose thereshold index j is minimal and yet satisfies $w\left(e\right) > c^j$. The cell that e was associated with is considered *unavailable* from now on.

Example 3.1.1 *Consider the following uniform matroid Uni(6,3):* $\mathcal{E} = \{1, 2, 3, 4, 5, 6\}$ *and* $\mathcal{I} = \{T \subseteq P\left(\mathcal{E}\right) : |T| \leq 3\}$. *Assume an arrival order* $4 \to 3 \to 1 \to 2 \to 6 \to 5$ *and a sample size* $x = 2$. *The set of cells and thresholds after the sample appears on the left side of figure 3.1. The final results of the algorithm are displayed on the figure's right.*

3.1.2 Analysis of the cells algorithm for uniform matroid

Theorem 3.1.2 *The cells algorithm is 4-competitive for uniform matroid.*

Claim 3.1.3 *The outcome of the algorithm (S) is an independent set.*

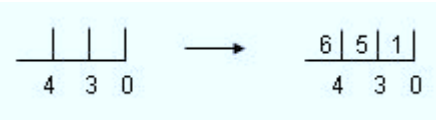

Figure 3.1: Running example of the cells algorithm on a uniform matroid

Proof: One element at the most can be selected and associated with each cell. It means that the algorithm selects up to k elements, thus constructing an independent set. □

Notation for the analysis of the cells algorithm for uniform matroid

- Denote $OPT = \{z_1, z_2, \ldots, z_k\}$ where $w(z_1) > w(z_2) > \ldots > w(z_k)$ (k is the matroid's rank).
- For an element e define $next(e)$ to be the next-heaviest element after e. For the lightest element $e_{lightest}$, define a dummy element with weight 0 as $next(e_{lightest})$.
- For a certain realization, that is, arrival order and toss of x (the number of sampled elements), denote a set $Y = \{e \in OPT | e \notin Sample \wedge next(e) \in Sample\}$, and the elements of Y in decreasing order of weight $\{y_1, y_2, \ldots, y_{|Y|}\}$. Note that y_i is usually not z_i. The idea is to later prove that in each realization in which $z \in OPT$ arrives after the sample, and $next(z)$ is in the sample, ALG gains $w(z)$.

Claim 3.1.4 *For each $1 \leq i \leq |Y|$ there are at least i elements in the algorithm's output that are heavier or equal to $w(y_i)$.*

Corollary 3.1.5 *For each realization* $w(ALG) \geq \sum_{i=1}^{|Y|} w(y_i)$

Proof: The sample is a restriction of the whole matroid, and therefore elements that belong to the optimal base of the matroid, are also part of the optimal base of the sample (see claim 2.3.17). Thus, according to the algorithm, $\{next(y_1), next(y_2) \ldots, next(y_{|Y|})\}$ are used as thresholds. Denote the cells that have those thresholds by $CLS = \{c_{Y1}, c_{Y2}, \ldots, c_{Y|Y|}\}$. Note that c_{Yi} is not necessarily c_i (for example, think what happens if $z_1, z_2 \in Sample$). If all cells in CLS are populated at the end of the algorithm's run, then the claim holds. Otherwise, assume cell c_{Yj} is empty. Then when y_j arrived, it had to catch that cell, and that's a contradiction. y_j could not catch a heavier cell because the threshold of that cell is larger than y_j (remember it is a uniform matroid). y_j could not catch a cell with lighter threshold because according to the algorithm it has to catch the cell with heaviest threshold that is still less than y_j. □

Claim 3.1.6 *The following randomization techniques form the same probability distribution:*

1. *Toss a coin with probability p for head independently for each of the n elements. Assign the value 1 for tosses that came up head, and 0 for tail.*

2. *Sample $x \sim Bin(n,p)$, and then choose uniformly at random a set of x elements (out of all sets with x elements). Those elements are assigned 1, and the rest get value 0.*

3. *Choose a permutation of the n elements uniformly at random and then sample $x \sim Bin(n,p)$. The first x elements get value 1, and the rest get value 0.*

Proof: For $i \in \{1,...,n\}$ denote by $b_i \in \{0,1\}$ the value that was tossed for element e_i: 1 means it is in the sample, and 0 means it comes after the sample. The probability of a certain realization $r = (b_1, b_2, \ldots, b_n)$ that contains d values of 1 under each of the randomization techniques:

1. $Pr[r] = p^d \cdot (1-p)^{n-d}$
2. $Pr[r] = \binom{n}{d} \cdot p^d \cdot (1-p)^{n-d} \cdot \left(\frac{1}{\binom{n}{d}} \right) = p^d \cdot (1-p)^{n-d}$
3. $Pr[r] = \left(\frac{1}{n!}\right) \cdot \binom{n}{d} \cdot p^d \cdot (1-p)^{n-d} \cdot d! \cdot (n-d)! = p^d \cdot (1-p)^{n-d}$. The last multiplications are because the order of elements within the set of ones and within the set of zeros does not matter.

□

Claim 3.1.7 *The cells algorithm for uniform matroid is 4-competitive.*

Proof: Denote by $Y(r)$ the set Y when ALG runs over a realization r. In the cells algorithm for uniform matroid there are two independent randomization processes: arrival order, and sample size. The expectation $E[w(ALG)]$ is on that probability space. A way to calculate this expectation is to create a table as seen in figure 3.2. The rows are the possible realizations, and there is a column for each element. The cell ij will contain either $w(e_j)$ - if in realization i the element e_j was selected by ALG, or 0 if e_j was not selected in realization i.

	z_1	z_2	...	z_k	e_{k+1}	...	e_n	$\Pr[r_i]\sum_j A_{ij}$
r_1	z_1	0		0	0		e_n	...
r_2	0	z_2		0	0		0	...
...	z_1	0		0	e_{k+1}		0	...
...	z_1	z_2		z_k	e_{k+1}		0	...
$r_{n!(n+1)}$	0	0		0	e_{k+1}		e_n	...
								$E=\sum_i \Pr[r_i]\sum_j A_{ij}$

Figure 3.2: Calculating expected outcome of ALG

Sum the numbers in each row, then multiply by the probability of that realization to occur, and finally sum the contribution of all rows to get the expected performance of ALG. In the next step I use corollary 3.1.5, for each realization r: $w(ALG(r)) \geq \sum_{y \in Y(r)} w(y)$. Now cell ij will contain $w(e_j)$ if in

realization i the element $e_j \in Y$, or 0 otherwise. Note that e_j was not necessarily selected by ALG in realization i. The weight gained by ALG in the realization i is splitted differently within the i'th row, such that $e_j \in Y$ is treated as if it was selected. Some of ALG's weight does not "get credit" anymore after this approximation step. See figure 3.3.

	z_1	z_2	...	z_k	e_{k+1}	...	e_n	$\Pr[r_i]\sum_{1\le t\le \lvert Y\rvert} y_t(r_i)$
r_1	**0**	0		0	0	0	0	...
r_2	0	z_2		0	0	0	0	...
...	z_1	0		0	0	0	0	...
...	**0**	z_2		**0**	0	0	0	...
$r_{n!(n+1)}$	0	0		0	0	0	0	...
								$E \ge \sum_i \Pr[r_i]\sum_{1\le t\le \lvert Y\rvert} y_t(r_i)$

