



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

NUEVOS Y MEJORES ALGORITMOS PARA EL PROBLEMA DE LA SECRETARIA
EN MATROIDES

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA
INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

ABNER TURKIELTAUB MELO

PROFESOR GUÍA:
JOSÉ SOTO SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
JOSÉ CORREA HAEUSSLER
MARCOS KIWI KRAUSKOPF

Este trabajo ha sido parcialmente financiado por por Fondecyt de Iniciación 11130266:
APPROXIMATION ALGORITHMS FOR INCREMENTAL SELECTION PROBLEMS

SANTIAGO DE CHILE
2017

RESUMEN DE LA MEMORIA PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN MATEMÁTICAS APLICADAS
Y AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO
POR: ABNER TURKIELTAUB MELO
FECHA: 2017
PROF. GUÍA: SR. JOSÉ SOTO SAN MARTÍN

NUEVOS Y MEJORES ALGORITMOS PARA EL PROBLEMA DE LA SECRETARIA EN MATROIDES

Estudiamos una generalización del problema de la secretaria llamada el problema matroidal de la secretaria propuesta en 2007 por Babaioff et al. [1]. En este problema, los elementos de una matroide se revelan en orden aleatorio. Al observar un elemento, debemos decidir de forma irrevocable si incluirlo o no en nuestra solución. Los elementos aceptados deben formar un conjunto independiente y deseamos que sea cercano al independiente óptimo. En su trabajo, Babaioff et al. [1] conjeturaron la existencia de un algoritmo $O(1)$ -competitivo para este problema en cualquier matroide. Dicha conjetura sigue abierta, y solo se ha podido probar en clases particulares de matroides.

Por un lado esta tesis sirve como lectura introductoria al problema ya que incluye una introducción a lo que son las matroides y al problema de la secretaria. Por otro lado presentamos nuevos resultados sobre este problema. Desarrollamos una nueva técnica para diseñar y analizar algoritmos con la cual obtenemos nuevos algoritmos $O(1)$ -competitivos para cuatro clases de matroides: transversales, gráficas, laminares y un tipo especial de matroides representables que llamaremos k -sparsas. En todos estos casos, nuestros algoritmos funcionan aún bajo hipótesis más restrictivas que las del problema original, y logran una mejor competitividad que la de los mejores algoritmos publicados para esos problemas. Además planteamos y estudiamos algunas variantes del problema.

*A Julieta, mi madre, por todo lo que hace y por existir.
A José, mi profesor guía, por la guía y los consejos.
A Lorna, mi elegida, por elegirme para ser felices juntos.*

Tabla de Contenido

Introducción	1
1. Preliminares	4
1.1. Notación	4
1.2. Matroides	5
1.2.1. Definiciones y propiedades	6
1.2.2. Clases de Matroides	10
2. Problema Clásico y Matroidal de la Secretaria	13
2.1. Problema de la Secretaria	13
2.2. Solución del Problema Clásico	14
2.3. Generalización a Matroides	16
3. Problema de Asignación Online y MSP en Matroides Transversales	20
3.1. Algoritmo óptimo para el problema de la secretaria en matroides transversales	21
4. Problema Gráfico de la Secretaria	26
4.1. Algoritmo para el Problema Gráfico de la Secretaria	27
4.2. Problema Gráfico de la Secretaria en Grafos con Cintura Ancha	30
5. MSP en Matroides Representables con Representación k-sparsa	35
5.1. Algoritmo para el MSP en Matroides Representables k -sparsas	36
6. MSP en Matroides Laminares	41
6.1. Algoritmo 8-competitivo para el LMSP	42
6.2. Algoritmo $3\sqrt{3}$ -competitivo para el LMSP	45
7. Síntesis de los Algoritmos Vistos, Obtención de Bases en el MSP y Algoritmo Cardinal $O(1)$-competitivo para el MSP.	50
7.1. Algoritmo Glotón Mañoso	50
7.2. Obtener Bases en el MSP	51
7.3. Algoritmo Cardinal $O(1)$ -competitivo para el MSP	53
8. Problema de la Secretaria con Despidos Limitados	55
Conclusión	60
Bibliografía	62

Introducción

El problema de la secretaria es un problema clásico de selección online. El problema consiste en elegir al mejor de n candidatos. Los candidatos se presentan en orden aleatorio y solo es posible elegir a un candidato antes de que se presente el próximo. El proceso termina cuando elegimos un candidato o después de que el último de los n candidatos se presente. El nombre del problema viene de que clásicamente se presentaba como un problema de contratación donde los candidatos eran postulantes a un puesto de secretaria. En 1963 Dynkin [5] presentó un algoritmo que elige al mejor candidato con probabilidad $1/e$, resultado que es asintóticamente óptimo.

Aunque ha pasado mucho tiempo desde que surgió el problema original, podría decirse que hace unos 10 años el problema tuvo un resurgimiento cuando Babaioff et al. [1] plantearon una rica generalización del problema: el problema matroidal de la secretaria. En este problema, en vez de tener que elegir un candidato entre n , el objetivo es elegir un conjunto independiente de una matroide dada. Al igual que en el problema original, los elementos de la matroide tienen un peso asignado adversarialmente. Estos pesos se revelan en orden aleatorio. Cada vez que se revela el peso de un elemento debemos decidir de forma irrevocable si incluir o no dicho elemento en nuestra solución. En este contexto, se suele juzgar a un algoritmo según su competitividad. Para ser más precisos decimos que un algoritmo es c -competitivo si la solución que devuelve tiene un valor esperado de al menos $w(\text{OPT})/c$ donde $w(\text{OPT})$ es el valor de la solución óptima (independiente de peso máximo). Bajo este parámetro, mientras más cercano a 1 sea c , mejor el algoritmo. Aunque se han diseñado algoritmos $O(1)$ -competitivos para varias clases de matroides, sigue abierta la conjetura planteada en [1] que estipula que existe un algoritmo $O(1)$ -competitivo para el problema matroidal de la secretaria en cualquier matroide.

A modo de ejemplo y motivación consideremos el siguiente problema de conectividad. Tenemos un grafo no dirigido $G = (V, E)$ que representa una red de comunicación con capacidades no negativas en las aristas $c: E \rightarrow \mathbb{N}$ y un servidor $s \in V$. Los clientes, que toman el lugar de los candidatos en el problema de la secretaria, son representados por vértices del grafo y quieren ser conectados con el servidor s a través de un camino no saturado. El número de clientes, así como los vértices asociados a cada uno son conocidos de antemano. Cada cliente está dispuesto a pagar un precio por ser conectado al servidor, pero no sabemos ni asumimos nada respecto a dicho precio, salvo que es no negativo. Los clientes revelan su precio uno a uno en orden aleatorio. Cada vez que un cliente hace una oferta, el operador de la red debe decidir antes de que llegue un nuevo cliente y de forma irrevocable si sirve o no al actual cliente. El objetivo es elegir un subconjunto de clientes de peso máximo que pueda

ser servido simultáneamente. Se sabe [17] que las restricciones impuestas sobre los conjuntos de clientes que pueden ser servidos simultáneamente corresponde a una clase particular de matroides, las gamoides.

Nuestros Principales Resultados

Además de la competitividad que ya mencionamos, en esta tesis consideramos otras dos nociones de competitividad, que llamamos fuerte y cardinal (Definiciones 2.2 y 2.3 en el Capítulo 2). La competitividad fuerte tiene que ver con la probabilidad de elegir elementos que pertenecen al óptimo e implica la competitividad usual. La competitividad cardinal tiene que ver con el tamaño esperado de la intersección entre el óptimo y lo que devolvemos. En esta tesis presentamos algoritmos fuertemente $O(1)$ -competitivos para 4 clases de matroides obteniendo constantes de competitividad mejores que las publicadas hasta el momento. Además presentamos un algoritmo cardinal $10/3$ -competitivo para matroides generales.

Las 4 clases de matroides que estudiamos en esta tesis y los principales resultados obtenidos en ellas son:

1. Transversales: tomamos el algoritmo e -competitivo presentado por Kesselheim et al. [11] y cambiamos el análisis para demostrar que es fuertemente e -competitivo.
2. Gráficas: presentamos un nuevo algoritmo y probamos que es fuertemente 4-competitivo, superando el algoritmo $2e$ -competitivo de Korula y Pál [13].
3. Representables: para matrices con a lo más k entradas no nulas por columna presentamos un nuevo algoritmo fuertemente $\sqrt[k-1]{k^k}$ -competitivo, superando el ke -competitivo logrado por Soto [19].
4. Laminares: modificamos los dos algoritmos presentados por Jaillet et al. [10]. Obtenemos un algoritmo fuertemente $3\sqrt{3}$ -competitivo, superando el algoritmo $3\sqrt{3}e$ -competitivo presentado en [10] y también el algoritmo fuertemente 9,6-competitivo de Ma et al. [16].

Finalmente, estudiamos una variante del problema matroidal de la secretaria en la que se permite descartar cierta cantidad de elementos elegidos previamente y mostramos que admite algoritmos $O(1)$ -competitivos en matroides generales.

Organización de la Tesis

Los primeros dos capítulos son introductorios. En el Capítulo 1 damos a conocer la notación que usaremos en el resto de la tesis y hacemos una breve introducción a lo que son las matroides. En el Capítulo 2 introducimos más formalmente el problema clásico y matroidal de la secretaria y mostramos el algoritmo óptimo para el caso clásico.

Entre los Capítulos 3 y 6 estudiamos el problema matroidal de la secretaria para distintas clases particulares de matroides. En cada capítulo damos una reseña del problema que estudiamos, indicando los principales aportes hechos hasta ahora en cada uno y presentamos

algoritmos fuertemente $O(1)$ -competitivos para cada clase estudiada. En el Capítulo 3 vemos el caso de las matroides transversales. En el 4 estudiamos el caso de las matroides gráficas. En el Capítulo 5 vemos un caso particular dentro de las matroides representables, que es cuando tenemos una matriz con una cantidad máxima de entradas no nulas por columna asociada a nuestra matroide. En el Capítulo 6 abordamos el problema matroidal de la secretaria en matroides laminares.

El Capítulo 7 tiene tres partes. En la primera sintetizamos los algoritmos presentados en los cuatro capítulos previos en un procedimiento genérico común a todos ellos. Después, como segundo resultado, mostramos como modificar un algoritmo para el problema matroidal de la secretaria de modo que el conjunto devuelto sea una base. Cerramos el capítulo presentando un algoritmo cardinal $O(1)$ -competitivo para el problema de la secretaria en cualquier matroide.

Finalmente, en el Capítulo 8, estudiamos una variante del problema matroidal de la secretaria en donde permitimos eliminar cierta cantidad de elementos previamente seleccionados.

Capítulo 1

Preliminares

Este capítulo tiene dos objetivos principales: primero establecer la notación que ocuparemos a lo largo de la tesis y segundo dar una rápida introducción al lector sobre qué son las matroides, viendo algunos ejemplos y propiedades. En general omitimos las demostraciones de propiedades conocidas o fáciles de probar. Se dan por conocidas nociones básicas de grafos y matrices.

1.1. Notación

- $A \pm a$: Dado un conjunto A y un elemento a , usamos $A + a$ para abreviar $A \cup \{a\}$. Análogamente, usamos $A - a$ para abreviar $A \setminus \{a\}$.
- A : denotamos la salida de los algoritmos por A .
- $|B|$: dado un conjunto B , $|B|$ denota su cardinal.
- \overline{B} : dado un conjunto B , \overline{B} denota su complemento.
- **BFS**: denota el algoritmo Breadth-first search que se utiliza para encontrar bosques generadores en un grafo.
- **c.c.:** componentes conexas.
- **cc:** número de c.c.
- e : número de Euler ($e \approx 2,71828$).
- E : conjunto de elementos de una matroide.
- E_t : conjunto de los primeros t elementos de E en revelarse.
- **e.v.:** espacio vectorial.
- $\langle F \rangle$: dado un conjunto de vectores F , $\langle F \rangle$ denota el e.v. generado por A .
- $[k]$: dado $k \in \mathbb{N}$, definimos el conjunto $[k] := \{1, \dots, k\}$.
- K_n : grafo completo en n vértices.
- $K_{n,m}$: grafo bipartito completo con lado izquierdo de tamaño n y lado derecho de tamaño m .
- **l.i.:** linealmente independiente.

- $\ln(\cdot)$: denota la función logaritmo natural.
- $\log(\cdot)$: denota la función logaritmo base 10.
- **Mejora**: decimos que un elemento x mejora a un conjunto $A \subseteq E$ si $x \in \text{OPT}(A + x)$.
- **MSP**: problema de la secretaria en matroides (por Matroid Secretary Problem).
- $\mathcal{M} \setminus X$: para una matroide $\mathcal{M} = (E, \mathcal{I})$ y un conjunto $X \subseteq E$, $\mathcal{M} \setminus X$ denota a la matroide \mathcal{M} borrando X .
- $\mathcal{M}|_F$: para una matroide $\mathcal{M} = (E, \mathcal{I})$ y un conjunto $F \subseteq E$, ocupamos $\mathcal{M}|_F$ para denotar a la matroide \mathcal{M} restringida a los elementos de F . Es decir, $\mathcal{M}|_F = \mathcal{M} \setminus F^c$.
- n : se reserva n para denotar $|E|$.
- **OPT**: conjunto óptimo (solución) de un problema de optimización.
- **OPT(F)**: conjunto óptimo de un problema de optimización cuando el dominio es restringido al conjunto F .
- **OPT $_t$** : conjunto óptimo de un problema de optimización restringido a lo que se ha visto hasta el tiempo t .
- r : denota el rango de una matroide.
- s : tamaño de muestra.
- **SPG**: sin pérdida de generalidad.
- $t(x)$: En ocasiones usaremos esta notación para referirnos al momento (orden correlativo) en que un elemento x se revela.
- **v.a.**: variable aleatoria.
- x : reservamos la equis minúscula para referirnos a los elementos de una matroide. Agregaremos subíndices a los elementos para denotar el momento en que su peso es revelado. Así x_t denota al t -ésimo elemento en revelarse.

El lector que ya se encuentre familiarizado con las matroides puede saltarse el resto de este capítulo.

1.2. Matroides

Las matroides son estructuras en las que se juntan varias áreas de la matemática que usualmente se estudian por separado: álgebra lineal y abstracta, teoría de grafos, geometría finita y otras. Fueron definidas por Hassler Whitney en su artículo fundacional [20] en 1935, actualmente dicho artículo tiene más de 1000 citas en *Google Scholar*. Las matroides han sido, son y probablemente seguirán siendo un gran tema de estudio e investigación en matemática. Para una introducción más completa a la teoría de matroides, así como para consultar la demostración de las propiedades básicas que presentamos en esta sección, recomendamos al lector interesado consultar [8].

1.2.1. Definiciones y propiedades

Definición 1.1. Par Hereditario: sea E un conjunto e $\mathcal{I} \subseteq \mathcal{P}(E)$ una familia no vacía, decimos que el par (E, \mathcal{I}) es par hereditario si:

$$A \in \mathcal{I} \text{ y } B \subseteq A \Rightarrow B \in \mathcal{I}.$$

Por ejemplo, si tomamos E como el conjunto de aristas de un grafo e \mathcal{I} como el conjunto de todos los emparejamientos contenidos en E , entonces el par (E, \mathcal{I}) es un par hereditario.

Definición 1.2. Matroide: un par hereditario $\mathcal{M} = (E, \mathcal{I})$ es una matroide si cumple la siguiente propiedad:

$$\text{Aumento: } (\forall I, J \in \mathcal{I}) (|I| > |J| \Rightarrow (\exists x \in I \setminus J) : J + x \in \mathcal{I}).$$

De aquí en adelante consideramos $\mathcal{M} = (E, \mathcal{I})$ una matroide. Para todos los efectos de este trabajo, consideramos E finito. Un conjunto $A \subseteq E$ se dirá independiente si $A \in \mathcal{I}$ y dependiente si $A \notin \mathcal{I}$. Para facilitar la comprensión de los conceptos, presentamos ahora dos clases particulares de matroides que servirán de ejemplos: las matroides gráficas y las representables.

Definición 1.3. Matroide Gráfica: dado un grafo $G = (V, E)$ no dirigido definimos la matroide gráfica asociada $\mathcal{M} = (E, \mathcal{I})$ donde los elementos de la matroide corresponden a las aristas del grafo y los conjuntos independientes son los bosques (conjuntos de aristas libres de ciclos).

Para un grafo G , denotamos por $\mathcal{M}[G]$ a la matroide gráfica asociada. En general permitimos que los grafos tengan aristas paralelas.

Definición 1.4. Matroide Representable: una matroide $\mathcal{M} = (E, \mathcal{I})$ se dice representable en un cuerpo \mathbb{K} si existe una matriz Q con coeficientes en \mathbb{K} tal que E se pueda identificar con las columnas de Q de modo que los conjuntos independientes de la matroide coincidan con los conjuntos de columnas l.i. de Q .

Por ejemplo, cualquier matroide gráfica $\mathcal{M}[G]$ es representable en \mathbb{Z}_2 por la matriz de incidencia de G . En general, toda matriz Q con coeficientes en un cuerpo \mathbb{K} define una matroide representable en \mathbb{K} que denotamos $\mathcal{M}[Q]$ y que tiene por elementos las columnas de Q y por conjuntos independientes los conjuntos de columnas l.i.

Definición 1.5. Circuito: decimos que $C \subseteq E$ es un circuito si $C \notin \mathcal{I}$ y $\forall x \in C, C - x \in \mathcal{I}$.