Figure 3.3: Approximating the expected outcome of ALG using Y

In the final step I group rows that share the same sample. There are 2^n rows - for all the combinations of possible sample (each element is either in or out of the sample). The order of arrival within the sample, or within the elements after the sample, is not interesting anymore. The grouping is possible due to the structure of the second table, where rows with the same sample look identical (they share the same set Y). Figure 3.4 shows the new table. In this table each row has the same probability to realize, according to claim 3.1.6. Therefore, caculating the desired expectation means just sum over all cells within the table, and divide by 2^n. Instead of going over all realizations, and for each realization sum the contribution of certain elements from OPT to that realization, the point of view is now switched to focus at the elements. That is, for every element in OPT, sum its contributions from certain realizations (the realizations in which it belongs to Y).

	z_1	z_2	...	z_k	e_{k+1}	...	e_n	
r_1	**0**	0		0	0	0	0	
r_2	0	z_2		0	0	0	0	
.	z_1	0		0	0	0	0	
.	**0**	z_2		**0**	0	0	0	
r_{2^n}	0	0		0	0	0	0	
	$\frac{1}{4}2^n z_1$	$\frac{1}{4}2^n z_2$	...	$\frac{1}{4}2^n z_k$	0	0	0	$E \ge \frac{1}{2^n}\sum_i\sum_j A_{ij} = \frac{1}{4}$ OPT

Figure 3.4: Grouping rows and then summing over columns

Summing up the above:

$$E_{r\in realizations}\left[w\left(ALG\left(r\right)\right)\right]\geq E_{r\in realizations}\left[\sum_{y\in Y(r)}w\left(y\right)\right]\geq\sum_{z\in OPT}Pr_{r\in realizations}\left[z\in Y\left(r\right)\right]\cdot w\left(z\right)\geq$$

$$\sum_{z\in OPT}\frac{1}{4}\cdot w\left(z\right)=\frac{1}{4}OPT.$$ □

3.1.3 Other algorithms for uniform matroid from previous works

This section specifies other algorithms that were suggested for the uniform matroid secretary problem (also known as multiple-choice secretary problem). The algorithms are presented without proofs. The interested reader is advised to follow the appropriate references.

Single threshold algorithm that is $\frac{1}{\left(1-5/\sqrt{k}\right)}$-competitive for large rank k

This algorithm of Kleinberg is presented in [4] as a solution to the "multiple-choice" secretary problem.

1. *Special treatment for* $k=1$ - If $k=1$, perform the classical secretary algorithm: sample $\frac{n}{e}$ elements without performing a selection, afterwards select an element if its weight is the heaviest seen so far.
2. *sample* - Else (where $k>1$), Sample the first $x\sim Bin\left(n,\frac{1}{2}\right)$ elements and order them from heavy to light: $e_1,e_2,\ldots,e_x$ without taking anyone.
3. *Choosing a single threshold* - Recursively select up to $l=\lfloor\frac{k}{2}\rfloor$ elements among the first x (the sample).
4. *Selection* - For each incoming element e after the sample, select it if its weight exceeds $w\left(e_l\right)$ and we have not selected k elements yet.

Multiple thresholds algorithm that is $O\left(e\right)$-competitive

This algorithm of Babaioff et. al is presented in [7].

1. *sample* - Sample the first $l=\lfloor\frac{n}{e}\rfloor$ elements and compose a reference set R that consists of the up to k heavieast elements seen during the sample. order them from heavy to light: $e_1,e_2,\ldots,e_R$ without taking anyone.
2. *Selection* - For each incoming element e after the sample, select it if and only if its weight exceeds the lightest element in R. After the selection, remove this lightest element from R.

An alternative selection phase is suggested as follows: For each incoming element e after the sample, select it if and only if its weight exceeds the lightest element in R that also belongs to the sample. After the selection, add e to R, and remove the lightest element from R (whether that element was part of the sample or not).

Babaioff et. al prove that this algorithm, with either one of the selection phases, is $O\left(e\right)$-competitive when the number of elements is large ($n\rightarrow\infty$). Moreover, each element of the optimal base (k heaviest elements) is selected with probability at least $\frac{1}{e}$.

3.2 Cells algorithm for laminar matroid of height 2

3.2.1 Notations for laminar matroid of height 2

- Constraint(b) - The label on the node b (the maximum number of elements that can be selected out of b's descendants). Without loss of generality, assume that each inner node b has at least $Constraint\left(b\right)+1$ children.
- The direct parent of $e \in \mathcal{E}$ is called $parent\left(e\right)$.
- The label on the root is sometimes called the *root-label*, and the label on a parent node is referred to as a *parent-label*.
- Denote $OPT = \{z_1, z_2, \ldots, z_k\}$ like it was defined in the uniform case.
- OPT(b) - The maximum-weight independent set in the laminar tree rooted at b. (It is unique under the assumption of different weights).
- OPT($b|X$) - A maximum-weight independent set in the laminar tree rooted at b, where the elements that are NOT in the set of elements X - weigh 0, while the elements in X have their true weight.

3.2.2 Cells algorithm for laminar matroid of height 2

The cells algorithm for laminar matroid of height 2 is composed of sample and selection phases, as the algorithm for uniform matroid had, but with additional consideration for the subtrees. In the sample part the algorithm updates thresholds for the subtrees as well as for the root. Afterwards, an element is selected by the algorithm only if it has two available cells, each with lower threshold: one of its parent, and one of the root. There is an addition of special treatment for z_1 by executing the secretary algorithm in a certain portion of the algorithm's runs, because the analysis that was used for uniform matroid, when applied to laminar matroid, could not promise the selection of z_1.

1. *`Special treatment for`* z_1 - Toss a fair coin. If it turned up Head, go to step 2. Otherwise, perform the secretary algorithm as was suggested in [3]: sample $\frac{n}{e}$ elements without performing a selection, afterwards select an element if its weight is the heaviest seen so far.
2. *`sample`* - Sample $x \sim Bin\left(n, \frac{1}{2}\right)$ elements $e_1, e_2, \ldots, e_x$ without taking anyone.
3. *`Cells construction`* - For each inner node b, calculate $OPT\left(b|Sample\right) = \left\{x_b^1, x_b^2, \ldots, x_b^{L(b)}\right\}$ where $L\left(b\right) \leq Constraint(b)$. Use it to create the thresholds for b's cells: $c_b^1 = w\left(x_b^1\right), c_b^2 = w\left(x_b^2\right), \ldots, c_b^{L(b)} = w\left(x_b^{L(b)}\right)$. If $L\left(b\right) < Constraint(b)$, pad with zeros the remaining lightest thresholds. A cell whose threshold is zero will be called a *zero-cell*.
4. *`Selection`* - At start, all cells are available and $S \leftarrow \emptyset$. For each incoming element e after the sample, if e has available cells both in its parent and in the root, in other words, there exist available cells i, j such that $w\left(e\right) > c_{root}^i \wedge w\left(e\right) > c_{parent(e)}^j$, then select e to the output S.

Associate e with the heaviest available cells m, n (smallest index of thresholds) that still satisfy $w(e) > c^m_{root} \wedge w(e) > c^n_{parent(e)}$. Each cell that e was associated with is considered unavailable from now on.

Example 3.2.1 *Consider the matroid from figure 2.3. Assume an arrival order* $3 \to 5 \to 1 \to 2 \to 4$ *and sample size* $x = 3$. *The set of cells and thresholds after the sample appears on the left side of figure 3.5. The final results of the algorithm are displayed on the figure's right.*

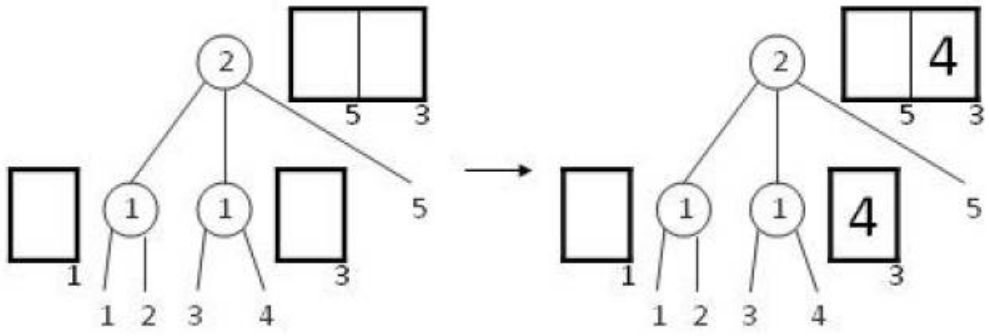

Figure 3.5: Running example of the cells algorithm on a laminar matroid of height 2

3.2.3 Analysis of the cells algorithm for laminar matroid of height 2

Theorem 3.2.2 *The cells algorithm is 9.5-competitive for laminar matroid of height 2.*

Notations for the analysis of the cells algorithm for laminar matroid of height 2

- Each inner node has cells. An inner node in laminar matroid of height 2 can either be the root, or one of the "parents" layer (each of their children are elements). The analysis uses the notations of *root-cell* and *parent-cell*.
- The following two modifications are made to make the analysis more clear.
  - For an element that is a direct child of the root, add a parent node with label "1".
  - Add an element that weigh zero under each of the lowest level of inner nodes (uniform matroids).
- For a certain realization, in which the secretary algorithm in step 1 is not executed, that is, arrival order and toss of x (the number of sampled elements):
  - For $e \in (OPT \setminus \{z_k\})$ denote the next-heaviest element after e in OPT: $next_{root}(e) = argmax_{e' \in OPT: w(e') < w(e)} \{w(e')\}$. Reminder: z_k is the least heaviest element in OPT.
  - For every element e, define its next-heaviest sibling as $next_{parent}(e) = argmax_{e': parent(e') = parent(e) \wedge w(e') < w(e)} \{w(e')\}$.

- $Y = \{e \in OPT | e \notin Sample \wedge next_{parent}(e) \in Sample \wedge next_{root}(e) \in Sample\}$. Denote by $\{y_1, y_2, \ldots, y_{|Y|}\}$ the elements of Y in decreasing order of weight. Note that y_i is usually not z_i. The idea is that in each realization in which $z \in OPT$ arrives after the sample, and certain "$keepers(z)$" are in the sample, ALG gains "almost" $w(z)$.
- For the analysis of a certain realization, define $|Y|$ "Super Cells" with thresholds as follows: $SC_i = w(next_{root}(y_i))$. $1 \leq i \leq |Y|$. Note that $w(y_i) > SC_i > w(y_{i+1})$.

Claim 3.2.3 *The outcome of the algorithm (S) is an independent set.*

Proof: Each inner node b has $Constraint(b)$ cells. One element at the most can be selected and associated with each cell. An element that does not have an available cell both in its parent and in the root cannot be selected. It means that the algorithm selects up to $Constraint(b)$ elements out of the descendants of each inner-node b, thus constructing an independent set. □

Claim 3.2.4 *If $e \in OPT$, then whenever $e' = next_{parent}(e) \in Sample$ and $e \notin Sample$, e' is one of the thresholds of $parent(e)$. In other words, $\exists i : c^i_{parent(e)} = e'$.*

Proof: If $e \in OPT$ then also $e \in OPT(parent(e))$ according to claim 2.3.17 (remember that in a laminar matroid each subtree is a restriction matroid of the whole tree). Therefore e is one of the $Constraint(parent(e))$ heaviest children of $parent(e)$. e', being the next heaviest child of $parent(e)$ after e, is definitely one of its $Constraint(parent(e)) + 1$ heaviest children. Since $e \notin Sample$, e' has to be one of the $Constraint(parent(e))$ heaviest children of $parent(e)$ in the sample. Thus, e' is one of the thresholds of $parent(e)$. □

Claim 3.2.5 *If the algorithm reached step 2, then for each $1 \leq i \leq |Y|$ there are at least i elements in the algorithm's output that are heavier than $w(next_{root}(y_i))$ (that is, SC_i).*

Corollary 3.2.6 *For each realization that the algorithm reached step 2, $w(ALG) \geq \sum_{i=1}^{|Y|} w(next_{root}(y_i))$*

Proof: By induction. As for SC_1, assume by contradiction that no element heavier than $w(next_{root}(y_1))$ was selected. Specifically, $y_1 \notin Sample$ was not selected, though $next_{root}(y_1) \in Sample$ and $next_{parent}(y_1) \in Sample$. The assumption was that no element heavier than $w(next_{root}(y_1))$ was selected, so when y_1 arrived, it had an available root-cell. Therefore, y_1 was not selected because it had no available parent-cell. Denote $e' = next_{parent}(y_1) \in Sample$. According to claim 3.2.4, $w(e')$ is a threshold of $parent(y_1)$. There is no descendant of $parent(y_1)$ with weight between $w(y_1)$ and $w(e')$, because that subtree is a uniform matroid. So if that cell was unavailable, it means an element heavier than y_1, thus, heavier than $next_{root}(y_1)$, was selected. This is a contradiction.

Assume the claim is correct up to SC_{i-1}. If there are more than $i-1$ elements in S that are heavier than $w(next_{root}(y_{i-1}))$, then the claim holds for SC_i too. Assume by contradiction that there are exactly $i-1$ elements from S in the $i-1$ heaviest super cells. If SC_i also contains an element from S, the claim holds for i too. Otherwise, all the cells in SC_i are available.

Consider the parent-cells whose thresholds are $w\left(next_{parent}\left(y_1\right)\right), ..., w\left(next_{parent}\left(y_j\right)\right), ..., w\left(next_{parent}\left(y_i\right)\right)$. Assume one of them, say $next_{parent}\left(y_j\right)$, is available. Then when y_j arrived, it had to be selected. It had an available root-cell in SC_i, and an available parent-cell - the one whose threshold is $w\left(next_{parent}\left(y_j\right)\right)$. It is not possible that y_j is associated with another parent-cell because according to the algorithm, it has to be associated with the parent-cell with heaviest threshold that is still lighter than y_j, and that is the parent-cell whose threshold is $w\left(next_{parent}\left(y_j\right)\right)$. This is a contradiction to the assumption that the cell whose threshold is $w\left(next_{parent}\left(y_j\right)\right)$ is available. Therefore, all of the above parent-cells are occupied. All the subtrees are uniform matroids, so if ALG selected an element heavier than $w\left(next_{parent}\left(y_j\right)\right)$, then this element is also heavier than y_j, and obviously heavier than $next_{root}\left(y_j\right)$. This observation is true for all $1 \leq j \leq i$. Thus, ALG selected at least i elements heavier than $next_{root}\left(y_i\right)$. □

Claim 3.2.7 *The cells algorithm for laminar matroid of height 2 is 16-competitive.*

Proof: Denote by $Y\left(r\right)$ the set Y when ALG runs over a realization r. In the cells algorithm for laminar matroid of height 2, there are three independent randomization processes: decision whether to execute the secretary algorithm, arrival order, and sample size.