En palabras, los circuitos corresponden a los conjuntos dependientes minimales. Notemos que es posible definir conjunto independiente como conjunto libre de circuitos. En el caso de las matroides gráficas, los circuitos son los ciclos del grafo. En el caso de las matroides representables es más difícil caracterizar los circuitos, pero podemos dar un ejemplo: las columnas de la siguiente matriz con coeficientes en \mathbb{R} forman un circuito:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Definición 1.6. Loop: decimos que $x \in E$ es un loop si $\{x\}$ es un circuito.

En las matroides gráficas los loops también reciben el nombre de loop y corresponde a aristas tales que ambos extremos corresponden al mismo vértice. En las matroides representables una columna de ceros es un loop.

Definición 1.7. Base: decimos que $B \in \mathcal{I}$ es una base si $\forall x \in E \setminus B, B + x \notin \mathcal{I}$.

En palabras, las bases corresponden a los conjuntos independientes maximales. En el caso de las matroides gráficas, las bases son los bosques generadores (árboles generadores si G es conexo). En las matroides representables un conjunto de columnas es una base en el sentido de matroides si y solo si es una base, en el sentido de álgebra lineal, del espacio generado por todas las columnas de la matriz. Las bases cumplen las siguientes propiedades:

1. Intercambio Fuerte: sean B_1 y B_2 bases distintas y sea $x \in B_1 \setminus B_2$, existe $y \in B_2 \setminus B_1$ tal que $B_1 - x + y$ y $B_2 - y + x$ son bases.
2. Cardinal de las bases: si B_1 y B_2 son bases entonces $|B_1| = |B_2|$.

No probaremos la propiedad de intercambio fuerte, pero la usaremos más adelante en la demostración de un lema. La propiedad de cardinal de las bases por otro lado se deduce directamente de la propiedad de aumento, y nos permite definir el siguiente concepto:

Definición 1.8. Rango: sea B una base de \mathcal{M} , definimos el rango de \mathcal{M} como $|B|$ y lo denotaremos por r .

Por ejemplo, si \mathcal{M} es una matroide gráfica asociada a un grafo conexo, entonces $r = |V| - 1$. Para un grafo cualquiera, $r = |V| - cc$. En el caso de matroides representables, el rango coincide con el que se define en álgebra lineal.

Definición 1.9. Borrado: sea $F \subseteq E$, se define la matroide borrando F como

$$\mathcal{M} \setminus F = (E \setminus F, \{I \in \mathcal{I} : I \subseteq E \setminus F\}).$$

En palabras, borrar F genera una nueva matroide donde los elementos son los mismos menos los de F , y los conjuntos independientes son los conjuntos disjuntos con F e independientes en la matroide original. En el caso gráfico, la matroide $\mathcal{M} \setminus F$ es la matroide gráfica asociada al grafo $(V, E \setminus F)$.

Definición 1.10. Rango de un conjunto: sea $A \subseteq E$, definimos el rango de A como el tamaño de un independiente maximal contenido en A y lo denotamos $r(A)$.

Si pensamos en la matroide que queda al borrar \bar{A} , también podemos definir $r(A)$ como el rango de $\mathcal{M} \setminus \bar{A}$ o lo que es lo mismo, el rango de la matroide restringida $\mathcal{M}|_A$. De modo similar, podemos referirnos a las bases de A que corresponden a los independientes maximales contenidos en A , o si se quiere, a las bases de $\mathcal{M} \setminus \bar{A}$.

Una propiedad útil y fácil de verificar es la siguiente:

$$r(A) = |A| \Leftrightarrow A \in \mathcal{I}.$$

Definición 1.11. Span: dado $A \subseteq E$, se define el conjunto generado por A o $\text{Span}(A)$ como el conjunto maximal tal que:

- $A \subseteq \text{Span}(A) \subseteq E$.
- $r(\text{Span}(A)) = r(A)$.

Si tomamos un grafo $G = (V, E)$ con su matroide gráfica $\mathcal{M}[G]$ y $F \subseteq E$, entonces $\text{Span}(F)$ contiene a todas las aristas de F más todas las aristas $x \in E$ que pertenecen a algún ciclo en $(V, F + x)$.

En el caso de matroides representables, si Q es una matriz con conjunto de columnas E y $\mathcal{M}[Q]$ es la matroide asociada, entonces para $F \subseteq E$, tenemos que $\text{Span}(F) = \langle F \rangle \cap E$.

Ahora enunciamos sin demostrar dos propiedades del Span que usaremos luego:

1. Es creciente: si $C \subseteq D$ entonces $\text{Span}(C) \subseteq \text{Span}(D)$.
2. Es idempotente: $\text{Span}(\text{Span}(C)) = \text{Span}(C)$.

Se dice que un conjunto C es generador si $\text{Span}(C) = E$. Notemos que si C es generador, entonces:

$$r(C) = r(\text{Span}(C)) = r(E) = r.$$

Como el rango de C es el tamaño del mayor independiente contenido en C , necesariamente C contiene una base. Las bases son los generadores minimales para inclusión. Ahora definimos dualidad en matroides:

Definición 1.12. Matroide Dual: dada una matroide $\mathcal{M} = (E, \mathcal{I})$ se define la matroide dual como $\mathcal{M}^* = (E, \mathcal{I}^*)$ con:

$$\mathcal{I}^* = \{I \subseteq E : \bar{I} \text{ es generador}\}.$$

Para referirnos a los independientes, a las bases, a los circuitos, a los loops y al rango de la matroide dual hablamos de coindependientes, cobases, cocircuitos, coloops y corango. Notemos que las bases son los complementos de las cobases. En efecto, el complemento de un coindependiente maximal debe ser un generador minimal, es decir, una base.

Existen múltiples formas de definir matroides. Por un lado existen propiedades equivalentes a la propiedad de aumento, pero además es posible definir las sin pasar directamente por la familia de conjuntos independientes. Existen definiciones a partir de las bases, de los circuitos, de la función de rango, del Span y algunas más. Hay una caracterización que nos interesa mencionar aquí y es la que guarda relación con los algoritmos glotones (greedy en inglés).

Consideremos el siguiente problema de optimización: tenemos un par hereditario (E, \mathcal{I}) y una función de peso $w : E \rightarrow \mathbb{R}$, queremos elegir un conjunto maximal en \mathcal{I} y de peso máximo. A continuación se muestra el algoritmo glotón para este problema:

Algoritmo 1 Glotón

Entrada: $(E, \mathcal{I}), w$.**Salida:** Un conjunto A , maximal en \mathcal{I} .Ordenamos $E = \{x_1, \dots, x_n\}$ de modo que $x_1 \geq x_2 \geq \dots \geq x_n$; $A \leftarrow \phi$;**Para** $t = 1 \dots n$ **hacer** **Si** $(A + x_t \in \mathcal{I})$ **entonces** $A \leftarrow A + x_t$;**Devolver** A ;

El algoritmo hace iterativamente lo siguiente: toma el elemento de mayor peso y lo agrega a la solución si el conjunto resultante es factible. No es difícil probar que el algoritmo anterior devuelve un conjunto maximal en \mathcal{I} , pero en general no se puede asegurar nada de su peso. Si por ejemplo E son las aristas de un grafo e \mathcal{I} fueran todos los emparejamientos, entonces es posible que no obtengamos el emparejamiento de peso máximo. La relación entre las matroides y el algoritmo glotón es la siguiente:

Teorema 1.13. *Sea (E, \mathcal{I}) par hereditario. Para toda función de peso $w : E \rightarrow \mathbb{R}$ el algoritmo glotón devuelve un conjunto maximal en \mathcal{I} de peso máximo si y solo si \mathcal{I} satisface la propiedad de aumento.*

En otras palabras, glotón tiene garantía de funcionar si y solo si se aplica en una matroide. Aunque la demostración no es muy difícil, aquí la omitimos. El lector que desee ver una demostración puede encontrarla en [8], Teorema 2.62.

A continuación se presentan un par de lemas que serán utilizados en capítulos posteriores.

Lema 1.14. *En el algoritmo glotón, la condición $A + x_t \in \mathcal{I}$ es equivalente a la condición $x_t \notin \text{Span}(\{x_1, \dots, x_{t-1}\})$.*

DEMOSTRACIÓN. Sea $t \in [n]$, y consideremos el conjunto A justo antes de comenzar la iteración t , se tiene que:

$$\begin{aligned} A + x_t \in \mathcal{I} &\Leftrightarrow r(A + x_t) = |A + x_t| = |A| + 1 = r(A) + 1 \\ &\Leftrightarrow x_t \notin \text{Span}(A). \end{aligned}$$

Para finalizar la demostración basta notar que $\text{Span}(A) = \text{Span}(\{x_1, \dots, x_{t-1}\})$. En efecto, como $A \subseteq \{x_1, \dots, x_{t-1}\}$, tenemos que $\text{Span}(A) \subseteq \text{Span}(\{x_1, \dots, x_{t-1}\})$. Por otro lado, es fácil ver que A es base de $\{x_1, \dots, x_{t-1}\}$, lo que implica que $r(\{x_1, \dots, x_{t-1}\}) = |A| = r(A)$, luego $\{x_1, \dots, x_{t-1}\} \subseteq \text{Span}(A)$. Aplicando Span a ambos lados se obtiene la inclusión faltante y se concluye la demostración. □

Lema 1.15. *Sea $\mathcal{M} = (E, \mathcal{I})$ matroide, $w : E \rightarrow \mathbb{R}$ función de peso inyectiva, entonces hay una única base de peso máximo en \mathcal{M} que llamaremos OPT. Además, OPT solo depende del orden total inducido por w . En otras palabras, si w y w' son dos funciones de peso inyectivas tales que $w(x) > w(y) \Leftrightarrow w'(x) > w'(y)$ entonces la única base de peso máximo para w es también la única base de peso máximo para w' .*

DEMOSTRACIÓN. Para probar esto usamos la propiedad de intercambio fuerte. Supongamos que hay dos bases distintas B_1 y B_2 de peso máximo, sea $x \in B_1 \setminus B_2$ existe $y \in B_2 \setminus B_1$ tal que $B_1 - x + y$ y $B_2 - y + x$ son bases. Como w es inyectiva, $w(x) \neq w(y)$. SPG, $w(x) > w(y)$, pero entonces se tiene que

$$w(B_2 - y + x) = w(B_2) - w(y) + w(x) > w(B_2) \rightarrow \leftarrow$$

Por lo tanto la base B de peso máximo es única y la llamamos OPT. Por el Teorema 1.13 OPT se obtiene con el algoritmo glotón, así que solo depende del orden que w induce en los elementos. \square

Observemos que si w es inyectiva, podemos definir sin ambigüedad para cada conjunto $F \subseteq E$, el conjunto $\text{OPT}(F)$ como la única base de peso máximo de $\mathcal{M}|_F$. De ahora en adelante supondremos SPG que w es inyectiva (si no, podemos romper empates de manera arbitraria).

Lema 1.16. *Sea $A \subseteq E$, se tiene que $\text{OPT} \cap A \subseteq \text{OPT}(A)$.*

DEMOSTRACIÓN. Digamos que $E = \{x_1, \dots, x_n\}$ con $w(x_1) > w(x_2) > \dots > w(x_n)$. Sea $x_k \in \text{OPT} \cap A$, en particular $x_k \in \text{OPT}$, luego x_k es aceptado por el algoritmo glotón. Por el Lema 1.14 tenemos entonces que $x_k \notin \text{Span}(\{x_1, \dots, x_{k-1}\})$. Por monotonía del Span, $x_k \notin \text{Span}(\{x_1, \dots, x_{k-1}\} \cap A)$, lo que significa que al usar glotón en $\mathcal{M}[A]$, x_k quedará en $\text{OPT}(A)$. \square

El lema anterior nos dice que los elementos del óptimo son elementos que mejoran a cualquier conjunto.

1.2.2. Clases de Matroides

En la sección anterior definimos las matroides gráficas y las representables, en esta sección definimos otros tipos de matroides de interés para esta tesis.

Definición 1.17. Matroide k -Uniforme: sea $k \in \mathbb{N}$, una matroide $\mathcal{M} = (E, \mathcal{I})$ se dice k -uniforme si $\mathcal{I} = \{I \subseteq E : |I| \leq k\}$, es decir, los conjuntos independientes son aquellos de cardinal a lo más k .

Definición 1.18. Matroide de Partición Unitaria: una matroide $\mathcal{M} = (E, \mathcal{I})$ se dice de partición unitaria si existe una partición \mathcal{P} de E tal que:

$$\mathcal{I} = \{I \subseteq E : |I \cap P| \leq 1, \forall P \in \mathcal{P}\}.$$

En palabras, los conjuntos independientes son aquellos que contienen a lo más un elemento en cada pedazo de la partición.

Definición 1.19. Matroide Transversal: al igual que con las matroides gráficas, asociamos las matroides transversales a un grafo. Sea $G = (E \cup F, H)$ un grafo bipartito, definimos la matroide transversal asociada como $\mathcal{M} = (E, \mathcal{I})$, con

$$\mathcal{I} = \{I \subseteq E : \text{ existe un emparejamiento que cubre } I\}.$$

En la Figura 1.1 observemos que las matroides de partición, y las k -uniformes son casos particulares de matroides transversales.

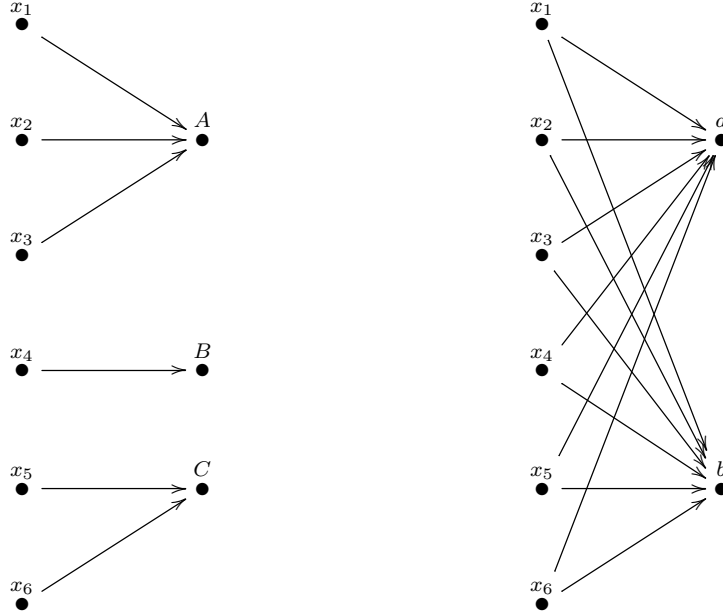


Figura 1.1: En el grafo de la izquierda vemos el grafo cuya matroide transversal coincide con la matroide de partición asociada a la partición $\{\{x_1, x_2, x_3\}, \{x_4\}, \{x_5, x_6\}\}$ mientras que el grafo de la derecha representa una matroide 2-uniforme con 6 elementos.

En general, una matroide de partición asociada a una partición \mathcal{P} puede representarse por una matroide transversal con grafo $(E \cup F, H)$ donde $F = \mathcal{P}$ y $(x, P) \in H$ si y solo si $x \in P$. Una matroide k -uniforme con n elementos puede representarse por la matroide transversal asociada al grafo $K_{n,k}$.

Antes de presentar la próxima clase de matroides necesitamos una definición adicional, la de familias laminares. En la Figura 1.2 se puede apreciar en un diagrama de Venn cómo se ve una familia laminar.

Definición 1.20. Familia Laminar: sea E un conjunto, y $\mathcal{L} \subseteq \mathcal{P}(E)$ una familia de subconjuntos de E . Decimos que \mathcal{L} es una familia laminar si:

$$\forall A, B \in \mathcal{L}, A \subseteq B \vee B \subseteq A \vee A \cap B = \phi.$$

En palabras, al tomar dos conjuntos de la familia, o bien uno está contenido en el otro, o bien son disjuntos.

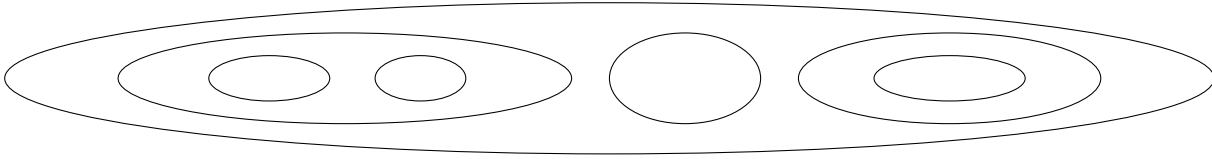


Figura 1.2: Diagrama de Venn de una familia laminar.

Definición 1.21. Matroide Laminar: una matroide $\mathcal{M} = (E, \mathcal{I})$ se dice laminar si existe una familia laminar $\mathcal{L} \subseteq \mathcal{P}(E)$ y números $\{b_L\}_{L \in \mathcal{L}}$ tales que:

$$\mathcal{I} = \{I \subseteq E : |I \cap L| \leq b_L, \forall L \in \mathcal{L}\}.$$

Observemos que tomando $\mathcal{L} = \{E\}$ y $b_E = k$, obtenemos una matroide k -uniforme. Por otro lado, tomando \mathcal{L} una partición de E y $b_L = 1, \forall L \in \mathcal{L}$, obtenemos una matroide de partición unitaria.

Terminamos este capítulo definiendo una clase particular de matroides representables.

Definición 1.22. Matroide Regular: una matroide se dice regular si es representable en cualquier cuerpo.