$E\left[w\left(ALG\right)\right] \geq \frac{1}{2} \cdot \frac{1}{e} \cdot w\left(z_1\right) + \frac{1}{2} \cdot E_{r \in \text{"Head" realizations}}\left[w\left(ALG\left(r\right)\right)\right]$

According to corollary 3.2.6, in each realization where the algorithm continued to step 2, $w\left(ALG\left(r\right)\right) \geq \sum_{y \in Y(r)} w\left(next_{root}\left(y\right)\right)$. Each element $z \in OPT \backslash \{z_k\}$ belongs to Y in $\frac{1}{8}$ of the realizations where the algorithm continued to step 2. This is because $Pr\left[z \notin Sample \wedge next_{parent}\left(z\right) \in sample \wedge next_{root}\left(z\right) \in sample\right] = Pr\left[z \notin Sample\right] \cdot Pr\left[next_{parent}\left(z\right) \in sample\right] \cdot Pr\left[next_{root}\left(z\right) \in sample\right] = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}$. The independence of probabilities is according to claim 3.1.6.

Hence, $E_{r \in \text{"Head" realizations}}\left[\sum_{y \in Y(r)} w\left(next_{root}\left(y\right)\right)\right] \geq \sum_{z \in OPT \backslash \{z_k\}} Pr_{r \in realizations}\left[z \in Y\left(r\right)\right] \cdot w\left(next_{root}\left(z\right)\right) = \sum_{z \in OPT \backslash \{z_k\}} \frac{1}{8} \cdot w\left(next_{root}\left(z\right)\right) = \frac{1}{8} \sum_{i=2}^{k} w\left(z_i\right)$. Concluding, $E\left[w\left(ALG\right)\right] \geq \frac{1}{2} \cdot \frac{1}{e} \cdot w\left(z_1\right) + \frac{1}{2} \cdot \frac{1}{8} \sum_{i=2}^{k} w\left(z_i\right) \geq \frac{1}{16} w\left(OPT\right)$. One can follow the logic of the proof of claim 3.1.7, using the three tables with minor changes, and get the same result. □

3.2.4 Enhanced cells algorithm for laminar matroid of height 2

Modify constants of the previous algorithm as follows:

- In the first coin toss (determining whether to perform the secretary algorithm or continue) use an unfair coin with probability $\frac{4e}{27+4e}$ to perform the secretary algorithm.
- In the second randomization, sample $x \sim Bin\left(n, \frac{2}{3}\right)$.

Claim 3.2.8 *The enhanced algorithm for laminar matroid of height 2 is* 9.5*-competitive.*

Proof: The same analysis as the proof of claim 3.2.7, using the new constants, shows that the enhanced algorithm is $\frac{27+4e}{4}$-competitive, and $\frac{27+4e}{4} < 9.5$. □

3.2.5 Cells algorithm for laminar matroid of general height

The following algorithm for laminar matroid of general height is almost the same as the one discussed earlier for height 2. Each inner node has its own structure of cells. After the sample, the same kind of thresholds update is done for every inner node. In the selection phase, an element e is selected only if it has an available cell in each of its ancestors. If e was selected, an appropriate cell is updated in each of e's ancestors to be unavailable.

I am unable to determine whether this algorithm is constant competitive. The natural extension of the analysis for laminar matroid of height 2 does not work for height 3. An example for that appears hereafter. However, in chapter 5 I show that the cells algorithm for laminar matroid of general height has the exact same performance as the circuits algorithm, which is discussed in chapter 4.

1. *Special treatment for* z_1 - Toss a fair coin. If it turned up Head, go to step 2. Otherwise, perform the secretary algorithm as was suggested in [3]: sample $\frac{n}{e}$ elements without performing a selection, afterwards select an element if its weight is the heaviest seen so far.

2. *sample* - Sample $x \sim Bin\left(n, \frac{1}{2}\right)$ elements $e_1, e_2, \ldots, e_x$ without selecting anyone. Denote the sample by X.

3. *Cells construction* - For each inner node b calculate $OPT\left(b|Sample\right) = \left\{x_b^1, x_b^2, \ldots, x_b^{L(b)}\right\}$ where $L\left(b\right) \leq Constraint\left(b\right)$ and use those elements' weights to create thresholds for b's cells. If $L\left(b\right) < Constraint(b)$, pad with zeros the remaining lightest thresholds.

4. *Selection* - At start, all cells are available and $S \leftarrow \emptyset$. For each incoming element e after the sample, if for each inner node b that is an ansector of e there is an available cell j such that $w\left(e\right) > x_b^j$, then select e to the output S. and for each of e's ancestors, associate e with the heaviest available cell m (smallest index of threshold) that still satisfies $w\left(e\right) > x_b^m$. Each cell that e was associated with is considered unavailable from now on. (Note that the index m is usually not the same for every ancestor of e).

The competitive analysis for laminar matroid of height 2 fails on height 3

Example 3.2.9 *Consider the laminar matroid of height 3 in figure 3.6, and assume an arrival order* $100 \rightarrow 4 \rightarrow 2 \rightarrow$ *Sample ends* $\rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 300 \rightarrow 400$.

The left figure shows the thresholds values after sample, and the right figure shows the cells population while the algorithm selects elements.

$OPT = \{400, 300, 100, 4\}$

In this realization $Y = \{300\}$.

The algorithm does not select $y_1 = 300$*, nor something heavier than* $next_{root}\left(300\right) = 100$.

Note that when applying the algorithm to the matroid whose root is the one with label 3, with the same arrival order, the analysis of laminar matroid of height 2 does work. In that case $OPT = \{400, 300, 4\}$*,* $Y = \{300\}$*, and the algorithm indeed selects* $5 > next_{root}\left(300\right) = 4$.

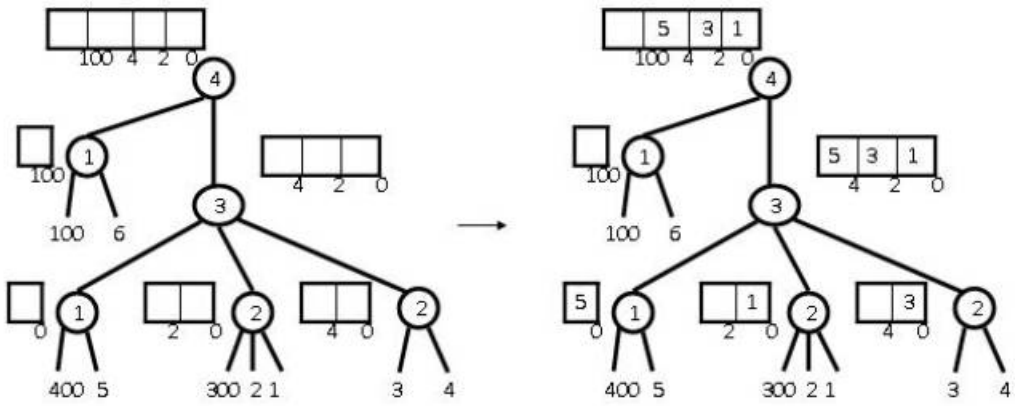

Figure 3.6: The analysis for the cells algorithm fails on laminar matroid of height 3

3.2.6 Another algorithm for laminar matroid that is constant competitive

In [10] Im and Wang showed an algorithm "KickNext" for laminar matroid that is $\frac{16 \cdot 10^3}{3}$-competitive. Their work and mine were done simultaneously, not knowing of each other.