Capítulo 2

Problema Clásico y Matroidal de la Secretaria

En este capítulo damos a conocer el problema de la secretaria, tanto la versión clásica como la generalización a matroides que nos interesa estudiar. En el caso de la versión clásica que fue resuelta en 1961, exponemos la solución y hacemos un análisis de ésta que sigue la misma línea de análisis que usaremos en capítulos siguientes. Nos interesa entonces usar el problema original para ejemplificar las técnicas que usaremos más adelante en casos más complicados. Finalmente, cerramos el capítulo con algunas observaciones generales sobre los algoritmos que veremos en capítulos posteriores.

2.1. Problema de la Secretaria

Los orígenes del problema no son claros y es posible que haya sido planteado en forma independiente por distintas personas. La primera solución publicada es del año 1961 y pertenece a Lindley [15]. El problema es el siguiente: hay n valores que se irán revelando uno a uno en orden aleatorio, cada vez que se revela un valor podemos rechazarlo y ver el siguiente o tomarlo y terminar el proceso. El objetivo es elegir el máximo de los n valores. El nombre del problema tiene su origen en la siguiente interpretación: hay n candidatas para un puesto vacante de secretaria. Usted las entrevistará haciéndolas pasar de una en una en orden aleatorio. Al final de cada entrevista usted será capaz de rankear las candidatas ya vistas y deberá decidir si contrata a la candidata que acaba de entrevistar o la deja ir para entrevistar a la siguiente. Las candidatas dejan de estar disponibles al salir de la entrevista, y no es posible hacer despidos, así que el proceso acaba cuando se elige una candidata. La idea es buscar una estrategia que maximice la probabilidad de elegir a la mejor candidata. Se hace el supuesto de que todos los valores de los elementos (o candidatas) son distintos. De aquí en adelante nos referiremos a este problema como el problema clásico o problema de la secretaria.

2.2. Solución del Problema Clásico

Consideremos la siguiente estrategia para el problema clásico, tomamos como muestra la primera mitad de los valores sin elegir ninguno y después elegimos el primer elemento que supere a todos los anteriores. Notemos que si el mayor valor queda fuera de la muestra y el segundo mayor valor queda dentro, entonces la estrategia anterior elige al máximo. Por lo tanto la probabilidad de elegir al máximo es al menos $\frac{n/2}{n} \frac{n/2}{n-1} > 1/4$ (supongamos de momento que n es par, si no, se puede tomar cajón superior de $n/2$ o forzar a que n sea par, pero eso lo dejamos para después).

La estrategia anterior resulta ser muy similar a la estrategia óptima, de hecho solo difieren en el tamaño de la muestra. Copiemos la estrategia anterior dejando el tamaño de la muestra como s , variable a optimizar. Tenemos el siguiente algoritmo:

Algoritmo 2

Se ven los primeros s elementos sin elegir ninguno;

$b \leftarrow \max_{i \in [s]} w(x_i)$;

Para $t = s + 1 \dots n$ **hacer**

Si $w(x_t) > b$ **entonces**

 Se elige x_t y se termina;

Queremos calcular la probabilidad de que el algoritmo elija al máximo, llamemos x al elemento de mayor peso y E al conjunto de candidatos. Si x llega en la muestra la probabilidad de elegirlo es 0. Supongamos entonces que x es el t -ésimo elemento en llegar, con $t > s$. Para que x sea elegido necesitamos que ningún elemento anterior a él y posterior a la muestra sea elegido. Ahora hay que aprovechar el hecho de que los elementos llegan en orden aleatorio. Que x llegue en tiempo t tiene probabilidad $1/n$ y condicional a eso podemos simular la elección de los primeros $t - 1$ elementos del siguiente modo: primero se elige E_{t-1} de manera equiprobable entre todos los subconjuntos de $E - x$ de tamaño $t - 1$. Luego los elementos de E_{t-1} se ordenan eligiendo iterativamente el último elemento de los que queden. Es decir, primero elegimos de manera equiprobable un elemento de E_{t-1} y lo designamos para ser x_{t-1} . Luego elegimos de manera equiprobable un elemento de $E_{t-1} - x_{t-1}$ y lo designamos para ser x_{t-2} , y así sucesivamente. La gracia es que sin importar qué elementos formen E_{t-1} , de las $t - 1$ opciones que quedarán disponibles para elegir x_{t-1} , la única que impide la elección de x es el máximo de E_{t-1} . Después, sin importar E_{t-1} y x_{t-1} , lo único relevante en la elección de x_{t-2} es que no sea el máximo de E_{t-2} . En general, si tomamos $i > s$, para elegir x_i tendremos i opciones de las cuales $i - 1$ estarán bien al no ser el máximo de E_i . Si $i \leq s$ todas las opciones estarán bien ya que el orden en que lleguen los elementos de la muestra no afecta el resultado del algoritmo. En conclusión, para cualquier E_{t-1} que fijemos, de las $(t - 1)!$ formas de ordenar sus elementos, hay $s! \prod_{i>s}^{t-1} (i - 1)$ con las cuales x es aceptado por el algoritmo. De este modo podemos calcular la probabilidad de que x sea elegido dado que llega en tiempo t :

$$\mathbb{P}(x \text{ sea elegido} | x = x_t) = \frac{s! \prod_{i>s}^{t-1} (i - 1)}{(t - 1)!} = \prod_{i=s+1}^{t-1} \frac{i - 1}{i} = \frac{s}{t - 1}.$$

De hecho, en este caso pudimos llegar al mismo resultado mucho más rápido, basta observar que para que x_t sea elegido, el máximo en E_{t-1} debe caer en la muestra, lo cual tiene

probabilidad $s/(t-1)$. Hicimos el análisis anterior porque más adelante usaremos la misma técnica variando solo la cantidad de elementos prohibidos en cada etapa.

Usando probabilidades totales tenemos que:

$$\begin{aligned}
 \mathbb{P}(x \text{ sea elegido}) &= \sum_{t=1}^n \mathbb{P}(x \text{ sea elegido} | x = x_t) \frac{1}{n} \\
 &= \sum_{t=s+1}^n \frac{s}{n(t-1)} \\
 &= \frac{s}{n} \sum_{t=s}^{n-1} \frac{1}{t} \\
 &= \frac{s}{n} \int_s^n \frac{1}{\lfloor t \rfloor} dt \\
 &> \frac{s}{n} \int_s^n \frac{1}{t} dt \\
 &= \frac{s}{n} \ln \left(\frac{n}{s} \right).
 \end{aligned}$$

Ahora definimos $p = s/n$ y maximizamos $f(p) = p \ln(1/p) = -p \ln(p)$ con $p \in (0, 1)$. Al derivar f con respecto a p se obtiene $-(1 + \ln(p))$, de modo que la segunda derivada, $-1/p$, es negativa. Por lo tanto al derivar e igualar a 0 estamos maximizando f . Despejando se obtiene que el p óptimo es $1/e$ y $f(1/e) = 1/e$.

Como s debe ser entero, no es posible fijarlo como n/e , pero si elegimos por ejemplo $s = \lfloor n/e \rfloor$ obtenemos un resultado que converge a $1/e$. Para obtener un resultado más limpio podemos hacer una pequeña modificación al algoritmo:

Algoritmo 3

Se simulan n v.a.'s independientes con distribución uniforme $(0, 1)$ y se ordenan de forma creciente obteniendo una secuencia U_1, \dots, U_n ;

$s \leftarrow \text{máx}\{i \in [n] : U_i < p\}$;

Se ven los primeros s elementos sin elegir ninguno;

$b \leftarrow \text{máx}_{i \in [s]} w(x_i)$;

Para $t = s + 1 \dots n$ **hacer**

Si $w(x_t) > b$ **entonces**

 Se elige x_t y se termina;

Como puede apreciarse, la única diferencia está en la forma de muestrear. Lo que estamos haciendo es asignarle a cada elemento un tiempo de llegada en $(0, 1)$ y tomar de muestra a los elementos que lleguen en $(0, p)$. Los n tiempos de llegada se obtienen simulando n v.a.'s uniformes independientes que se asignan de menor a mayor a los elementos de E conforme se van revelando. SPG podemos asumir que los tiempos de llegada son todos distintos ya que el evento complementario tiene probabilidad nula. En lo que queda de esta sección, cuando

hablemos del tiempo de llegada de un elemento nos estaremos refiriendo al U_i que le toca, no al puesto en que llega relativo a los demás elementos.

Sea T el tiempo en que llega x . Podemos ver que $T \sim \text{Uniforme}(0, 1)$ ya que toma al azar el valor de alguna de las variables uniformes. Sigue que:

$$\mathbb{P}(x \text{ sea elegido}) = \int_p^1 \mathbb{P}(x \text{ sea elegido} | T = t) dt.$$

Donde la integral parte de p debido a que si $t < p$ la probabilidad de elegir a x es nula. Consideremos $E(T)$ como el conjunto de elementos que llegan antes de x . Existen dos posibilidades: que $E(T)$ sea vacío, en cuyo caso x es elegido o que $E(T)$ no sea vacío, en cuyo caso tiene un máximo que denotamos z . En el segundo caso, x es elegido siempre y cuando z caiga en la muestra, lo cual tiene probabilidad p/t . Por lo tanto:

$$\mathbb{P}(x \text{ sea elegido}) \geq \int_p^1 \frac{p}{t} dt = -p \ln(p).$$

Que como ya sabemos vale $1/e$ para $p = 1/e$.

2.3. Generalización a Matroides

El problema que nos interesa abordar en esta tesis es el problema matroidal de la secretaria planteado por Babaioff et al. en 2007 [1] (MSP en adelante). En el MSP hay una matroide $\mathcal{M} = (E, \mathcal{I})$ conocida de antemano y un adversario que escoge una función de peso $w : E \rightarrow \mathbb{R}_+$. Los pesos son en un comienzo desconocidos y se revelan en orden aleatorio. Cada vez que se revela el peso $w(x)$ de un elemento x debemos decidir si incluir o no a x en nuestra solución, sin posibilidad de arrepentirnos luego. Al final del proceso debemos retornar un conjunto independiente y buscamos maximizar el peso esperado de éste. El problema clásico corresponde al caso en que \mathcal{M} es una matroide 1-uniforme.

Tanto el problema clásico como el MSP caen dentro del área de los algoritmos online. Entendemos por algoritmo online aquel que recibe su entrada (input) o parte de ella a lo largo de su ejecución. Cada vez que se revela algún dato nuevo, el algoritmo debe tomar una decisión irrevocable usando solo la información revelada hasta el momento. Por contraste llamamos algoritmos offline a los algoritmos usuales que cuentan con toda la información de la entrada desde el inicio de su ejecución. Para evaluar que tan bueno o malo es un algoritmo online para el MSP, en la literatura se ocupa la siguiente noción de competitividad:

Definición 2.1. *Decimos que un algoritmo online para el MSP es c -competitivo si entrega un conjunto independiente A tal que:*

$$w(\text{OPT}) \leq c \mathbb{E}(w(A)).$$

En la definición anterior c puede ser una constante o una función (por ejemplo del rango). También es posible decir, por ejemplo, que un algoritmo es $O(\log(r))$ -competitivo si existe una función $O(\log(r))$ que pueda tomar el lugar de c en la definición. El Algoritmo 2 que usamos para el problema clásico, por ejemplo, es e -competitivo, puesto que:

$$\mathbb{E}(w(A)) \geq w(x)\mathbb{P}(x \in A) = \frac{w(x)}{e} = \frac{w(\text{OPT})}{e}.$$

Mientras la versión offline del MSP (buscar un independiente de peso máximo) se soluciona fácilmente usando el algoritmo glotón, la versión online aún guarda preguntas sin responder. Probablemente la más importante de estas preguntas sea si es cierta o no la conjetura formulada por Babaioff et al. [1] hace ya 10 años. La conjetura dice que para cualquier matroide existe un algoritmo $O(1)$ -competitivo para el MSP asociado. Aunque la conjetura sigue abierta, se han hecho bastantes avances, los cuales dividimos en tres categorías destacando los siguientes trabajos:

1. Se han diseñado algoritmos $O(1)$ -competitivos para varias clases de matroides, por ejemplo:
 - k -uniformes: [12].
 - Transversales: [3], [13], [11].
 - Gráficas: [1], [13], [19].
 - Laminares: [9], [10].
2. Se han diseñado algoritmos para el caso general con competitividad no constante, destacamos:
 - $O(\log(r))$ -competitivo: [1], [19].
 - $O(\sqrt{\log(r)})$ -competitivo: [2].
 - $O(\log(\log(r)))$ -competitivo: [14], [6].
3. Se han diseñado algoritmos $O(1)$ -competitivos en modelos ligeramente distintos, por ejemplo:
 - Random-assignment: en este modelo el adversario no elige la función de peso, solo elige un conjunto W de n valores y w es una biyección elegida de manera equiprobable entre todas las biyecciones que hay entre E y W . Este modelo es planteado por Soto en [19] junto con un algoritmo $O(2e^2/(e-1))$ -competitivo.
 - Random-assignment adversarial-order: aquí el adversario elige el orden en que los elementos serán presentados al algoritmo y la función de peso se elige del mismo modo que en el modelo anterior. En [7] se propone este modelo junto con un algoritmo $O(1)$ -competitivo.
 - Free-order: este modelo es planteado en [10], aquí el adversario elige la función de peso, pero en vez de que los elementos se revelen en orden aleatorio, el algoritmo elige quien será el t -ésimo elemento en revelarse, basado en la estructura de la matroide completa y en los elementos ya vistos. En el mismo artículo se presenta un algoritmo 4-competitivo para este modelo.

En este trabajo presentamos algoritmos nuevos para matroides gráficas, laminares y casos particulares de matroides representables donde obtenemos constantes de competitividad mejores que las logradas hasta ahora. También veremos un modelo donde aumentando la

libertad del algoritmo se puede obtener competitividad constante.

Consideramos en este trabajo tres tipos de competitividad. La que ya definimos es la que se ocupa normalmente en la literatura. A la que le damos más énfasis es a la competitividad fuerte que definimos a continuación:

Definición 2.2. *Decimos que un algoritmo online para el MSP es fuertemente c -competitivo si:*

$$\forall x \in \text{OPT}, \mathbb{P}(x \text{ sea aceptado}) \geq \frac{1}{c}.$$

Notemos que si un algoritmo es fuertemente c -competitivo, entonces:

$$\mathbb{E}(w(A)) = \sum_{x \in E} w(x)\mathbb{P}(x \in A) \geq \sum_{x \in \text{OPT}} w(x)\mathbb{P}(x \in A) \geq \sum_{x \in \text{OPT}} \frac{w(x)}{c} = \frac{w(\text{OPT})}{c}.$$

Es decir, la competitividad fuerte implica la competitividad usual. Además, si un algoritmo es fuertemente c -competitivo entonces:

$$\mathbb{E}(|\text{OPT} \cap A|) = \sum_{x \in \text{OPT}} \mathbb{P}(x \in A) \geq \frac{|\text{OPT}|}{c}.$$

Lo que quiere decir que el conjunto entregado por el algoritmo no solo es de algún modo cercano al óptimo en peso, si no que además en los elementos que los componen. La propiedad anterior define nuestra tercera noción de competitividad:

Definición 2.3. *Decimos que un algoritmo online para el MSP es cardinal c -competitivo si entrega un conjunto independiente A tal que:*

$$\frac{\mathbb{E}(|\text{OPT} \cap A|)}{|\text{OPT}|} \geq \frac{1}{c}.$$

En el Capítulo 7 presentamos un algoritmo cardinal $O(1)$ -competitivo para el MSP.

A continuación mostramos un cuadro con nuestros principales avances y los mejores resultados obtenidos hasta ahora en los mismo problemas, :

Tipo de Matroide	Mejor Algoritmo Previo	Nuestro Algoritmo
Transversal	e -competitivo [11]	Fuertemente e -competitivo
Gráfica	$2e$ -competitivo [13]	Fuertemente 4-competitivo
Representable k -sparsa	ke -competitivo [19]	Fuertemente $k^{\frac{k}{k-1}}$ -competitivo
Laminar	Fuertemente 9, 6-competitivo [16]	Fuertemente $3\sqrt{3}$ -competitivo

Tabla 2.1: Nuestros principales resultados

Para finalizar este capítulo queremos hacer dos observaciones generales respecto a los resultados que puedan obtenerse. Una es que n puede aumentarse a cualquier $N > n$ que se desee agregando loops fantasmas a la matroide. Como los loops no pertenecen a ningún

conjunto independiente no pueden ser elegidos. Para mantener la hipótesis de que los elementos llegan en orden aleatorio, el algoritmo puede simular la llegada de los loops. Previo a que el algoritmo comience se elige de manera equiprobable un conjunto A de entre todos los subconjuntos de $[N]$ que tienen tamaño n . Luego cuando el algoritmo deba ver el i -ésimo elemento, se le revelará un loop fantasma si $i \notin A$ o el elemento que realmente viene si $i \in A$. Debido a esto, siempre podemos suponer n tan grande como queramos, razón por la cual no nos preocupamos de los resultados asintóticos.

La otra observación es que la matroide 1-uniforme es un caso particular de matroides transversales y laminares, como ya se dijo en los preliminares. También es un caso particular de matroide gráfica (considere un grafo con solo dos vértices y n aristas paralelas entre ellas) y por consiguiente de matroides representables. Dado que Dynkin demostró que en el problema clásico no es posible obtener algo mejor que e en competitividad [5], tampoco es posible lograr superar esa barrera en los casos mencionados.