KickNext is almost the same as the cells algorithm. The only difference is that after the sample ends, they toss an unfair coin independently for each of the following elements. With probability $\frac{1}{1000}$ KickNext continues processing this element like the cells algorithm does. With probability $\frac{999}{1000}$ the element is not selected. When an element is not selected, none of the data structures is updated in both algorithms.

The analysis first divides the elements from OPT to "good" and "bad" elements, and shows an injective function in which each bad element is adjusted with a heavier good element. Afterwards they show that if a good element was not in the sample (probability $\frac{1}{2}$), and got the toss $\frac{1}{1000}$, then with probability of at least $\frac{3}{4}$ it has an available cell in each of its ancestors, and therefore it is selected. For the full and challenging proof, the reader is referred to [10].

Chapter 4

Circuits Algorithm

In this section, only the natural constraints of a general matroid are being used. This is as opposed to previous work done on specific classes of matroids like transversal or graphic, or the previous chapter on laminar matroids, where the additional properties of the certain class are used in order to design a constant competitive algorithm. Reminder: a *circuit* is a minimal dependent set. I.e, removing any element from the circuit yields an independent set. It is possible to use circuits in order to encapsulate the dependency structure of a matroid. In fact, the following theorem can be used as an altrnative definition for a matroid.

Theorem 4.0.10 *A matroid is a pair* $(\mathcal{E}, \mathcal{C})$ *with* $\mathcal{C} \subseteq P(\mathcal{E})$ *(called the* circuits *of* $\mathcal{E}$*) satisfying the axioms:*

- $\emptyset \notin \mathcal{C}$.
- *The proper supersets of a circuit are not circuits. That is,* $\forall C_1, C_2 \in \mathcal{C} : C_1 \subseteq C_2 \rightarrow C_1 = C_2$.
- *If* x *is in the intersection of two distinct circuits* S *and* T*, then there is a circuit* $U \subseteq S \cup T \setminus \{x\}$.

The matroid $(\mathcal{E}, \mathcal{C})$ *is normal if no singleton of* $\mathcal{E}$ *is a circuit.*

In this section, an algorithm in terms of circuits (without additional assumptions like properties of matroids from certain classes) is presented. The analysis is generic. However, that analysis is not tight. The next chapter shows a tighter analysis for the special case of laminar matroids.

4.1 Circuits Algorithm

The circuits algorithm follows the cells algorithm by being composed of 3 parts: special treatment for z_1, sample and selection phases. The special treatment for z_1 does not only take care that z_1 is selected in a constant fraction of the algorithm's run, but assures a "semi"-selection of other elements as well. During the sample phase, the algorithm updates thresholds of cells for every circuit. Afterwards, an element is selected if and only if it has an available cell in each of the circuits in which it takes part.

1. *Special treatment for* z_1 - Toss a fair coin. If it turned up Head, go to step 2. Otherwise, perform the secretary algorithm - as is described in the cells algorithm.

2. *Initialization* - For each circuit $C \in \mathcal{C}$ construct $|C| - 1$ cells. Each cell has a threshold, initialized to be zero.

3. *sample* - Sample the first $x \sim Bin(n, p)$ elements without selecting anyone.

4. *Thresholds update* - $\forall e \in Sample$ and $\forall C, e \in C$: If there is a cell in C, whose threshold is zero, update that threshold to be $w(e)$.

5. *Selection* - At start, all cells in all circuits are available. A cell whose threshold remained zero after the thresholds update is called a *zero-cell*. For each element e of the following $n - x$ arrivals:

   - $\forall C, e \in C$: If there is an available cell in C, with threshold's weight lower than $w(e)$, let e temporarily catch the cell of the heaviest such threshold.
   - If e had available cells in all circuits C such that $e \in C$, then take e, and catch the aforementioned cells (which means they are not available anymore). Otherwise, do not take e, and "free" the temporarily caught cells.

Example 4.1.1 *Consider the matroid from figure 2.3. Assume an arrival order:* $2 \to 3 \to 1 \to 4 \to 5$ *and sample size* $x = 2$. *The set of cells and thresholds after the sample appears on the left side of figure 4.1. The final results of the algorithm are displayed on the figure's right.*

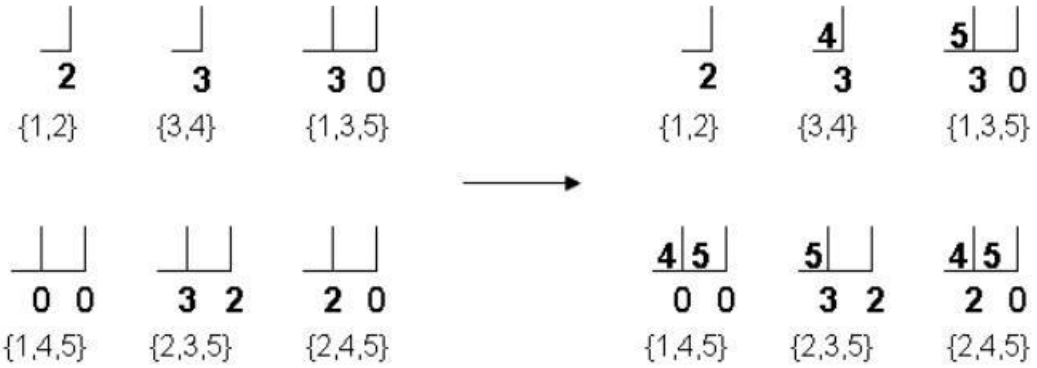

Figure 4.1: Running example of the circuits algorithm

4.2 Analysis of the circuits algorithm

4.2.1 Notations for the analysis of the circuits algorithm

- An element e is *covered* by a set of elements T if $\forall C \in \mathcal{C} : e \in C \to \exists e' \in C \cap T \ (e' \neq e)$. In this case T is called a *covering set* of e.

- An element e is called *d-good* if it is possible to cover e with up to d elements.
- An element e is called *d-bad* if it is not d-good.
- A matroid is *d-coverable* if each one of its bases contains up to d elements that are d-bad (and the rest are d-good).

Example 4.2.1 *The example of graphic matroid in figure 4.2 was given in [1] as a counter example for possible solutions to the matroid secretary problem. The graph contains circuits of size 3 and circuits of size 4. Each element participates in $O(n)$ circuits, but all the elements except one can be covered each with one element. In this case, the matroid is 1-coverable.*

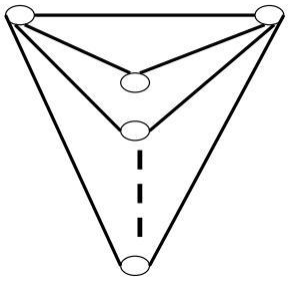

Figure 4.2: Graphic example for the general analysis of the circuits algorithm

Theorem 4.2.2 *The circuits algorithm is 8(d+1)-competitive for d-coverable matroid.*

Claim 4.2.3 *A circuit C does not approve the selection of an element $e \in C$ if and only if all of the following occur:*

- *e was the last to arrive from C.*
- *$argmin_{e' \in C \cap Sample} w(e') > argmax_{e' \in C \cap \neg Sample} w(e')$. That is, each element in $C \cap Sample$ was heavier than each element in $C \cap \neg Sample$.*
- *All the elements in $(C \cap \neg Sample \setminus \{e\})$ were selected (given the previous condition, they were selected on zero-cells). Recall: $e' \in C$ might not get selected due to another circuit C'.*

A summary of those requirements appears in figure 4.3.