Capítulo 3

Problema de Asignación Online y MSP en Matroides Transversales

En este capítulo estudiamos el problema de la secretaria en matroides transversales, recordemos que definimos estas matroides en el Capítulo 1 (Definición 1.19). Ilustramos el problema online con una venta de terrenos. Supongamos que hacemos una venta de terrenos y tenemos inscritos n potenciales compradores que llegarán a ver los terrenos. Cada comprador desea adquirir un único terreno, pero puede que le interesen solo algunos de los ofrecidos. Por ejemplo, puede ser que un comprador quiera un terreno plano, o uno por el que pase un riachuelo, etc. Al azar se le asigna a cada posible comprador un número de llegada. Luego de verlos, cada comprador revelará qué terrenos le interesan y ofrecerá una suma por cualquiera de ellos. Antes de que el comprador se retire y llegue el próximo, debemos decidir si le vendemos o no alguno de los terrenos que le interesan

Haciendo el paralelo con el MSP en matroides transversales tenemos que los terrenos corresponden a los vértices del lado derecho y los compradores corresponden a los vértices del lado izquierdo. Lo que ofrece un comprador corresponde a su peso, y los terrenos que le interesan determinan cuáles son las aristas del grafo. Suponemos entonces que el algoritmo conoce de antemano el número de vértices del lado izquierdo ($|E| = n$) y los vértices del lado derecho (F). Mientras que se revelan de manera online los vecinos y los pesos de los vértices del lado izquierdo. Si bien en el MSP que definimos en el Capítulo 2 los vecinos serían conocidos de antemano, aquí consideramos que se revelan de modo online por razones históricas del problema.

El año 2007, Babaioff et al. [1] presentaron un algoritmo que es $4d$ -competitivo para el caso en que el grado de los vértices de E está acotado superiormente por d , es decir, cada comprador se interesa en a lo más d terrenos. El algoritmo funciona como sigue. Primero tomamos como muestra a los primeros $n/2$ compradores en llegar. Luego fijamos un precio a cada terreno que corresponde al máximo precio que se haya ofrecido por él durante la muestra. Después de la muestra, cada vez que llega un nuevo comprador, se le vende el terreno disponible con menor precio fijado de los que le interesan, siempre y cuando su oferta supere dicho precio.

En 2008, Dimitrov et al. [3] presentan el primer algoritmo $O(1)$ -competitivo, y prueban que es 16-competitivo. El algoritmo es bastante sencillo: se toma una muestra E_S de los primeros S compradores que llegan, con $S \sim \text{Binomial}(n, 1/2)$. Cada vez que llega un nuevo comprador x , se usa el algoritmo glotón para emparejamientos en el grafo inducido por $E_S + x$ y F . Si el comprador x es emparejado por glotón con el terreno f y éste no ha sido vendido aún, entonces le vendemos f a x .

Existe un problema que generaliza un poco el MSP en matroides transversales y es el de emparejamiento online en grafos bipartitos. La única diferencia entre este problema y el anterior es que ahora los pesos van en las aristas. Haciendo el paralelo con la venta de terrenos, ahora cada comprador puede hacer una oferta distinta por cada terreno de su interés. En 2009, Korula y Pál [13] adaptan el algoritmo anterior a este caso y muestran que es 8-competitivo.

Finalmente en 2013, Kesselheim et al. [11] muestran el algoritmo e -competitivo que describimos en la siguiente sección. El algoritmo se asemeja bastante al anterior, siendo la principal diferencia que en vez de usar glotón en la muestra con el nuevo elemento (que da una 2-aproximación del problema de emparejamiento de peso máximo en el caso offline), se calcula el emparejamiento óptimo de todo lo ya visto en cada etapa. Aunque los algoritmos son similares, el análisis de los algoritmo cambia, explotándose mucho en el último el hecho de que los elementos llegan en orden aleatorio. Por la observación hecha al final del Capítulo 2, este algoritmo es óptimo.

3.1. Algoritmo óptimo para el problema de la secretaria en matroides transversales

Este algoritmo fue presentado en [11], donde además se prueba que es e -competitivo para el problema de emparejamiento online en grafos bipartitos y por lo tanto también resuelve el MSP en matroides transversales por ser este un caso particular en que el peso de las aristas solo depende del extremo izquierdo (si se quiere, podemos decir que el peso va en los vértices del lado izquierdo y no en las aristas). Este resultado es óptimo, ya que el problema de la secretaria corresponde al caso particular en que F tiene un solo vértice y cada vértice de E está conectado a él. En esta sección modificamos el análisis de la competitividad para probar que en el caso de matroides transversales el algoritmo es fuertemente e -competitivo. El algoritmo adaptado para el MSP en matroides transversales es el siguiente:

En palabras: primero muestreamos una fracción de E . Luego, por cada elemento x_t que llegue después de la muestra, se calcula el emparejamiento óptimo M_t de lo ya visto y emparejamos x con su pareja en M_t si ésta existe y se encuentra disponible.

Es claro que el conjunto A devuelto por el Algoritmo 4 es independiente, ya que existe M que lo cubre.

Notemos también que si el peso va en los vértices, M_t cubre exactamente a los vértices de OPT_t . En efecto, sea N_t un emparejamiento que cubre los vértices de OPT_t , y sea $B \subseteq E_t$ el

Algoritmo 4

1: $s \leftarrow \lfloor \frac{n}{e} \rfloor$;
2: $A, M \leftarrow \phi$;
3: **Para** $t = s + 1 \dots n$ **hacer**
4: $M_t \leftarrow$ emparejamiento óptimo en $G[E_t \cup F]$;
5: **Si** x_t está M_t -cubierto **entonces**
6: $f_t \leftarrow$ la pareja de x_t en M_t ;
7: **Si** f_t está M -expuesto **entonces**
8: $M \leftarrow M + (x_t, f_t)$;
9: $A \leftarrow A + x_t$;
10: **Devolver** A ;

conjunto de vértices cubiertos por M_t , entonces:

$$w(B) = w(M_t) \geq w(N_t) = w(\text{OPT}_t).$$

Lo que implica que B es óptimo en $\mathcal{M}[E_t]$ al igual que OPT_t . Por el Lema 1.15 tenemos entonces que $B = \text{OPT}_t$. Además, lo anterior implica que podemos encontrar un emparejamiento óptimo por medio del algoritmo glotón para matroides transversales: para chequear si un conjunto de vértices $V \subseteq E$ es independiente basta usar un algoritmo que calcule emparejamiento de cardinal máximo en $G[V \cup F]$ y chequear si cubre todo V . De este modo no necesitamos conocer los verdaderos pesos de los elementos, es suficiente que podamos rankearlos. Veamos ahora la competitividad del algoritmo.

Teorema 3.1. *El Algoritmo 4 es fuertemente e -competitivo para el MSP en matroides transversales.*

DEMOSTRACIÓN. Tomemos $x \in \text{OPT}$ y supongamos que llega en la posición $t(x) = t > s$. Gracias al Lema 1.16 tenemos que $x_t \in \text{OPT}_t$ y por la observación hecha antes del teorema x_t será cubierto por M_t . Así, para que x_t sea aceptado basta que f_t no haya sido emparejado en M . Tenemos entonces que:

$$\begin{aligned} \mathbb{P}(x_t \in A) &\geq \mathbb{P}\left(\bigcap_{i=s+1}^{t-1} f_i \neq f_t\right) \\ &\geq \prod_{i=s+1}^{t-1} \frac{i-1}{i}. \end{aligned}$$

Donde en la última desigualdad usamos la misma técnica que cuando analizamos la solución del problema clásico. Recordemos que interpretamos orden aleatorio de los vértices del siguiente modo: primero se elige E_{t-1} , el conjunto de los $t-1$ primeros vértices, y luego se elige al azar quien ocupa el puesto $t-1$, quien el $t-2$, etc. Así, para elegir al i -ésimo vértice tenemos i opciones, de las cuales sin importar E_{t-1} ni los vértices previamente elegidos, a lo más hay uno (la pareja de f_t en M_i si existe) que no debe ocupar la i -ésima posición. Por lo tanto, usando la propiedad telescópica y reemplazando el valor de s tenemos que:

$$\mathbb{P}(x_t \in A) \geq \frac{\lfloor n/e \rfloor}{t-1}.$$

Sea $x \in \text{OPT}$, usando probabilidades totales:

$$\begin{aligned}
\mathbb{P}(x \in A) &= \sum_{t=\lceil n/e \rceil}^n \mathbb{P}(x \in A | x = x_t) \frac{1}{n} \\
&\geq \sum_{t=\lceil n/e \rceil}^n \frac{\lfloor n/e \rfloor}{n(t-1)} \\
&= \frac{\lfloor n/e \rfloor}{n} \sum_{t=\lceil n/e \rceil}^{n-1} \frac{1}{t} \\
&> \left(\frac{1}{e} - \frac{1}{n} \right) \int_{\lceil n/e \rceil}^n \frac{1}{\lfloor t \rfloor} dt \\
&> \left(\frac{1}{e} - \frac{1}{n} \right) \int_{n/e}^n \frac{1}{t} dt \\
&= \frac{1}{e} - \frac{1}{n}.
\end{aligned}$$

Como se puede ver, el resultado es en realidad asintótico en n . Como se mencionó en el Capítulo 2, n puede hacerse tan grande como se quiera, de modo que ese término extra no es muy relevante. \square

Presentamos a continuación una pequeña modificación en la forma de muestrear que nos permite deshacernos del término $1/n$ entregando un resultado más limpio. Además esta modificación nos permite mostrar una técnica de acotamiento que volveremos a usar en capítulos siguientes. Sea $p \in (0, 1)$ variable que optimizaremos más adelante, consideremos el siguiente algoritmo:

Algoritmo 5

- 1: Se simulan n v.a.'s independientes con distribución uniforme $(0, 1)$ y se ordenan de forma creciente obteniendo una secuencia U_1, \dots, U_n ;
 - 2: $s \leftarrow \text{máx}\{i \in [n] : U_i < p\}$;
 - 3: $A, M \leftarrow \phi$;
 - 4: **Para** $t = s + 1 \dots n$ **hacer**
 - 5: $M_t \leftarrow$ emparejamiento óptimo en $G[E_t \cup F]$;
 - 6: **Si** x_t está M_t -cubierto **entonces**
 - 7: $f_t \leftarrow$ la pareja de x_t en M_t ;
 - 8: **Si** f_t está M -expuesto **entonces**
 - 9: $M \leftarrow M + (x_t, f_t)$;
 - 10: $A \leftarrow A + x_t$;
 - 11: **Devolver** A ;
-

La única diferencia entre este algoritmo y el anterior es que ahora el tamaño de la muestra es aleatorio. Usamos esta misma técnica en el Capítulo 2 para obtener la competitividad $1/e$ en el problema clásico. En lo que queda de este capítulo, hablamos del tiempo de llegada de un elemento para referirnos al U_i que se le asigna, no a su momento de llegada relativo a los

demás elementos.

Teorema 3.2. *El Algoritmo 5 es fuertemente e -competitivo para el MSP en matroides transversales.*

DEMOSTRACIÓN. Sea $x \in \text{OPT}$, definimos tres v.a.'s:

- T : tiempo de llegada de x .
- S : número de elementos en la muestra sin x .
- K : número de elementos que llegan antes que x .

Sea $f_{T,K,S}$ la densidad conjunta del vector aleatorio (T, K, S) . Tenemos que:

$$\mathbb{P}(x \in A) = \int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^{n-1} \mathbb{P}(x \in A \mid T = t, K = k, S = s) f_{T,K,S}(t, k, s) dt.$$

Si $T < p$ entonces x cae en la muestra y no puede ser elegido, por esta razón la integral parte de p . Dado que x no está en la muestra, ésta puede tener a lo más $n - 1$ elementos, por lo cual la suma en s llega hasta $n - 1$.

Para calcular $\mathbb{P}(x \in A \mid T = t, K = k, S = s)$ sabiendo que $t \geq p$, lo único que importa es que x es el elemento $k + 1$ en llegar y que la muestra tiene tamaño s . Por los mismos argumentos usados en la demostración del Teorema 3.1 obtenemos que:

$$\mathbb{P}(x \in A \mid T = t, K = k, S = s) \geq \mathbb{P}\left(\bigcap_{i=s+1}^k f_i \neq f_{k+1}\right) \geq \cdots \geq \frac{s}{k}.$$

Por lo tanto:

$$\begin{aligned} \mathbb{P}(x \in A) &\geq \int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^{n-1} \frac{s}{k} f_{T,K,S}(t, k, s) dt \\ &= \int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^{n-1} \frac{s}{k} f_{S|T,K}(s|T = t, K = k) f_{T,K}(t, k) dt \\ &= \int_p^1 \sum_{k=0}^{n-1} \frac{1}{k} f_{T,K}(t, k) \left(\sum_{s=0}^{n-1} s f_{S|T,K}(s|T = t, K = k) \right) dt \\ &= \int_p^1 \sum_{k=0}^{n-1} \frac{1}{k} f_{T,K}(t, k) \mathbb{E}(S|T = t, K = k) dt. \end{aligned}$$

Notemos que condicional a $T = t > p$ y $K = k$, S distribuye como una variable Binomial($k, p/t$) ya que cada una de las k variables uniformes que llegan antes del tiempo t tiene probabilidad

p/t de llegar antes del tiempo p . Como la esperanza condicional de S es pk/t , sigue que:

$$\begin{aligned}
 \mathbb{P}(x \in A) &\geq \int_p^1 \sum_{k=0}^{n-1} \frac{p}{t} f_{T,K}(t, k) dt \\
 &= \int_p^1 \frac{p}{t} \sum_{k=0}^{n-1} f_{K|T}(k|T=t) f_T(t) dt \\
 &= \int_p^1 \frac{p}{t} dt \\
 &= -p \ln(p).
 \end{aligned}$$

En el desarrollo anterior usamos que $T \sim \text{Uniforme}(0, 1)$ por lo que $f_T(t) = 1$ para $0 < t < 1$. Además $f_{K|T}(k|t)$ es una densidad discreta, por lo que al sumarse en todos los valores posibles de k el resultado es 1. Finalmente, ya optimizamos la función $-p \ln(p)$ en el Capítulo 2, sabemos que se maximiza en $p = 1/e$ y que obtenemos una garantía de $1/e$, lo que concluye la demostración. \square

Observemos que las demostraciones que hicimos de competitividad fuerte no funcionan para el problema en que los pesos van en las aristas y no en los vértices. Cuando los pesos van en los vértices podemos argumentar que si tomamos un vértice $x \in \text{OPT}$ entonces $x \in \text{OPT}(D)$, $\forall D \subseteq E$ que contenga a x . Pero esto no es cierto en el problema de emparejamiento online en grafos bipartitos. Por ejemplo, consideremos la siguiente figura:



Figura 3.1: A la izquierda, cuando solo conocemos x_1 y x_2 , el emparejamiento óptimo contiene a las aristas de pesos 6 y 8 (aquí glotón no funciona). A la derecha, cuando se revela x_3 , el óptimo queda formado por las aristas de pesos 10 y 5. Vemos que la arista de peso 10 que pertenece a OPT , no pertenece a OPT_2 .

Para terminar esta sección, vale la pena notar que no se usaron las hipótesis típicas del MSP. Con las hipótesis usuales, sabríamos de antemano el grafo completo y bastaría entregar al final un conjunto independiente. Para la venta de terrenos esto significa dos cosas. Primero que sabríamos los terrenos que le interesan a cada comprador desde un comienzo. Y segundo que al momento de hacer una venta bastaría con comprometernos a reservarle al comprador alguno de los terrenos que le interesan. No sería necesario especificar qué terreno recibe cada comprador sino hasta el final del proceso. Una vez que todo termine debemos ser capaces de cumplir todos nuestros compromisos, esto es encontrar un emparejamiento factible entre los compradores que aceptamos y los terrenos. Pese a las restricciones adicionales que impusimos, logramos un resultado óptimo, lo que significa que aún si quitamos estas restricciones no seremos capaces de obtener una mejor garantía.

Capítulo 4

Problema Gráfico de la Secretaria

Ahora estudiamos otro caso particular del MSP, cuando la matroide es una matroide gráfica. Recordemos que en el Capítulo 1 definimos estas matroides (Definición 1.3). Para el MSP suponemos que G es conocido y que las aristas revelan su peso en orden aleatorio. El algoritmo debe decidir si acepta o no una arista al momento de saber su peso, sin posibilidad de arrepentirse luego. SPG asumimos que no hay loops, ya que si hubieran, podemos hacer que el algoritmo simplemente los rechace ya que no pertenecen a ningún conjunto independiente.

En un artículo de Babaioff et al. [1] de 2007, se presenta un algoritmo 16 -competitivo que es una adaptación de su algoritmo $4d$ -competitivo para matroides transversales con grado izquierdo acotado por d . Lo que hacen allí es crear un grafo bipartito G' a partir de G donde el lado izquierdo corresponde a las aristas de G y el derecho a los vértices de G : un vértice se conecta a una arista G' si era extremo de dicha arista en G , de este modo los vértices del lado izquierdo (aristas de G) quedan con grado 2 . Se puede probar que todo bosque en G puede ser representado por un emparejamiento en G' , sin embargo no todo emparejamiento en G' corresponde a un bosque en G . Por eso se agrega la condición de que una arista de G' solo puede ser emparejada en G' si no forma ciclos en G con las aristas ya emparejadas.

En 2009, Korula y Pál [13] presentan un algoritmo $2e$ -competitivo que ocupa el algoritmo del problema clásico como subrutina. Su algoritmo hace lo siguiente: considera una numeración de los vértices y dos grafos dirigidos G_1 y G_2 . En uno de ellos las aristas se orientan del vértice menor al mayor y en el otro se orientan de modo contrario. Se elige una de estas orientaciones al azar y luego para cada vértice se intenta escoger la arista más pesada que sale de él ocupando el algoritmo del problema clásico.

En 2013, Soto [19] recupera la misma garantía al ver el problema como un caso particular de matroides representables sparsas (ver siguiente capítulo).

A continuación presentamos un algoritmo que es fuertemente 4 -competitivo, superando así el $2e$ logrado en [13] y [19].

4.1. Algoritmo para el Problema Gráfico de la Secretaria

Si bien trabajamos con grafos no dirigidos, en nuestro algoritmo vamos asignando direcciones a las aristas. Esto permite entender mejor por qué y cómo funciona el algoritmo. De hecho, durante el algoritmo guardamos dos bosques, A y \vec{A} . El primero es la solución que se entrega al terminar, mientras que \vec{A} es una orientación de A que no juega un rol en el algoritmo, pero facilita la comprensión de éste. A diferencia del algoritmo de Korula y Pál [13], las direcciones que asignamos a las aristas pueden variar cada vez que una nueva arista llega. Dejamos s , el tamaño de la muestra, como una variable a optimizar más adelante.

Algoritmo 6

- 1: Elegimos una numeración arbitraria de los vértices del grafo;
 - 2: Fijamos s ;
 - 3: $A, \vec{A} \leftarrow \phi$;
 - 4: $D \leftarrow V$;
 - 5: **Para** $t = s + 1 \dots n$ **hacer**
 - 6: Calculamos OPT_t (bosque óptimo en (V, E_t));
 - 7: **Si** $x_t \in \text{OPT}_t$, **entonces**
 - 8: En cada c.c. de OPT_t tomamos el menor vértice y dirigimos las aristas haciendo BFS desde dicho vértice;
 - 9: Sean u_t y v_t la cola y cabeza de x_t respectivamente en esta orientación;
 - 10: **Si** $\{u_t, v_t\} \subseteq D$, **entonces**
 - 11: $D \leftarrow D - v_t$;
 - 12: $A \leftarrow A + x_t$;
 - 13: $\vec{A} \leftarrow \vec{A} + (u_t, v_t)$;
 - 14: **Devolver** A ;
-

En la Figura 4.1 se ejemplifica como se dirigen las aristas en OPT_t .

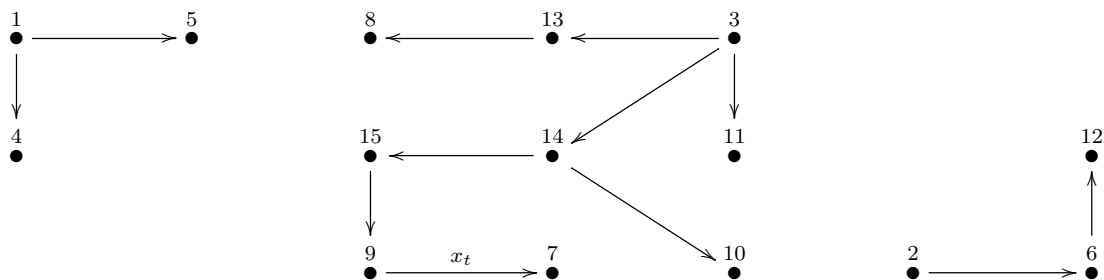


Figura 4.1: En este caso OPT_t tiene 3 c.c., en cada una las aristas se dirigen según BFS partiendo del menor vértice. Si x_t llega en este paso, queda dirigida del vértice 9 al 7. En caso de que x_t sea aceptada, el vértice 7 sale de D y el arco $(9, 7)$ se agrega a \vec{A} .

Como dijimos antes, \vec{A} es una orientación de A . El conjunto D guarda los nodos que no tienen arcos incidentes en \vec{A} . Decimos que estos nodos están disponibles. De este modo, al agregar un arco (u, v) a \vec{A} , el nodo v deja de estar disponible. Como esto solo ocurre si hasta el momento v estaba disponible, nos aseguramos de que en \vec{A} el grado de entrada de cada

nodo es a lo más 1.

Lema 4.1. *El Algoritmo 6 devuelve un bosque.*

DEMOSTRACIÓN. En un comienzo $A = \phi$ es acíclico, veamos que se mantiene libre de ciclos en cada paso. Si no, consideremos A en el primer momento que contiene un ciclo C y sea $x = uv$ la última arista que entró en A . Es claro que x debe pertenecer a C . SPG digamos que x entra orientada desde u hacia v en \vec{A} . Como x fue aceptada, podemos asegurar que antes de que x llegara no habían arcos incidentes ni a u ni a v en \vec{A} . Cuando agregamos el arco (u, v) a \vec{A} , u se mantiene disponible. Como u tiene grado de entrada 0, no pertenece a ningún diciclo en \vec{A} . Por lo tanto, el ciclo C no puede ser un ciclo dirigido en \vec{A} . Como u es parte de C y no tiene arcos entrantes, u debe tener grado de salida 2 en el ciclo. Pero esto implica que algún otro nodo del ciclo debe tener grado de entrada 2 en \vec{A} , lo cual nunca ocurre. Por lo tanto, A se mantiene siempre acíclico. \square

Teorema 4.2. *El Algoritmo 6 es fuertemente 4-competitivo.*

DEMOSTRACIÓN. Debemos probar que si $x \in \text{OPT}$ entonces $\mathbb{P}(x \in A) \geq 1/4$. Lo primero a observar es que si $x_t \in \text{OPT}$ entonces $x_t \in \text{OPT}_t$, esto gracias al Lema 1.16. Así, para que un elemento del óptimo x_t sea aceptado basta que cuando llegue, los extremos estén disponibles. Antes de continuar, definimos algo de notación:

$$N_i^-(x) := \{y \in \text{OPT}_i : y \text{ apunta a algún extremo de } x \text{ según la orientación de } \text{OPT}_i\}.$$

Notemos que $\forall i, x |N_i^-(x)| \leq 2$, esto debido a cómo se orienta OPT_i . Al usar BFS solo se puede entrar una vez a cada vértice y x tiene solo dos extremos.

Supongamos que $x_t \in \text{OPT}$ con $t > s$, entonces:

$$\begin{aligned} \mathbb{P}(x_t \in A) &\geq \mathbb{P}\left(\bigcap_{i=s+1}^{t-1} x_i \notin N_i^-(x_t)\right) \\ &\geq \prod_{i=s+1}^{t-1} \frac{i-2}{i} \\ &= \frac{(s-1)s}{(t-2)(t-1)}. \end{aligned}$$

Donde para la última desigualdad usamos el mismo razonamiento que en capítulos anteriores. Recordemos que simulamos la llegada de las aristas anteriores a x_t del siguiente modo: primero se elige el conjunto de las $t-1$ primeras aristas aleatoriamente dentro de $E - x_t$. Luego se elige al azar cuál va en el puesto $t-1$, cuál va en el $t-2$, etc. Así, en cada paso tenemos $i - |N_i^-(x_t)| \geq i-2$ elecciones para x_i que no bloquean a la arista x_t de un total de i posibilidades.

Sea $x \in \text{OPT}$, usando probabilidades totales tenemos:

$$\begin{aligned}
\mathbb{P}(x \in A) &= \sum_{t=s+1}^n \mathbb{P}(x \in A | x = x_t) \frac{1}{n} \\
&\geq \sum_{t=s+1}^n \frac{(s-1)s}{n(t-2)(t-1)} \\
&= \frac{(s-1)s}{n} \sum_{t=s+1}^n \frac{1}{(t-2)(t-1)} \\
&= \frac{(s-1)s}{n} \sum_{t=s+1}^n \left[\frac{1}{(t-2)} - \frac{1}{(t-1)} \right] \\
&= \frac{(s-1)s}{n} \left(\frac{1}{s-1} - \frac{1}{n-1} \right) \\
&= \frac{s}{n} \left(1 - \frac{s-1}{n-1} \right) \\
&> \frac{s}{n} \left(1 - \frac{s}{n} \right) .
\end{aligned}$$

La última desigualdad se debe a que como $s < n$, al aumentar en uno el numerador y denominador de $(s-1)/(n-1)$ obtenemos una fracción más cercana a 1. Tomando $p = s/n$ y optimizando la función $p(1-p)$ obtenemos que el p óptimo es $1/2$ que logra una garantía de $1/4$. Para asegurar s entero tomamos $s = \lceil n/2 \rceil$, veamos que con esto se mantiene la misma garantía. Si n es par, $\lceil n/2 \rceil = n/2$ y no hay nada que probar. Supongamos que es impar, $n = 2m + 1$. Entonces $s = \lceil n/2 \rceil = m + 1$, reemplazando en la penúltima línea del desarrollo anterior tenemos:

$$\begin{aligned}
\mathbb{P}(x \in A) &\geq \frac{m+1}{2m+1} \left(1 - \frac{m}{2m} \right) \\
&> \frac{m+1}{2m+2} \cdot \frac{1}{2} \\
&= \frac{1}{4} .
\end{aligned}$$

□

Notemos que para este algoritmo, al igual que para el de matroides transversales, no es necesario conocer el grafo de antemano ya que en todo momento se usa solo la información ya revelada. Tampoco es necesario que se revele el peso real de las aristas, solo necesitamos ser capaces de rankearlas para calcular OPT_t en cada paso.

En la próxima sección supondremos que el grafo del problema no contiene ciclos cortos y haremos una modificación del algoritmo para aprovechar esta estructura y obtener una mejor garantía.

4.2. Problema Gráfico de la Secretaria en Grafos con Cintura Ancha

Supongamos ahora que el grafo asociado a la matroide gráfica del problema tiene cintura ancha, esto es, que el ciclo más corto tiene muchas aristas. Para este caso haremos algunas modificaciones para obtener una mejor competitividad. Dos son los cambios importantes que le hacemos al Algoritmo 6. El primero es que ahora al considerar una arista solo nos fijaremos en uno de sus extremos. El segundo cambio es que ahora iremos guardando dos conjuntos, A y A_q . Cada vez que una arista x vaya a ser incluida a la solución, se lanzará una moneda cargada con probabilidad q de salir cara. Si sale cara, x se incluye en ambos conjuntos. Si sale sello solo se incluye en A_q , con esto agregamos una barrera de entrada a los elementos. En realidad, no es necesario definir A_q para que el algoritmo funcione, pero tenerlo simplifica el posterior análisis. Tomamos una muestra de tamaño aleatorio del mismo modo que en el Capítulo 3. El algoritmo va como sigue:

Algoritmo 7

- 1: Elegimos una numeración arbitraria de los vértices del grafo;
 - 2: Simulamos n v.a.'s independientes con distribución Bernoulli(q) obteniendo una secuencia X_1, \dots, X_n ;
 - 3: Simulamos n v.a.'s independientes con distribución Uniforme($0, 1$) y las ordenamos de forma creciente obteniendo una secuencia U_1, \dots, U_n ;
 - 4: $s \leftarrow \max\{i \in [n] : U_i < p\}$;
 - 5: $A, A_q \leftarrow \phi$;
 - 6: $D \leftarrow V$;
 - 7: **Para** $t = s + 1 \dots n$ **hacer**
 - 8: Calculamos OPT_t ;
 - 9: **Si** $x_t \in \text{OPT}_t$ **entonces**
 - 10: En cada c.c. de OPT_t tomamos el menor vértice y dirigimos las aristas haciendo BFS desde dicho vértice;
 - 11: Sean u_t y v_t la cola y cabeza de x_t respectivamente en esta orientación;
 - 12: **Si** $v_t \in D$ **entonces**
 - 13: $D \leftarrow D - v_t$;
 - 14: $A_q \leftarrow A_q + x_t$;
 - 15: **Si** x_t no crea ciclos en A y $X_t = 1$ **entonces**
 - 16: $A \leftarrow A + x_t$;
 - 17: **Devolver** A ;
-

Notemos en primer lugar que ahora protegemos solo un extremo de cada arista. Las variables X_i tienen dos efectos, por un lado pueden obligarnos a rechazar aristas que queremos aceptar. Pero por otro lado hacen más difícil que se cree un ciclo, y esto ayuda a aceptar aristas. Esta vez es claro que el algoritmo entrega un bosque, veamos un lema sobre la estructura de A_q .

Lema 4.3. *Las componentes conexas de A_q tienen a lo más un ciclo.*

DEMOSTRACIÓN. Pensemos A_q con la orientación que se le da a las aristas cuando entran en A_q . Podemos ver que todos los vértices tienen grado de entrada menor o igual que 1, lo que implica que el grado de entrada promedio es menor o igual a 1 en cada c.c. Para probar el resultado veamos que si se forman dos ciclos en una misma c.c. el grado de entrada promedio de dicha componente debe ser mayor que 1. Notemos que la suma de los grados de entrada de una c.c. corresponde al número de aristas de la componente. Para un árbol sabemos que el número de aristas es uno menos que el número de vértices. Luego, una componente de k vértices con un único ciclo tendrá k aristas y una componente de k vértices con dos o más ciclos tendrá al menos $k+1$ aristas, lo que no puede ocurrir pues el grado promedio de entrada sería mayor a 1. \square

Antes de estudiar la competitividad vamos a ver un pequeño lema de probabilidades que usaremos en el teorema que viene después.

Lema 4.4. *Sea $Z \sim \text{Binomial}(n, p)$, entonces $\mathbb{E}(Z(Z - 1)) = n(n - 1)p^2$.*

DEMOSTRACIÓN. Recordemos que $\mathbb{E}(Z) = np$ y $\text{Var}(Z) = np(1 - p)$, con esto tenemos que:

$$\begin{aligned}
\mathbb{E}(Z(Z - 1)) &= \mathbb{E}(Z^2) - \mathbb{E}(Z) \\
&= \mathbb{E}(Z^2) - \mathbb{E}(Z)^2 + \mathbb{E}(Z)^2 - \mathbb{E}(Z) \\
&= \text{Var}(Z) + (np)^2 - np \\
&= np(1 - p) + (np)^2 - np \\
&= (np)^2 - np^2 \\
&= n(n - 1)p^2.
\end{aligned}$$

\square

Ahora estamos listos para estudiar la competitividad de nuestro algoritmo.

Teorema 4.5. *El Algoritmo 7 es fuertemente $(q^c p(1 - p) + (q - q^c)p \ln(1/p))^{-1}$ -competitivo para grafos con cintura c .*

DEMOSTRACIÓN. Sea $x \in \text{OPT}$, debemos probar que $\mathbb{P}(x \in A) \geq q^c p(1 - p) + (q - q^c)p \ln(1/p)$. Al igual que en la demostración del Teorema 3.2 definimos las v.a.'s T , S y K :

- T : tiempo de llegada de x .
- S : número de elementos en la muestra sin x .
- K : número de elementos que llegan antes que x .

Notemos que si $K = k$, entonces $x = x_{k+1}$. Sean $t \in (p, 1)$, $s, k \in \{0, \dots, n - 1\}$, para abreviar anotamos:

$$\mathbb{P}_{t,s,k}(\cdot) := \mathbb{P}(\cdot \mid T = t, K = k, S = s).$$

Además dados t , s y k definimos los siguientes eventos:

- $L_1 : v_{k+1} \in D$.

- $L_2 : \{v_{k+1}, u_{k+1}\} \subseteq D$.
- $L_3 : v_{k+1} \in D$ y $u_{k+1} \notin D$.
- $Q : x_{k+1}$ no crea ciclos en A .