Proof: *Sufficiency*: in the predescribed case, e does not have an available cell. All the elements of $C \cap Sample$ created heavy thresholds that none of the following elements passed, thus those cells remained available. e could not catch any of them as well, being too light. The rest of the cells were used by the elements from $C \cap \neg Sample \setminus \{e\}$ which all preceeded e, and were selected.

Necessity:

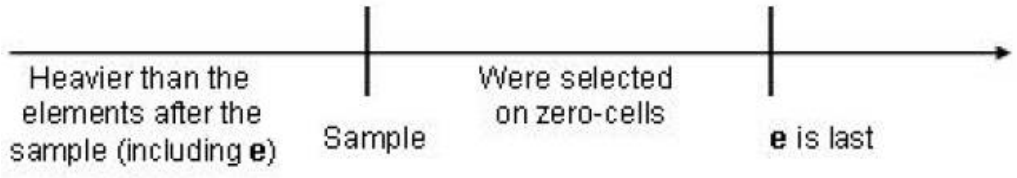

Figure 4.3: Circuit that does not approve an element

- A circuit C has $|C|-1$ cells, so the first $|C|-1$ elements of C are approved by C. This is because any $e \in C \cap \neg Sample$ (when e is not the last to arrive) has an available cell - either a cell with lower threshold that was set during the sample, or a zero-cell. Therefore, C does not prevent the selection of the first $|C|-1$ elements of C.

- If $\exists e \in C \cap \neg Sample$ such that $w(e) > argmin_{e' \in C \cap Sample} w(e')$, then at least one of the elements of $C \cap \neg Sample$ will "use" a cell whose threshold was set during sample. In that case the last element of C to arrive will have an available cell in C. Note that a special case of this is when the last element of C to arrive is itself heavier than $argmin_{e' \in C \cap Sample} w(e')$. In such a case this last element is said to have a "keeper" in the sample.

- If an element from $C \cap \neg Sample \setminus \{e\}$ was not selected, then it leaves an available cell in C for e even if e is the last element to arrive from C.

□

Claim 4.2.4 *If e is $d-$good and the algorithm reached step 2, then $Pr[e \text{ is selected}] \geq (1-p)^{d+1} \frac{1}{d+1}$. (where p is the probability that an element belongs to the sample)*

Proof: Whenever e arrives after the sample and before the (up to) d elements of its covering set, all circuits approve e because there is no circuit in which e arrives last (see claim 4.2.3). $Pr[e \text{ and all its covering set came after the sample AND } e \text{ was the first among them}] = (1-p)^{d+1} \cdot \frac{1}{d+1}$. Independent probabilities stem from claim 3.1.6. The $\frac{1}{d+1}$ stems from the random arrival order. Each of the $d+1$ elements has equal probability of arriving first. □

Claim 4.2.5 *The circuits algorithm is $8(d+1)$-competitive for d-coverable matroids.*

Proof: The proof is similar to the proof of claim 3.2.7. If e is d-good and the algorithm reached step 2, then e is being selected with probability at least $(1-p)^{d+1} \frac{1}{d+1}$ according to claim 4.2.4. Setting $p = \frac{1}{d+1}$ gives $(1-p)^{d+1} = \left(1 - \frac{1}{d+1}\right)^{d+1} \geq \frac{1}{4}$ and then $Pr[d\text{-good } e \text{ is selected}] \geq \frac{1}{4}\frac{1}{d+1}$. There are up to d d-bad elements, each of them is lighter than (or equal to) z_1. The first step of the algorithm assures the selection of z_1 with probability $\frac{1}{2} \cdot \frac{1}{e}$. Alternatively, one can say that each of the d d-bad elements and z_1 contribute their weight with probability $\frac{1}{2} \cdot \frac{1}{e} \cdot \frac{1}{d+1}$ (at least).

$E[w(ALG)] \geq \frac{1}{2}\cdot\frac{1}{e}w(z_1)+\frac{1}{2}\sum_{e\in OPT\wedge\ e\ is\ d\text{-}good} w(e)\cdot Pr[e\ was\ selected\ by\ ALG] \geq \frac{1}{2}\frac{1}{e}\frac{1}{d+1}(d+1)w(z_1)+\frac{1}{2}\frac{1}{4}\frac{1}{d+1}\sum_{e\in OPT\wedge\ e\ is\ d\text{-}good} w(e) \geq \frac{1}{2}\frac{1}{e}\frac{1}{d+1}\sum_{e\in OPT\wedge\ e\ is\ d\text{-}bad} w(e)+\frac{1}{2}\frac{1}{4}\frac{1}{d+1}\sum_{e\in OPT\wedge\ e\ is\ d\text{-}good} w(e) \geq \frac{1}{8(d+1)}OPT$

The third inequality stems from the fact that there are only up to d elements which are d-bad, and each one of them is lighter than z_1 (or has the same weight, if z_1 itself is d-bad). □

4.2.2 Possible selections of d in transversal and graphic matroids

- *Transversal matroid*: one way to cover an element in a transversal matroid, is to keep one of its right neighbours, r, available. That is, all neighbours of r, except e, will be e's covering set. The best neighbour r to "keep" is obviously the one with lowest degree $deg(r)$. $deg(r)$ will be a candidate of e for the value of d. Going over the candidates of all elements leads to d which is the maximal candidate.In a worst case scenario d is just the maximum degree of a right node. Note this is different from the result of [1]. There, d is the maximal degree of a left node.

- *Graphic matroid*: similarly to the transversal analysis, a way to make sure an edge e is covered in all its circuits is to choose one of its ends v and let all the $degree(v)-1$ edges that touch v, except e, be e's covering set. Hence, a possible value for d can be the maximal degree of a node in the graph.

Chapter 5

Identical output of the circuits and the cells algorithms for laminar matroids

5.1 Proof of identical output of the circuits and the cells algorithms for laminar matroids

Both algorithms use the same three randomization processes: decision whether to execute the secretary algorithm, arrival order, and sample size. The proof below shows that for each realization (that is, same randomization results in all three processes) both algorithms select exactly the same elements. If the first coin toss turned such that the secretary algorithm is executed, it obviously does the same as part of the cells algorithm or as part of the circuits algorithm. Otherwise, the algorithms reached step 2. In this case, the proof is by induction on the arrival order of the elements.

5.1.1 Notations and assumptions

- Assume that the laminar tree is normalized in a sense that if an inner node b_{parent} is a parent of an inner node b_{child}, then $Constraint\left(b_{parent}\right) > Constraint\left(b_{child}\right)$. Otherwise the label of b_{child} does not impose a true constraint: just make b_{parent} the direct parent of all nodes and elements that b_{child} was their parent, and remove the node b_{child}.
- Denote a function $f : Circuits \rightarrow Nodes$ as follows: $f\left(C\right)$ is the lowest (closest to the leaves) node b such that all the elements in C are descendants of b. Such a node must exist and is unique because of the tree structure.

Claim 5.1.1 *For each pair of circuit C and node $b = f\left(C\right)$, $|C| - 1 = Constraint\left(b\right)$.*

Proof: By definition of the function f, $b = f\left(C\right)$ if b is the lowest node such that all the elements in C are descendants of b. Every proper subset $C' \subset C$ is independent, by definition of a circuit. Every such C' contains only descendants of b according to the function f. A necessary condition for a set of elements to be independent in a laminar matroid is that it does not violate any node's constraint. Therefore, for any proper subset $C' \subset C$, and in specific one such that $|C'| = |C| - 1$, the constraint of b holds: $|C'| = |C| - 1 \leq Constraint\left(b\right)$.