Sobre estos eventos debemos tener en cuenta dos cosas. Lo primero es que L_1 es la unión disjunta de L_2 y L_3 . Lo segundo que debemos notar es que L_2 implica Q , la demostración de esto es la misma prácticamente que la del Lema 4.1. Tenemos que:

$$\begin{aligned}
\mathbb{P}_{t,s,k}(x \in A) &= \mathbb{P}_{t,s,k}(X_{k+1} = 1, L_1, Q) \\
&= q\mathbb{P}_{t,s,k}(L_1, Q) \\
&= q[\mathbb{P}_{t,s,k}(L_2, Q) + \mathbb{P}_{t,s,k}(L_3, Q)] \\
&= q[\mathbb{P}_{t,s,k}(L_2) + \mathbb{P}_{t,s,k}(L_3)\mathbb{P}_{t,s,k}(Q|L_3)] \\
&= q[\mathbb{P}_{t,s,k}(L_2) + (\mathbb{P}_{t,s,k}(L_1) - \mathbb{P}_{t,s,k}(L_2))\mathbb{P}_{t,s,k}(Q|L_3)].
\end{aligned}$$

Acotemos primero el término $\mathbb{P}_{t,s,k}(Q|L_3)$. Notemos que si x_{k+1} cierra un ciclo en A , también cierra un ciclo en A_q . Gracias al Lema 4.3, si x_{k+1} cierra un ciclo en A_q , éste es único. Sea C el ciclo cerrado por x_{k+1} y sea $\{x_j\}_{j \in J}$ el conjunto de aristas que forman C . Sabemos que $k+1 \in J$ y dado que el grafo tiene cintura c , $|J| \geq c$. Luego:

$$\mathbb{P}(Q|L_3) = 1 - \mathbb{P}(\overline{Q}|L_3) \geq 1 - \mathbb{P}(X_j = 1, \forall j \in J - \{k+1\}) \geq 1 - q^{c-1}.$$

Al unir lo que llevamos obtenemos que:

$$\begin{aligned}
\mathbb{P}_{t,s,k}(x \in A) &= q[\mathbb{P}_{t,s,k}(L_2) + (\mathbb{P}_{t,s,k}(L_1) - \mathbb{P}_{t,s,k}(L_2))\mathbb{P}_{t,s,k}(Q|L_3)] \\
&\geq q[\mathbb{P}_{t,s,k}(L_2) + (\mathbb{P}_{t,s,k}(L_1) - \mathbb{P}_{t,s,k}(L_2))(1 - q^{c-1})] \\
&= q[\mathbb{P}_{t,s,k}(L_2)q^{c-1} + \mathbb{P}_{t,s,k}(L_1)(1 - q^{c-1})].
\end{aligned}$$

Podemos acotar la probabilidad de L_2 tal como hicimos en la demostración del Teorema 4.2. Lo único que importa es que la muestra tiene tamaño s y que han llegado k elementos antes de x , obtenemos que:

$$\mathbb{P}_{t,s,k}(L_2) \geq \frac{(s-1)s}{(k-1)k}.$$

Para acotar la probabilidad de L_1 usamos el mismo razonamiento que ocupamos en la demostración del Teorema 3.2:

$$\mathbb{P}_{t,s,k}(L_1) \geq \mathbb{P}_{t,s,k}\left(\bigcap_{i=s+1}^k \{v_i \neq v_{k+1}\}\right) \geq \prod_{i=s+1}^k \frac{i-1}{i} = \frac{s}{k}.$$

Juntando lo anterior, nos queda que:

$$\begin{aligned}
\mathbb{P}_{t,s,k}(x \in A) &\geq q[\mathbb{P}_{t,s,k}(L_2)q^{c-1} + \mathbb{P}_{t,s,k}(L_1)(1 - q^{c-1})] \\
&\geq q\left[\frac{(s-1)s}{(k-1)k}q^{c-1} + \frac{s}{k}(1 - q^{c-1})\right] \\
&= \frac{(s-1)s}{(k-1)k}q^c + \frac{s}{k}(q - q^c).
\end{aligned}$$

Para concluir tenemos que:

$$\begin{aligned}
\mathbb{P}(x \in A) &= \int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^{n-1} \mathbb{P}_{t,s,k}(x \in A) f_{T,K,S}(t, k, s) dt \\
&\geq \int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^{n-1} \left(\frac{(s-1)s}{(k-1)k} q^c + \frac{s}{k} (q - q^c) \right) f_{T,K,S}(t, k, s) dt \\
&= \underbrace{q^c \int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^{n-1} \frac{(s-1)s}{(k-1)k} f_{T,K,S}(t, k, s) dt}_{R_1} + (q - q^c) \underbrace{\int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^n \frac{s}{k} f_{T,K,S}(t, k, s) dt}_{R_2}.
\end{aligned}$$

De la demostración del Teorema 3.2 tenemos que $R_2 = p \ln(1/p)$. Calculemos R_1 :

$$\begin{aligned}
R_1 &= \int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^{n-1} \frac{(s-1)s}{(k-1)k} f_{T,K,S}(t, k, s) dt \\
&= \int_p^1 \sum_{k=0}^{n-1} \sum_{s=0}^{n-1} \frac{(s-1)s}{(k-1)k} f_{S|T,K}(s|T=t, K=k) f_{T,K}(t, k) dt \\
&= \int_p^1 \sum_{k=0}^{n-1} \frac{1}{k(k-1)} f_{T,K}(t, k) \sum_{s=0}^{n-1} (s-1)s f_{S|T,K}(s|T=t, K=k) dt \\
&= \int_p^1 \sum_{k=0}^{n-1} \frac{1}{(k-1)k} f_{T,K}(t, k) \mathbb{E}((S-1)S|T=t, K=k) dt.
\end{aligned}$$

Pero condicional a $T = t > p$ y $K = k$, S distribuye como una binomial de parámetros k y p/t , ya que cada una de las k variables uniformes que llegan antes del tiempo t tiene probabilidad p/t de llegar antes del tiempo p . Usando el Lema 4.4 sigue que:

$$\begin{aligned}
R_1 &= \int_p^1 \sum_{k=0}^{n-1} \frac{p^2}{t^2} f_{T,K}(t, k) dt \\
&= p^2 \int_p^1 t^{-2} \sum_{k=0}^{n-1} f_{T,K}(t, k) dt \\
&= p^2 \int_p^1 t^{-2} f_T(t) dt \\
&= p^2 \int_p^1 t^{-2} dt \\
&= p(1-p).
\end{aligned}$$

Finalmente, juntando todo llegamos al resultado buscado:

$$\mathbb{P}(x \in A) \geq q^c p(1 - p) + (q - q^c)p \ln(1/p).$$

□

Si tomamos $q = 1$ y $p = 1/2$ recuperamos el $1/4$ de la sección anterior. Ahora veamos lo que podemos lograr si c es muy grande.

Teorema 4.6. *La competitividad tiende a e cuando $c \rightarrow \infty$.*

DEMOSTRACIÓN. Con la garantía obtenida no es fácil elegir p, q óptimos en función de c , pero para probar este resultado basta tomar:

$$q = \frac{1}{\sqrt[c]{c}}, \quad p = \frac{1}{e}.$$

Reemplazando obtenemos que si $x \in \text{OPT}$:

$$\mathbb{P}(x \in A) \geq \frac{1}{ce} \left(1 - \frac{1}{e}\right) + \left(\frac{1}{\sqrt[c]{c}} - \frac{1}{c}\right) \frac{1}{e}.$$

Que converge a $1/e$ cuando c tiende a infinito.

□

Capítulo 5

MSP en Matroides Representables con Representación k -sparsa

Una clase bastante amplia de matroides es la clase de las matroides representables (ver Definición 1.4). Como comentamos en el Capítulo 1, las matroides gráficas son representables. También son representables las matroides transversales (ver [8], Teorema 7.2) y por consiguiente las de partición unitaria y las k -uniformes también son representables.

Dada una matroide representable \mathcal{M} , surgen dos preguntas naturales:

1. ¿Podemos encontrar una matriz Q tal que $\mathcal{M}[Q] = \mathcal{M}$?
2. ¿Es única dicha matriz Q ? Y si no lo es: ¿Podemos encontrar a partir de Q otra matriz Q' tal que $\mathcal{M}[Q'] = \mathcal{M}[Q]$?

No intentaremos responder la primera pregunta aquí (recomendamos consultar [8], Sección 6.2 para obtener una respuesta). Sobre la segunda pregunta podemos comenzar diciendo que Q no es única. Notemos primero que podemos ponderar cualquier columna de Q por cualquier escalar no nulo sin cambiar los conjuntos de columnas independientes. Si Q tuviera filas nulas, podemos eliminarlas sin que cambien las relaciones de independencia. Si Q tiene m filas, podemos tomar cualquier matriz invertible R de $m \times m$ y tendremos que $\mathcal{M}[RQ] = \mathcal{M}[Q]$. Lo último se debe a que las transformaciones lineales invertibles preservan independencia lineal y multiplicar por R equivale a aplicar una transformación lineal invertible a cada una de las columnas de Q . En resumen, si una matroide es representable, entonces hay muchas matrices que sirven para representarla.

Ahora una definición, decimos que una matriz es k -sparsa si cada columna tiene a lo más k coordenadas no nulas. Por ejemplo, la matriz de incidencia de un grafo es 2-sparsa, cada columna corresponde a una arista y tiene solo dos unos indicando los vértices conectados por la arista. Nos interesa mantener un paralelo con el caso gráfico porque el algoritmo de este capítulo generaliza al Algoritmo 6 del capítulo anterior, las columnas toman el lugar de las aristas y las filas toman de algún modo el lugar de los vértices.

Vamos a presentar un algoritmo online para el MSP en matroides representables $\mathcal{M}[Q]$ con

Q matriz k -sparsa. Siendo honestos, no sabemos cuál es el alcance real de este algoritmo. En general no sabemos bajo que condiciones una matroide representable \mathcal{M} admite una matriz k -sparsa Q que la represente, y de existir Q no sabemos cómo encontrarla. Pensamos que caracterizar este tipo de matroides puede ser un tema de estudio por si mismo. Por ejemplo, una clase interesante de matroides que sí pueden ser representadas por matrices con pocos no ceros por columna, es la clase de las matroides de rigidez [21]. Sin entrar en demasiados detalles, podemos definir una matroide de rigidez \mathcal{M} a partir de un grafo $G = (V, E)$ dibujado en \mathbb{R}^d donde los vértices son puntos de \mathbb{R}^d y las aristas segmentos rectos que conectan dos vértices. Es posible definir una matriz $2d$ -sparsa Q con $|E|$ columnas y $d|V|$ filas, tal que $\mathcal{M} = \mathcal{M}[Q]$. Cada arista en E corresponde a una columna de Q , y las únicas entradas no nulas son aquellas que describen las coordenadas de sus extremos.

Aún no se ha obtenido un algoritmo $O(1)$ -competitivo para matroides representables en general. Recientemente, en un artículo del 2014, Dinitz y Kortsarz [4] estudian el problema en matroides regulares (ver Definición 1.22) y obtienen un algoritmo $9e$ -competitivo apoyados fuertemente en un teorema de Seymour [18] sobre descomposición de matroides regulares.

El caso que estudiamos en este capítulo fue abordado antes en 2013 por Soto [19] quien logró un algoritmo ke -competitivo. En este trabajo superamos este resultado obteniendo un algoritmo fuertemente $\sqrt[k-1]{k^k}$ -competitivo.

5.1. Algoritmo para el MSP en Matroides Representables k -sparsas

Suponemos que tenemos una matroide representable \mathcal{M} y además conocemos una matriz k -sparsa Q de m por n tal que $\mathcal{M}[Q] = \mathcal{M}$. Nuestro input es la matriz Q . Denotamos por E al conjunto de columnas de Q y por F al conjunto de filas. Para $x \in E, f \in F$ denotamos $x(f) := Q(x, f)$ la coordenada en la fila f de x .

Algoritmo 8

- 1: Simulamos S v.a. Binomial(n, p). $s \leftarrow S$;
 - 2: $A \leftarrow \phi$;
 - 3: $D \leftarrow F$;
 - 4: **Para** $t = s + 1 \dots n$ **hacer**
 - 5: Calculamos OPT_t ;
 - 6: **Si** $x_t \in \text{OPT}_t$ **entonces**
 - 7: Construimos el grafo bipartito $G = (\text{OPT}_t \cup F, H)$ donde $xf \in H \Leftrightarrow x(f) \neq 0$;
 - 8: Calculamos M emparejamiento que cubre OPT_t en G ;
 - 9: $f_t \leftarrow$ pareja de x_t en M ;
 - 10: $N_t \leftarrow \{f \in F : x_t(f) \neq 0\}$;
 - 11: **Si** $N_t \subseteq D$ **entonces**
 - 12: $D \leftarrow D - f_t$;
 - 13: $A \leftarrow A + x_t$;
 - 14: **Devolver** A ;
-

Como vimos en los Capítulos 2 y 3, tener una muestra de tamaño fijo puede traer problemas al momento de calcular la garantía, ya que para tener una muestra de tamaño entero debemos redondear. En ambas ocasiones el problema se solucionó aleatorizando el tamaño de la muestra. En esta ocasión consideramos una muestra de tamaño aleatorio desde un principio. Como se ve en el algoritmo, elegimos el tamaño de la muestra simulando una v.a. Binomial(n, p). Más adelante vamos a optimizar el parámetro p .

Para el análisis del algoritmo debemos mostrar tres cosas: que el emparejamiento M buscado existe, que el conjunto A devuelto es l.i. y que el algoritmo es fuertemente c_k -competitivo, con c_k una constante que depende solo de k .

Lema 5.1. *El emparejamiento M buscado en el Algoritmo 8 existe.*

DEMOSTRACIÓN. Para demostrar que M existe vamos a usar el Teorema de Hall. Sea J cualquier subconjunto l.i. de columnas de Q . Veamos que $|N_G(J)| \geq |J|$. Consideremos la matriz B formada por las columnas de J , usando una conocida propiedad del álgebra lineal que nos dice que el rango por columnas es igual al rango por filas tenemos que:

$$|J| = r(B) = \dim\langle\{\text{filas de } B\}\rangle = \dim\langle\{\text{filas no nulas de } B\}\rangle = \dim\langle N_G(J)\rangle \leq |N_G(J)|.$$

Como solo ocupamos que J era l.i. y dado que la independencia se hereda a los subconjuntos, tenemos que para todo $I \subseteq J$, también se cumple que $|I| \leq |N_G(I)|$. Usando el Teorema de Hall concluimos que si J es l.i., existe un emparejamiento en G que cubre J . Como OPT_t es independiente, M existe. \square

Vale la pena observar que podemos encontrar M usando un algoritmo para emparejamiento de cardinal máximo. Además, para que M no dependa del orden de llegada de las columnas, podemos permutar al azar las columnas de OPT_t para construir G antes de calcular M . Después usamos la permutación inversa para saber a que columna se asigna cada fila.

Lema 5.2. *El conjunto A retornado por el Algoritmo 8 es l.i.*

DEMOSTRACIÓN. Sea $A = \{x_{h_1}, \dots, x_{h_{|A|}}\}$, donde la secuencia h_i es creciente. Notemos que por como funciona el algoritmo se tiene la siguiente propiedad para todo i :

$$x_{h_i}(f_{h_i}) \neq 0 \text{ y } x_{h_j}(f_{h_j}) = 0, \forall j > i.$$

En palabras, cada columna x_{h_i} entra asociada a una fila f_{h_i} donde su coordenada es no nula, y todas las columnas que sean agregadas después deben ser nulas en esa fila. Sean $\lambda_1, \dots, \lambda_{|A|}$ tales que:

$$\sum_{i=1}^{|A|} \lambda_i x_{h_i} = \vec{0}.$$

Al mirar la igualdad en la fila f_{h_1} vemos que:

$$0 = \sum_{i=1}^{|A|} \lambda_i x_{h_i}(f_{h_i}) = \lambda_1 x_{h_1}(f_{h_1}).$$

Por lo tanto, $\lambda_1 = 0$. Ahora podemos observar la coordenada f_{h_2} para concluir que $\lambda_2 = 0$. Inductivamente se ve que $\lambda_i = 0, \forall i \in [|A|]$. Por lo tanto, A es l.i. \square

Ahora solo falta estudiar la competitividad del Algoritmo 8, pero antes un lema técnico:

Lema 5.3. Sean $k, s, n \in \mathbb{N}$ tales que $2 \leq k \leq s \leq n$, entonces:

$$\sum_{t=s+1}^n \binom{s}{k} \binom{t-1}{k}^{-1} = \frac{1}{k-1} \left(s - n \binom{s}{k} \binom{n}{k}^{-1} \right).$$

DEMOSTRACIÓN. Probamos esta igualdad por inducción en n . El caso base $n = s$ es directo. Veamos el paso inductivo:

$$\begin{aligned} \sum_{t=s+1}^{n+1} \binom{s}{k} \binom{t-1}{k}^{-1} &= \binom{s}{k} \binom{n}{k}^{-1} + \sum_{t=s+1}^n \binom{s}{k} \binom{t-1}{k}^{-1} \\ &= \binom{s}{k} \binom{n}{k}^{-1} + \frac{1}{k-1} \left(s - n \binom{s}{k} \binom{n}{k}^{-1} \right). \end{aligned}$$

Donde para el último paso usamos la hipótesis de inducción. Reordenando nos queda:

$$\begin{aligned} \binom{s}{k} \binom{n}{k}^{-1} + \frac{1}{k-1} \left(s - n \binom{s}{k} \binom{n}{k}^{-1} \right) &= \frac{1}{k-1} \left(s - (n+1-k) \binom{s}{k} \binom{n}{k}^{-1} \right) \\ &= \frac{1}{k-1} \left(s - \binom{s}{k} \frac{(n+1-k)(n-k)!k!}{n!} \right) \\ &= \frac{1}{k-1} \left(s - \binom{s}{k} \frac{(n+1-k)!k!(n+1)}{(n+1)!} \right) \\ &= \frac{1}{k-1} \left(s - (n+1) \binom{s}{k} \binom{n+1}{k}^{-1} \right). \end{aligned}$$

Lo que concluye la inducción. \square

Teorema 5.4. El Algoritmo 8 es fuertemente $k^{-1}\sqrt{k^k}$ -competitivo para $k \geq 2$.

DEMOSTRACIÓN. Consideremos primero $s \geq k$ fijo, sea $x_t \in \text{OPT}$ con $t > s$ y consideremos el conjunto D antes de la llegada de x_t . Como ya sabemos por el Lema 1.16, $x_t \in \text{OPT}_t$, tenemos entonces:

$$\mathbb{P}(x_t \in A \mid S = s) = \mathbb{P}(N_t \subseteq D) = \mathbb{P} \left(\bigcap_{i=s+1}^{t-1} \{f_i \notin N_t\} \right).$$

Ahora, como la matriz es k -sparsa, $|N_t| \leq k$. Con la interpretación que le damos al orden aleatorio de los primeros $t-1$ elementos, tenemos que al elegir x_i hay i opciones de las cuales siempre hay al menos $i-k$ tales que $f_i \notin N_t$. Por lo tanto:

$$\begin{aligned}
\mathbb{P}\left(\bigcap_{i=s+1}^{t-1} \{f_i \notin N_t\}\right) &\geq \prod_{i=s+1}^{t-1} \frac{i-k}{i} \\
&= \frac{s! \prod_{i=s+1}^{t-1} (i-k)}{(t-1)!} \\
&= \frac{s! \prod_{i=s+1-k}^{t-1-k} i}{(t-1)!} \\
&= \frac{s!(t-1-k)!}{(s-k)!(t-1)!} \\
&= \frac{s!k!(t-1-k)!}{k!(s-k)!(t-1)!} \\
&= \binom{s}{k} \binom{t-1}{k}^{-1}.
\end{aligned}$$

Usando probabilidades totales obtenemos que si $x \in \text{OPT}$, entonces:

$$\begin{aligned}
\mathbb{P}(x \in A \mid S = s) &= \sum_{t=s+1}^n \mathbb{P}(x \in A \mid S = s, x = x_t) \frac{1}{n} \\
&\geq \frac{1}{n} \sum_{t=s+1}^n \binom{s}{k} \binom{t-1}{k}^{-1} \\
&= \frac{1}{k-1} \left(\frac{s}{n} - \binom{s}{k} \binom{n}{k}^{-1} \right).
\end{aligned}$$

Donde la última igualdad se debe al Lema 5.3. Ahora consideremos $s \leq k$ fijo. Si $x \in \text{OPT}$, basta que $t(x) = s+1$ para que x sea aceptado. Por lo tanto:

$$\mathbb{P}(x \in A \mid S = s) \geq \mathbb{P}(x = x_{s+1}) = \frac{1}{n} \geq \frac{s}{n(k-1)}$$

Finalmente, como $S \sim \text{Binomial}(n, p)$, nos queda que para $x \in \text{OPT}$:

$$\begin{aligned}
\mathbb{P}(x \in A) &= \sum_{s=0}^n \mathbb{P}(x \in A | S = s) \binom{n}{s} p^s (1-p)^{n-s} \\
&\geq \sum_{s=0}^{k-1} \frac{s}{n(k-1)} \binom{n}{s} p^s (1-p)^{n-s} + \sum_{s=k}^n \frac{1}{k-1} \left(\frac{s}{n} - \frac{\binom{s}{k}}{\binom{n}{k}} \right) \binom{n}{s} p^s (1-p)^{n-s} \\
&= \frac{1}{k-1} \left(\frac{1}{n} \sum_{s=0}^n s \binom{n}{s} p^s (1-p)^{n-s} - \sum_{s=k}^n \frac{\binom{s}{k}}{\binom{n}{k}} \binom{n}{s} p^s (1-p)^{n-s} \right) \\
&= \frac{1}{k-1} \left(\frac{\mathbb{E}(S)}{n} - \sum_{s=k}^n \binom{n-k}{s-k} p^s (1-p)^{n-s} \right) \\
&= \frac{1}{k-1} \left(p - \sum_{s=0}^{n-k} \binom{n-k}{s} p^{s+k} (1-p)^{n-k-s} \right) \\
&= \frac{1}{k-1} (p - p^k) .
\end{aligned}$$

Ahora para optimizar p basta derivar e igualar a 0. Al hacerlo encontramos que el p óptimo es $\sqrt[k-1]{1/k}$, reemplazando este valor de p obtenemos:

$$\mathbb{P}(x \in A) \geq \frac{\sqrt[k-1]{1/k} (1 - \frac{1}{k})}{(k-1)} = \sqrt[k-1]{k^{-k}}.$$

□

Notemos que esta garantía efectivamente supera el anterior ke de [19]. Debemos probar que $ke \geq \sqrt[k-1]{k^k}$, o lo que es igual, $e \geq \sqrt[k-1]{k}$. Al calcular la derivada de la función $f(k) = \sqrt[k-1]{k} = \exp\left(\frac{1}{k-1} \ln(k)\right)$ obtenemos:

$$f'(k) = \sqrt[k-1]{k} \left(\frac{1}{k(k-1)} - \frac{\ln(k)}{(k-1)^2} \right) \leq \sqrt[k-1]{k} \left(\frac{1 - \ln(k)}{(k-1)^2} \right).$$

Luego, para $k \geq 3$, f decrece. Por lo tanto, basta ver que $f(2) = 2$ y $f(3) = \sqrt{3}$ son ambos menores que e . Por lo tanto la competitividad obtenida es mejor que ke para todo $k \geq 2$.

Por último notemos que si $k = 1$, estamos en presencia de una matroide de partición unitaria. En efecto, primero notemos que SPG no hay columnas nulas (si hay, simplemente las rechazamos). Si definimos por cada fila f un conjunto $P_f = \{x \in E : x(f) \neq 0\}$ tenemos una partición de E y es claro que al tomar un vector de cada parte se obtiene un conjunto l.i. Además, si tomamos dos vectores de una misma parte, el conjunto obtenido es dependiente ya que los dos vectores serán múltiplos uno del otro. Por lo tanto, para $k = 1$ definimos la partición ya mencionada y aplicamos la solución del problema clásico en cada uno obteniendo un algoritmo fuertemente e -competitivo.

Capítulo 6

MSP en Matroides Laminares

En este capítulo abordamos el MSP en matroides laminares, en lo que sigue llamaremos a este problema LMSP. Recordemos que definimos familias laminares y matroides laminares en el Capítulo 1 (ver Definiciones 1.20 y 1.21). Durante este capítulo suponemos que tenemos una matroide laminar $\mathcal{M} = (E, \mathcal{I})$ asociada a una familia laminar \mathcal{L} y a un conjunto de restricciones $\{b_L\}_{L \in \mathcal{L}}$. SPG asumimos que $E \in \mathcal{L}$, ya que si E no estuviera en \mathcal{L} , puede agregarse junto con $b_E = |E|$ sin cambiar la familia \mathcal{I} de conjuntos independientes. Además, para nuestro problema suponemos que $b_L \geq 1$, $\forall L \in \mathcal{L}$. Si no fuera así, podemos eliminar todos los elementos que pertenezcan a algún L con $b_L = 0$ (loops) antes de empezar.

Los primeros en probar que este problema admite un algoritmo $O(1)$ -competitivo fueron Im y Wang en 2011 [9]. En su trabajo presentan un nuevo algoritmo para este problema y prueban que es $16000/3$ -competitivo. En 2013, Jaillet et al. [10] mejoran por mucho el resultado obtenido en [9]. En su paper, Jaillet et al. diseñan dos nuevos algoritmos, y prueban que el primero es $27e/2$ -competitivo mientras que el segundo alcanza una competitividad de $3\sqrt{3}e$. Nos referiremos a estos algoritmos como Algoritmo JSZ-1 y Algoritmo JSZ-2 respectivamente. En 2016, Ma et al. [16] logran una nueva mejora, un algoritmo fuertemente 9,6-competitivo.

Presentamos aquí dos algoritmos inspirados en aquellos desarrollados en [10]. Con leves modificaciones y aplicando las técnicas que hemos venido usando en capítulos anteriores, logramos mejorar la competitividad, obteniendo un algoritmo fuertemente 8-competitivo y otro fuertemente $3\sqrt{3}$ -competitivo. Al igual que en [10], usamos la siguiente propiedad (la dejamos sin demostrar, pero no es difícil de probar por inducción):

Proposición 6.1. *Sea \mathcal{L} una familia laminar sobre E . Existe un orden de E , digamos $E = \{x^1, \dots, x^n\}$, tal que que todo conjunto $L \in \mathcal{L}$ es de la forma $\{x^k : i \leq k \leq j\}$.*

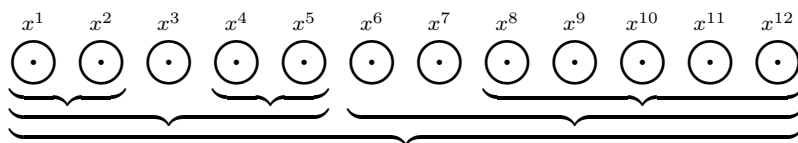


Figura 6.1: Es posible ordenar E de modo que cada $L \in \mathcal{L}$ es un intervalo.

6.1. Algoritmo δ -competitivo para el LMSP

En esta ocasión suponemos que tenemos una matroide laminar definida por una familia laminar \mathcal{L} y un conjunto de restricciones $\{b_L : L \in \mathcal{L}\}$. Asumimos que conocemos un orden de E como el de la Proposición 6.1. Denotamos la posición de un elemento x en este orden con un superíndice (mantenemos así los subíndices para indicar el orden en que los elementos se revelan al algoritmo). Además en ocasiones usamos $p(x)$ para indicar la posición de x en el orden de la familia laminar. El algoritmo se detalla a continuación:

Algoritmo 9

- 1: $s \leftarrow \lceil \frac{n}{2} \rceil$;
 - 2: $A \leftarrow \phi$;
 - 3: Calculamos $\text{OPT}_s = \{x^{j_1}, \dots, x^{j_m}\}$ con j_k creciente;
 - 4: $j_0 \leftarrow 0, j_{m+1} \leftarrow n + 1$;
 - 5: **Para** $k = 0 \dots m$ **hacer**
 - 6: $B_k \leftarrow \{x^{j_k+1}, x^{j_k+2}, \dots, x^{j_{k+1}-1}\}$;
 - 7: $X \leftarrow \text{Bernoulli}(1/2)$;
 - 8: **Para** $t = s + 1 \dots n$ **hacer**
 - 9: Calculamos OPT_t ;
 - 10: Calculamos k_t tal que $x_t \in B_{k_t}$;
 - 11: **Si** $x_t \in \text{OPT}_t, k_t \equiv_2 X$ **y** $|B_{k_t} \cap A| = 0$ **entonces**
 - 12: $A \leftarrow A + x_t$;
 - 13: **Devolver** A ;
-

Como todos los algoritmos que hemos visto, el Algoritmo 9 comienza tomando una muestra. Esta vez fijamos de antemano el tamaño de la muestra como $\lceil n/2 \rceil$. Dicha elección tendrá justificación más adelante, cuando reaparezcan expresiones que vimos en el Capítulo 4 durante el análisis del Algoritmo 6. Teniendo la muestra, calculamos su óptimo OPT_s y marcamos las posiciones de sus elementos para definir una partición en el resto de E como ilustra el siguiente ejemplo:

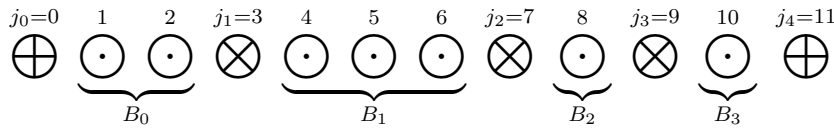


Figura 6.2: Aquí $n = 10$, marcamos los elementos de OPT_s con una equis, de modo que $\text{OPT}_s = \{x^3, x^7, x^9\}$. Esto define la partición $\{B_0, B_1, B_2, B_3\}$ con $B_0 = \{x^1, x^2\}$, $B_1 = \{x^4, x^5, x^6\}$, $B_2 = \{x^8\}$ y $B_3 = \{x^{10}\}$.

La v.a. X es como una moneda que se usa para elegir entre los conjuntos B_k , si descartamos aquellos con índice par o aquellos de índice impar. Después de la muestra, cada vez que se revela un elemento $x_t \in B_{k_t}$, éste es aceptado si y solo si se cumple que:

1. x_t mejora lo visto ($x_t \in \text{OPT}_t$).
2. k_t tiene la misma paridad de X ($k_t \equiv_2 X$).
3. Aún no hemos elegido ningún elemento en B_{k_t} ($|B_{k_t} \cap A| = 0$).

Este algoritmo es casi idéntico al Algoritmo JSZ-1, hay una única diferencia además del tamaño de la muestra. Aquí tomamos de cada B_k no descartado, el primer elemento que mejore lo que hemos visto. En cambio, en el Algoritmo JSZ-1 se busca el elemento más pesado de cada B_k no descartado usando el algoritmo del problema clásico (Algoritmo 2).

En [10], Lema 3, demuestran que el Algoritmo JSZ-1 entrega un conjunto independiente. Su demostración también aplica a nuestro algoritmo y la incluimos aquí, prácticamente igual salvo notación.

Lema 6.2. *El conjunto A devuelto por el Algoritmo 9 es independiente.*

DEMOSTRACIÓN. Sea $L \in \mathcal{L}$, recordemos que L es un intervalo de elementos consecutivos de E . Debemos probar que $|L \cap A| \leq b_L$. Como supusimos $b_L \geq 1$, si $|L \cap A| = 1$ estamos listos. Supongamos $|L \cap A| \geq 2$, vamos a mostrar que entre dos elementos de $L \cap A$ siempre hay al menos dos elementos de $L \cap \text{OPT}_s$, lo cual implica que $|L \cap A| \leq |L \cap \text{OPT}_s| \leq b_L$.

Sean $x^p, x^q \in L \cap A$ con $p < q$. Sean k, k' tales que $x^p \in B_k$ y $x^q \in B_{k'}$. Como el algoritmo elige a lo más un elemento por cada intervalo B_k , y no elige elementos de intervalos consecutivos, podemos afirmar que B_{k+1} se encuentra contenido entre x^p y x^q . Luego, los dos elementos de OPT_s que delimitan B_{k+1} están entre x^p y x^q , y pertenecen a $L \cap \text{OPT}_s$. \square

En [10], llaman a un elemento $x \in \text{OPT}$ solitario si cae fuera de la muestra en un conjunto B_k tal que $B_k \cap \text{OPT} = \{x\}$ y $X \equiv_2 k$. Para demostrar que el Algoritmo JSZ-1 es $27e/2$ -competitivo primero prueban que cualquier elemento de OPT tiene probabilidad $2/27$ de ser solitario. Si x es solitario, de su intervalo se elige con probabilidad $1/e$ un elemento que pesa lo mismo o más que él. Luego, por cada $x \in \text{OPT}$ su algoritmo elige con probabilidad $2/27e$ un elemento y con $w(y) \geq w(x)$.

Aunque los algoritmos son, como ya dijimos, muy similares, nuestro análisis toma un rumbo distinto. Antes de ver la competitividad de nuestro algoritmo necesitamos un par de lemas.

Lema 6.3. $\forall x \in E$, $\{t : x \in \text{OPT}_t\}$, el conjunto de tiempos en los cuales x está en el óptimo de lo visto, es o bien vacío o bien un intervalo de la forma $\{t(x), \dots, t(x) + j\}$.

DEMOSTRACIÓN. Este lema es prácticamente un corolario del Lema 1.16. En primer lugar notemos que x no puede estar en el óptimo de lo visto antes de ser visto (antes de $t(x)$). Ahora, sea $T > t$ tal que $x \in \text{OPT}_T$ y sea $i \in \{t, \dots, T\}$. Tomemos como matroide $\mathcal{M}|_{E_T}$, por el Lema 1.16, $\text{OPT}_T \cap E_i \subseteq \text{OPT}_i$. Luego $x \in \text{OPT}_i$, de donde se concluye. \square

Antes de continuar introducimos un poco de notación, para cada $t \in [n]$ anotamos:

- $I_t(x) = x^I$ con $I = \text{máx}(\{i : i < p(x) \wedge x^i \in \text{OPT}_t\} \cup \{0\})$.
- $D_t(x) = x^D$ con $D = \text{mín}(\{d : d > p(x) \wedge x^d \in \text{OPT}_t\} \cup \{n + 1\})$.
- $J_t(x) = \{y \in E : p(I_t(x)) < p(y) < p(D_t(x))\}$.

En palabras, $I_t(x)$ es el último elemento de OPT_t que está a la izquierda de x según el

orden de la familia laminar y $D_t(x)$ el primero que aparece a la derecha de x (si alguno de estos elementos no existe, consideramos elementos fantasmas con posición 0 o $n + 1$ respectivamente). El conjunto $J_t(x)$ contiene a los elementos con posiciones entre $I_t(x)$ y $D_t(x)$.

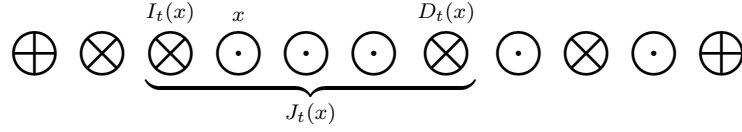


Figura 6.3: Aquí $n = 9$, $\text{OPT}_t = \{x^1, x^2, x^6, x^8\}$ y $p(x) = 3$. En este caso $I_t(x) = x^2$, $D_t(x) = x^6$ y $J_t(x) = \{x^2, x^3, x^4, x^5, x^6\}$.

Lema 6.4. *Sea $t > s$, si $x_i \notin \{I_i(x_t), D_i(x_t)\}$, $\forall i \in \{s + 1, \dots, t - 1\}$, entonces ningún elemento de B_{k_t} es elegido antes del tiempo t .*

DEMOSTRACIÓN. Probemos primero que con la hipótesis del lema, $J_{i-1}(x_t) \subseteq J_i(x_t)$ para todo $i \in \{s + 1, \dots, t - 1\}$. Lo anterior equivale a probar que $p(I_{i-1}(x_t)) \geq p(I_i(x_t))$ y $p(D_{i-1}(x_t)) \leq p(D_i(x_t))$. Comprobemos la primera desigualdad (la segunda es análoga). Por la hipótesis del lema, $I_i(x_t) \neq x_i$, eso implica que $I_i(x_t) \in E_{i-1}$ (por Lema 6.3). Además $I_i(x_t) \in \text{OPT}_i$, aplicando el lema anterior tenemos que $I_i(x_t) \in \text{OPT}_{i-1}$. Por lo tanto, $p(I_{i-1}(x_t)) \geq p(I_i(x_t))$. Inductivamente tenemos que $B_{k_t} = J_s(x_t) \subseteq J_i(x_t)$, $\forall i \in \{s + 1, \dots, t - 1\}$ lo que se traduce en que antes del tiempo t nunca entró en A ningún elemento de B_{k_t} . \square

Ahora estudiamos la competitividad de nuestro algoritmo y quedará en evidencia la razón por la cual tomamos $s = \lceil n/2 \rceil$.