Assume by contradiction that $|C| - 1 < Constraint(b)$, or in other words $|C| \leq Constraint(b)$. C is a dependent set, so there is a descendant of b, call it b', whose constraint is violated by C. If all the elements in C are descendants of b', it is a contradiction to the definition of $b = f(C)$ as the lowest such node. Otherwise, just a proper subset of C violates the constraint of b'. This contradicts C being a circuit because it contains a dependent set. □

Claim 5.1.2 *For each realization of arrival order and sample size, the circuits and the cells algorithms for laminar matroid select the same elements.*

Proof: By induction on the arrival order. The first x elements, also known as the sample, are not selected in both algorithms. If $x = 0$, then both algorithms trivially select the first element. Assume $x < n$, let e be an arbitrary element that arrived after sample, and assume that the circuits and the cells algorithms made the exact same selections for all the elements that preceded e.

Assume e was not selected by the circuits algorithm. Hence, there exists a circuit C in which e was the last to arrive, all the elements in $C \cap Sample$ were heavier than e, and all the elements in $C \cap \neg Sample \setminus \{e\}$ were selected on zero-cells (by claim 4.2.3). The following will show that the node $b = f(C)$ had no available cell for e, and therefore e was not selected by the cells algorithm. In other words, when e arrived all of b's cells either had thresholds heavier than e, or were already assigned other elements. According to claim 5.1.1 $Constraint(b) = |C| - 1$. At least $|C \cap \neg Sample \setminus \{e\}|$ descendants of b were selected according to the induction assumption. At least other $|C \cap Sample|$ of b's cells had thresholds heavier than the selected elements $C \cap \neg Sample$. Thus, none of the $Constraint(b)$ cells of b was available for e, and e was not selected by the cells algorithm.

On the other way, assume that e was not selected by the cells algorithm. So it had no available cell in one of its ansectors. Denote the lowest such node b. Look at b's cells, ordered by thresholds from heaviest to lightest. Denote by t the lightest threshold of an unpopulated cell in b. $t > w(e)$ because e could not be assigned to any available cell in b. Denote by B^{1C} the elements that preceded e which are descendants of b, and were selected on cells with thresholds lighter than t. Let B^{2C} be the set of elements (from the sample) that set b's thresholds which are heavier or equal to t. Thus, $\left|B^{1C} \cup B^{2C}\right| = Constraint(b)$.

Consider $C = B^{1C} \cup B^{2C} \cup \{e\}$. Obviously, if this is a circuit, it prevented the selection of e in the circuits algorithm according to claim 4.2.3. I argue that C is indeed a circuit. It is obviously a dependent set, because C contains exactly $Constraint(b) + 1$ descendants of b.

Assume by contradiction that C is not a minimal dependent set. Denote by $C' \subset C$ a minimal dependent set, or in other words, a circuit. Also denote $B^1 = B^{1C} \cap C'$ and $B^2 = B^{2C} \cap C'$. Consider $b' = f(C')$. b' is a descendant of b according to the definition of f (because $C' \subset C$). All the elements of B^1 were assigned a cell of b'. Each element of B^2 had to set a threshold in b' according to claim 2.3.17 (a subtree in a laminar matroid is a restriction of the whole matroid). None of the elements of B^1 could be assigned a cell whose threshold comes from B^2 because $min_{e' \in B^2} w(e') \geq t > max_{e' \in B^1} w(e')$. If $e \notin C'$ there is a contradiction since it is not possible that all of C's $|Constraint(b')| + 1$ elements (according to claim 5.1.1) caught each a different cell of b' - either as a threshold, or as a selected element. If $e \in C'$, then the lowest node that had no available cell for e is not b but b' or lower node, and that's a contradiction to the choice of b as the lowest node that prevented the selection of e.

In case that such t does not exist, it means $|Constraint\,(b)|$ descendants of b were selected prior to e's arrival. Denote this set of b's descendants by A. Then $A \cup \{e\}$ is a circuit. It is a dependent set because it contains $|Constraint\,(b)|+1$ descendants of b. Removing any element from $A \cup \{e\}$ leaves an independent set. Remember that according to the definition of b, all lower nodes approved the selection of e.

□

5.2 The former analysis of the circuits algorithm is not tight

Even in the most simple case of a uniform matroid Uni(n,k), each element might participate in a very large number of circuits. Each set of $k+1$ elements is a circuit (and there are no other circuits). Therefore, a covering-set of any element must consist of $n-k$ elements. (If the covering set was of size $n-k-1$, then there had been an uncovered circuit of size $k+1$). Therefore, the analysis in the previous chapter yields a bad competitive ratio ($O\,(n)$) due to a very large covering value d.

However, the cells algorithm for uniform matroids as written in chapter 3 is 4-competitive. Adding the first step of tossing a fair coin would change the analysis there to give 8-competitiveness. The same logic of the proof above can be used to show that this new cells algorithm for uniform matroids and the circuits algorithm select exactly the same elements for any uniform matroid in every realization. Therefore the circuits algorithm is at least 8-competitive for uniform matroids.

Chapter 6

Conclusion and suggestions for future research

6.1 Conclusion

In this thesis I studied the matroid secretary problem. I started from the class of laminar matroids, because of its resemblance to previously solved classes, and its potential to reveal interesting properties for larger classes (such as gammoids). I defined cells algorithm for uniform matroids and showed it is 4-competitive, then generalized the cells algorithm for laminar matroids of height 2, showing it is 9.5-competitive. Next, I used the ideas from the laminar research for matroids in general and presented the circuits algorithm. I showed that it has constant competitive ratios for some special cases of matroids that were not studied before. An interesting property of the circuits algorithm is that it uses only terms of general matroids, not relying on properties of a certain class of matroids. Finally, I proved that the circuits and the cells algorithms yield exactly the same output for laminar matroid of general height.

6.2 Suggestions for future research

6.2.1 Unsolved classes

Up to now, a large portion of the research of the matroid secretary problem was done by solving (that is, finding a constant competitive algorithm) different classes of matroids, with hope that the understanding of special cases will eventually lead to a solution for the general matroid secretary problem. Continuing this path, it seems like the next classes to work on should be the following two: (see figure 2.2)

- *Gammoid* - A generalization of both transversal and laminar matroids, which were already solved.
- *Binary* - A generalization of graphic, that was already solved, and with very tight relation to circuits. For example, according to [8] (page 304), the following statements are equivalent for a matroid M (only partial list here):
  - M is binary.

- If C_1, C_2 are distinct circuits of M, the symmetric difference of C_1 and C_2 contains a circuit C.
- If C_1, C_2 are distinct circuits of M, the symmetric difference of C_1 and C_2 contains a disjoint union of circuits.
- The symmetric difference of any set of circuits is either empty or contains a circuit.
- The symmetric difference of any set of circuits is a disjoint union of circuits.

6.2.2 Operations and other concepts of matroids

Similar to the reslut of Babaioff et al. at [1] on *truncation*, and somewhat similar to claim 2.3.17 on *restriction* in this thesis, it might be useful to understand how different operations on matroids and between matroids effect the competitive ratio of the resulted matroid(s). Here are some of those concepts: *contraction*, *minors*, *direct sum*, *duality*. One motivating example to study those concepts in the context of the matroid secretary algorithms is the following theorem about duality: a matroid is a strict gammoid if and only if its dual is a transversal matroid. [8] (page100).

6.2.3 Circuits

Understand better other concepts related to circuits, that might be useful for new algorithms or better analysis of the circuits algorithm presented in this thesis. For example, prove competitive ratio for matroids that comply with certain properties of concepts like *separators*, *bridges*. (instead of my intuitive use of "coverage"). If going this path, I suggest starting from the book [9], which studies matroids from the prespective of circuits.

6.2.4 Linear programming - packing-covering problem

Below is a simple phrasing of the matroid secretary problem as a linear programming packing-covering problem. It uses circuits as the natural primal constraints:

$j \in \{1, ..., m\}$ for all circuits. l_j is the size of circuit j.

$i \in \{1, ..., n\}$ for all elements.

- *Packing problem*

  $x_i \in \{0, 1\}$ for element i.

  $x_i = 1$ if and only if i is selected. Smoothing - $x_i \geq 0$.

  $Max \sum_{i=1}^{n} w_i x_i$

  $\forall j \in \{1, ..., m\} : \sum_{i \, is \, in \, circuit \, j} x_i \leq l_j - 1$ constraint for each circuit

- *Covering problem*

  $Min \sum_{j=1}^{m} (l_j - 1) \, y_j$

  $y_j \geq 0$ variable for each circuit

  $\forall i \in \{1, ..., n\} : \sum_{j \, is \, a \, circuit \, that \, contains \, i} y_j \geq w_i$ Constraint for each element.

There might be other phrasings of the problem. For example - a variable $x_i^j \in \{0,1\}$ that is 1 if the i'th element to arrive was e_j and it was selected, and 0 otherwise. Moreover, one can consider variables of the kind $x_i^{j|k} \in \{0,1\}$ that is 1 if the i'th element to arrive was e_j and it was the k'th heaviest until now, and it was selected, and 0 otherwise. Buchbinder et. al use similar phrasing of the secretary problem in [12]. They also phrase as linear programming what they call "the J-choice K-best secretary problem". In that problem the algorithm is allowed to select J elements and its objective is to select as many of the K heaviest elements as possible. When $J = K$ this is almost the matroid secretary problem for uniform matroids. The difference is that in the matroid secretary problem the objective is the sum of weights of all the selected elements, and not a sum of hit/miss scores of the top K.

It is interesting to apply techniques from the literature of *packing-covering* linear programming to the online setting of the matroid secretary problem. For example, estimate the algorithm's performance using its dual objective function instead of the current analysis, which uses the primal variables (elements). This method of analysis is called *dual fitting*.

A more ambitious idea is to use the dual problem (where the variables are circuits) to guide the decision making of the algorithm. Buchbinder and Naor in [13] present such a *primal-dual* approach for general online packing-covering linear programming problems. They first solve the fractional version of the problem (in the matroid setting - it means the algorithm can select fraction of an element). For packing problems the algorithm maintains a pair of feasible solutions, updating both primal and dual variables upon the arrival of each primal (packing) variable, and without violating the feasibility of either one of the solutions in hand. The competitive ratio is $O(log(n))$. Afterwards they show how to modify this fractional algorithm into an integral one for some specific problems.

If it's possible to achieve a constant competitive algorithm for the matroid secretary problem using a similar primal-dual algorithm, I guess one should use the relation between constraints (that is, circuits) that characterize matroids and distinguish them from general subset systems. It might be easier to start from the fractional problem, and maybe narrow it down to a certain class of matroids. For example, binary matroids have interesting circuits properties as written at the "unsolved classes" section above.

In the spirit of duality, refer to the algorithm presented by Korula and Pal in [5] for transversal matroids, and more interesting, to their proof that the algorithm is constant-competitive. Recall that in a transversal matroid, the left nodes of a bipartite graph are the elements, and the right nodes are constraints on a feasible outcome. Their analysis uses summation over the "contribution" of each right node to the algorithm's output, unlike my work here, that sums up "contribution" of elements. I believe that some kind of generalization of their idea to the natural constraints of a matroid - which are its circuits, might give a better competitive ratio to the circuits algorithm and maybe to other algorithms as well. Note that such a calculation actually means summing over dual variables.

6.2.5 Markov chains

Try to follow the idea of Markov chains like was done in one of the e-competitiveness proofs ([11]) in order to decide in each step whether to select the element or not. Define states according to knowledge gained during the process of the algorithm, and use backward induction in order to achieve best decisions. I do not know if the same idea is indeed extendable to this problem where there are several

selections and not one. Maybe the computations become impossible to get a good approximation.

6.2.6 Incentives handling

Incentives handling would probably be in the spirit of VCG. That is, each winner should pay a price of another bidder. One needs to think of what information is available for each of the arriving bidders: just the matroid's structure, also the number of bidders who preceded this bidder, their identity, their bids? Assuming an algorithm that encourages truthfulness is found and the bidders are rational. Does it yield maximal revenue for the seller?

Bibliography

[1] M. Babaioff, N. Immorlica, R. Kleinberg. Matroids, secretary problems, and online mechanisms. *Proc. of ACM-SIAM Symposium on Discrete Algorithm*, pages 434-443, January 2007.

[2] F. Constantin, J. Feldman, S. Muthukrishnan, M. Pal. Online add slotting with cancellations. *Workshop on Ad Auctions, 2008. arXiv:0805.1213v1*, 2008.

[3] P. R. Freeman. The secretary problem and its extensions: A review. *International Statistical Review*, 51:189-206, August 1983.

[4] R. Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. *Proc. of ACM-SIAM Symposium on Discrete Algorithm*, pages 630-631, January 2005.

[5] N. Korula, M. Pal. Algorithms for secretary problems on graphs and hypergraphs. *Proc. of International Colloquium on Automata, Language and programming*, 2009.

[6] C. H. Papadimitriou, K. Steiglitz. Combinatorial optimization algorithms and complexity. *Dover publications N.Y.*, pages 214,505, 1992.

[7] M. Babaioff, N. Immorlica, D. Kempe, R. Kleinberg. A knapsack secretary problem with applications. *Proc. International Workshop on Approximation Algorithms for Combinatorial Optimization Problem (APPROX)*, pages 16-28, August 2007.

[8] J. G. Oxley. Matroid Theory. *Oxford university press*, 1992.

[9] W. T. Tutte. Introduction to the Theory of Matroids. *Modern analytic and computational methods in science and mathematics, American elsevier publishing company, inc, N.Y.*, 1971.

[10] Sungin Im, Yajun Wang. Secretary Problems: Laminar Matroid and Interval Scheduling. *Proc. of ACM-SIAM Symposium on Discrete Algorithm*, 2011.

[11] M.J. Beckmann. Dynamic programming and the Secretary Problem. *Computers and Mathematics with Applications, Pergamon Press*, Vol. 19, No. 11, pages 25-28, 1990.

[12] N. Buchbinder, K. Jain, M. Singh. Secretary Problems via Linear Programming. *The 14th Conference on Programming and Combinatorial Optimization*, 2010.

[13] N. Buchbinder, J. Naor. Online Primal-Dual Algorithms for Covering and Packing Pronlems. *Mathematics of Operations Research*, vol. 34, No. 2, pages 270-286, May 2009.