Teorema 6.5. *El Algoritmo 9 es fuertemente 8-competitivo.*

DEMOSTRACIÓN. Sea $x_t \in \text{OPT}$ con $t > s$ y consideremos el conjunto A antes de la llegada de x_t . Tenemos que:

$$\mathbb{P}(x_t \text{ sea aceptado}) = \mathbb{P}(k_t \equiv_2 X, |B_{k_t} \cap A| = 0) = \frac{\mathbb{P}(|B_{k_t} \cap A| = 0)}{2}.$$

Gracias al lema anterior podemos acotar esta probabilidad:

$$\begin{aligned} \mathbb{P}(|B_{k_t} \cap A| = 0) &\geq \mathbb{P}\left(\bigcap_{i=s+1}^{t-1} x_i \notin \{I_i(x_t), D_i(x_t)\}\right) \\ &\geq \prod_{i=s+1}^{t-1} \frac{i-2}{i}. \end{aligned}$$

La última desigualdad sigue la misma lógica que hemos ocupado antes. En cada tiempo i hay i opciones para x_i y de ellas solo prohibimos 2. Esto es exactamente lo que teníamos en la demostración del Teorema 4.2 salvo porque tenemos un 2 extra en la garantía. Repitiendo los cálculos de dicha demostración obtenemos que si $x \in \text{OPT}$ entonces $\mathbb{P}(x \in A) \geq 1/8$. \square

6.2. Algoritmo $3\sqrt{3}$ -competitivo para el LMSP

Resumiendo parte de lo que vimos en la sección anterior, tanto el Algoritmo JSZ-1 presentado en [10] como nuestro Algoritmo 9 hacen lo siguiente: toman una muestra, definen una partición en base a la muestra, botan la mitad de los conjuntos de la partición y de los conjuntos que quedan eligen un elemento. La diferencia es que el Algoritmo JSZ-1 trata de elegir el elemento más pesado en cada conjunto mientras que el Algoritmo 9 toma en cada conjunto el primer elemento que mejore lo visto.

La forma en que particionamos los elementos en el Algoritmo JSZ-1 no nos permite tomar un elemento de cada conjunto de la partición, de hecho nos fuerza a descartar la mitad de los conjuntos en la partición. En el Algoritmo JSZ-2 se define una nueva partición a partir de la muestra que sí permite tomar un elemento de cada parte. A continuación explicamos esta nueva partición definida en [10].

Para cualquier independiente $I \in \mathcal{I}$, definimos la partición $\mathcal{P}(I)$ de E como sigue. Si $I = \emptyset$, fijamos $\mathcal{P}(I) = \{E\}$. Si no, $\mathcal{P}(I)$ contiene un conjunto $P(y)$ por cada elemento $y \in I$, es decir, $\mathcal{P}(I) = \{P(y) : y \in I\}$. Para determinar la partición $\mathcal{P}(I)$, vamos a especificar a qué conjunto $P(y)$ pertenece cada $x^i \in E$. Sea $L \in \mathcal{L}$ el conjunto más pequeño que contiene a x^i e interseca a I . Tal conjunto existe debido a que suponimos $E \in \mathcal{L}$. Si existe algún $j \leq i$ tal que $y^j \in L \cap I$ tomamos el mayor y ponemos x^i en $P(y^j)$. Si no, tomamos el menor $j > i$ tal que $y^j \in L \cap I$ y colocamos x^i en $P(y^j)$.

En la siguiente figura se ilustra la definición anterior:

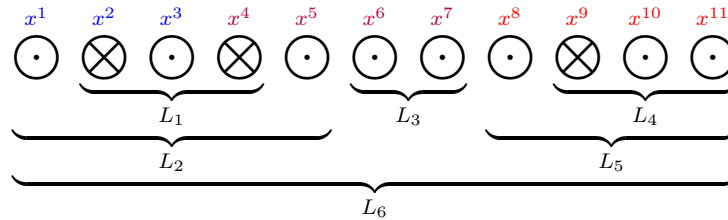


Figura 6.4: aquí $\mathcal{L} = \{L_1, \dots, L_6\}$, tomamos como conjunto independiente $I = \{x^2, x^4, x^9\}$ (elementos marcados con \times). Los colores representan la partición: $P(x^2)$ queda formado por los elementos azules, $P(x^4)$ por los morados y $P(x^9)$ por los rojos.

Sea $I \in \mathcal{I}$, para cada elemento $x \in E$ denotamos por $L_I(x)$ al menor conjunto $L \in \mathcal{L}$ que contiene a x e interseca a I . Cuando no haya ambigüedad sobre el conjunto I anotaremos simplemente $L(x)$. Por ejemplo, en la figura anterior $L(x^3) = L_1$, $L(x^5) = L_2$ y $L(x_6) = L_6$ ya que L_3 no interseca a I .

En la sección anterior, la diferencia entre el Algoritmo JSZ-1 y el Algoritmo 9 era que en vez de buscar el máximo en cada parte, tomábamos el primer elemento que mejorara lo visto en cada parte. Hacemos la misma modificación al Algoritmo JSZ-2 para obtener nuestro siguiente algoritmo:

Algoritmo 10

- 1: $s \leftarrow \text{Binomial}(n, 1/\sqrt{3})$;
 - 2: $A \leftarrow \phi$;
 - 3: Calculamos $\mathcal{P}(\text{OPT}_s)$;
 - 4: **Para** $t = s + 1 \dots n$ **hacer**
 - 5: Calculamos OPT_t ;
 - 6: Calculamos $P(x_t) \in \mathcal{P}(\text{OPT}_s)$ tal que $x_t \in P(x_t)$;
 - 7: **Si** $x_t \in \text{OPT}_t$ y $|P(x_t) \cap A| = 0$ **entonces**
 - 8: $A \leftarrow A + x_t$;
 - 9: **Devolver** A ;
-

En esta oportunidad la forma de mostrar es la misma que la usada en el Algoritmo JSZ-2. Justificamos esta elección más adelante cuando estudiamos la competitividad.

En [10] demuestran que el Algoritmo JSZ-2 devuelve un conjunto independiente probando dos lemas (Lemas 5 y 6). Estos lemas también prueban que el Algoritmo 10 devuelve un conjunto independiente y los incluimos a continuación. El primer lema corresponde a una propiedad de los conjuntos $P \in \mathcal{P}(I)$ que puede apreciarse en la Figura 6.4:

Lema 6.6. *Sea $I \in \mathcal{I}$ y $x^i \in I$, existen $j \leq i \leq k$ tales que $P(x^i) = \{x^j, \dots, x^k\}$.*

DEMOSTRACIÓN. Es directo que $x^i \in P(x^i)$. Debemos probar que si $x^p \in P(x^i)$ entonces todos los elementos entre x^p y x^i están en $P(x^i)$. Distinguiamos dos casos:

- $p < i$: sea $q \in \{p, \dots, i\}$, queremos probar que $x^q \in P(x^i)$. Como $x^p \in P(x^i)$, no existe $l \leq p$ tal que $x^l \in L(x^p)$. Tampoco puede existir $p < m < i$ tal que $x^m \in L(x^p) \cap I$. Recordemos que $L(x^p)$ es una secuencia de elementos consecutivos, como $x^i, x^p \in L(x^p)$ también x^q está en $L(x^p)$. Por definición $L(x^q) \subseteq L(x^p)$, así que tampoco existe $1 < l < i$ tal que $x^l \in L(x^q)$. Por lo tanto, $x^q \in P(x^i)$.
- $i < p$: sea $q \in \{i, \dots, p\}$, nuevamente queremos ver que $x^q \in P(x^i)$. Como antes $L(x^q) \subseteq L(x^p)$. Además, si no son iguales, entonces $x^p \notin L(x^q)$, lo que implica $x^m \notin L(x^q)$ para $m \geq p$. Dado que $x^p \in P(x^i)$, los elementos en $\{x^{i+1}, \dots, x^p\}$ no están en I . Como $L(x^q) \cap I \neq \phi$, $x^i \in L(x^q)$ y no hay elementos de I en $\{x^{i+1}, \dots, x^q\}$. Por lo tanto, $x^q \in P(x^i)$.

□

Lema 6.7. *El conjunto A devuelto por el Algoritmo 10 es independiente.*

DEMOSTRACIÓN. Sea $L \in \mathcal{L}$, veamos que $|L \cap A| \leq b_L$. Si $\text{OPT}_s \cap L = \phi$, entonces todos los elementos de L se asignan a un mismo $P \in \mathcal{P}(\text{OPT}_s)$. Luego, $|L \cap A| \leq |P \cap A| \leq 1 \leq b_L$. Supongamos entonces $\text{OPT}_s \cap L \neq \phi$. Notemos que en este caso cada elemento de L es asignado a un conjunto $P(x)$ con $x \in \text{OPT}_s \cap L$, es decir,

$$L \subseteq \bigcup_{x \in \text{OPT}_s \cap L} P(x).$$

Por lo tanto:

$$|A \cap L| \leq \left| A \cap \bigcup_{x \in \text{OPT}_s \cap L} P(x) \right| = \left| \bigcup_{x \in \text{OPT}_s \cap L} A \cap P(x) \right| \leq |\text{OPT}_s \cap L| \leq b_L.$$

□

Antes de estudiar la competitividad notemos algo curioso. El Algoritmo JSZ-2 es $3\sqrt{3}e$ -competitivo, probaremos que el Algoritmo 10 es $3\sqrt{3}$ -competitivo. En el Algoritmo JSZ-2 se usa el algoritmo para el problema clásico en cada pedazo de la partición y de ahí sale el factor e de la competitividad. En nuestro algoritmo no aplicamos la solución al problema clásico y no obtenemos el factor e en la competitividad. Por todo lo anterior sería esperable que los análisis de competitividad fueran parecidos en ambos casos, pero no es así. El análisis para el Algoritmo 10 es de hecho parecido al análisis del Algoritmo 9. Antes de mostrar la competitividad, necesitamos un poco más de notación y un par de lemas.

Lema 6.8. Sean $I, J \in \mathcal{I}$ con $I \subseteq J$ entonces la partición $\mathcal{P}(J)$ es más fina que $\mathcal{P}(I)$.

DEMOSTRACIÓN. Por inducción basta ver el caso en que $J = I + y$. Nos enfocamos en $\mathcal{P}(I)$, recordemos que cada $P \in \mathcal{P}(I)$ es un intervalo de elementos consecutivos y contiene al único elemento $a \in I$ que lo define (Lema 6.6). Sean $x, z \in I$ los primeros elementos que aparecen a la izquierda y a la derecha de y respectivamente, considere la siguiente ilustración:

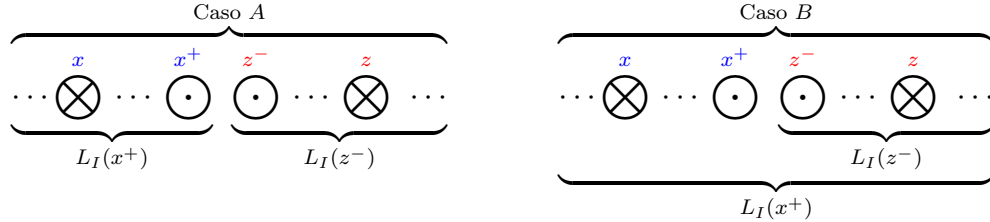


Figura 6.5: partición inducida por I . x^+ representa al último elemento de izquierda a derecha que pertenece a $P(x)$ y z^- al primer elemento de izquierda a derecha que pertenece a $P(z)$. Vamos a probar que SPG estamos en uno de estos casos.

Lo que queremos probar puede pensarse gráficamente del siguiente modo. El conjunto I induce una partición, que representamos con colores. Cuando agregamos y y calculamos la nueva partición se agrega un nuevo color, digamos lila. Lo que tenemos que probar es que todos los elementos de color lila tenían el mismo color en $\mathcal{P}(I)$. Antes de continuar notemos que como $I \subseteq J$ si L intersecta a I , también intersecta a J . Por lo tanto, para todo $w \in E$ tenemos $L_J(w) \subseteq L_I(w)$. El resto de la demostración se organiza en tres partes. Primero nos ponemos en el caso A, después en el caso B y finalmente mostramos que SPG estamos en uno de esos casos.

Caso A: si y es rojo inicialmente, queremos mostrar que ningún elemento azul se vuelve lila. Por el Lema 6.6 basta ver que x^+ siga azul. Para que x^+ se vuelva lila, es necesario que $L_J(x^+)$ contenga a x^+ y a y , lo que implica $z^- \in L_J(x^+) \subseteq L_I(x^+)$, pero esto no ocurre en el

caso *A*. Análogamente, si y es azul inicialmente, hay que probar que z^- sigue rojo en $\mathcal{P}(J)$. Si no, al igual que antes queda $x^+ \in L_J(z^-) \subseteq L_I(z^-)$ que nuevamente no ocurre en A .

Caso *B*: Si y es inicialmente azul se sigue exactamente igual que en el caso *A*. Si y es inicialmente rojo, queremos ver que x^+ siga azul. Para que cambiara de color sería necesario que, $L_J(x^+)$ contenga a x^+ y z^- , y que no contenga a x . Entonces $L_J(x^+)$ intersecta a $L_I(z^-)$ sin estar contenido en él. Como \mathcal{L} es laminar, esto implica $L_I(z^-) \subseteq L_J(x^+) \subseteq L_I(x^+)$, en particular $x^+, z \in L_J(x^+)$ lo que implicaría que x^+ era rojo para empezar, lo que es una contradicción.

Ahora veamos que SPG podemos asumir que estamos en el caso *A* o en el *B*. Para comenzar, notemos que los siguiente casos son evidentes:

- Si x no existe, los elementos lila eran todos rojos antes.
- Si z no existe, los elementos lila eran todos azules antes.
- Si $x^+ = x$, los elementos lila eran todos rojos antes.
- Si $z^- = z$, los elementos lila eran todos azules antes.

SPG podemos suponer entonces que x, x^+, z^- y z existen y son todos distintos.

Como $x^+ \in P(x)$, $x^+, x \in L_I(x^+)$, del mismo modo $z^-, z \in L_I(z^-)$. Veamos que $x^+ \notin L_I(z^-)$, por contradicción:

$$\begin{aligned} \text{Si } x^+ \in L_I(z^-) &\Rightarrow L_I(x^+) \subseteq L_I(z^-) \\ &\Rightarrow x \in L_I(z^-) \\ &\Rightarrow z^- \in P(x) \rightarrow \leftarrow \end{aligned}$$

Finalmente, $L_I(x^+)$ debe contener a x y x^+ , dado que \mathcal{L} es laminar, solo le quedan dos alternativas: o es disjunto de $L_I(z^-)$ (caso *A*) o lo contiene (caso *B*). □

Ahora algo más de notación, para abreviar usamos \mathcal{P}_i para referirnos a $\mathcal{P}(\text{OPT}_i)$. Anotamos $P_i(x)$ para denotar al conjunto de \mathcal{P}_i que contiene a x . Además, para $i, t \in [n]$ definimos:

- $v_i(t)$ (vecino en tiempo i de x_t): es el elemento de OPT_i tal que $x_t \in P_i(v_i(t))$.
- $v_i^-(t)$ (vecino izquierdo en tiempo i de x_t): es el elemento de OPT_i que se encuentra a la izquierda de $v_i(t)$.
- $v_i^+(t)$ (vecino derecho en tiempo i de x_t): es el elemento de OPT_i que se encuentra a la derecha de $v_i(t)$.

Lema 6.9. *Sea $t > s$, si $x_i \notin \{v_i^-(t), v_i(t), v_i^+(t)\} \forall i \in \{s+1, \dots, t-1\}$, entonces ningún elemento de $P(x_t)$ es elegido antes del tiempo t .*

DEMOSTRACIÓN. Este lema es similar al Lema 6.4, y la demostración sigue el mismo esquema. Primero vamos a demostrar que bajo la hipótesis del lema la familia de conjuntos $P_i(x_t)$ es creciente en i . Después concluimos que ningún elemento de $P(x_t) = P_s(x_t)$ pudo ser elegido antes de que x_t se revelara.

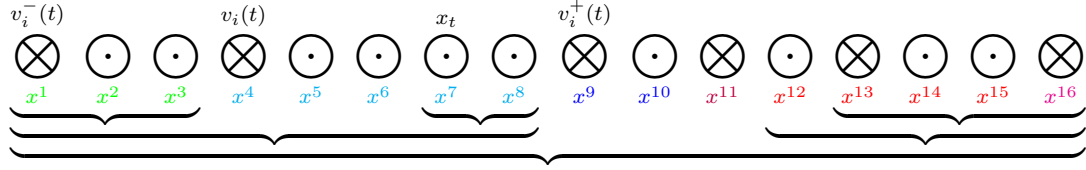


Figura 6.6: aquí se ilustran las definiciones anteriores. Al igual que en la figura anterior las llaves muestran los conjuntos de la familia \mathcal{L} . OPT_i está formado por los elementos marcados con \times . Los colores representan la partición.

Veamos que para todo $i \in \{s+1, \dots, t-1\}$, $P_{i-1}(x_t) \subseteq P_i(x_t)$. Tenemos que $v_i^-(t)$, $v_i(t)$ y $v_i^+(t)$ están en OPT_i , dado que x_i no es ninguno de ellos, $v_i^-(t), v_i(t), v_i^+(t) \in \text{OPT}_{i-1}$ (por el Lema 6.3). Si OPT_{i-1} tiene un elemento y que no esté en OPT_i y éste cae fuera del intervalo definido por $v_i^-(t)$ y $v_i^+(t)$, entonces $P_{i-1}(x_t) = P_i(x_t)$. Si y está entre $v_i^-(t)$ y $v_i^+(t)$, por la demostración del Lema 6,8 se tiene $P_{i-1}(x_t) \subseteq P_i(x_t)$. Por lo tanto, $P_i(x_t)$ es creciente en i .

Finalmente, supongamos que el algoritmo elige a $x_i \in P(x_t) = P_s(x_t) \subseteq P_{i-1}(x_t) \subseteq P_i(x_t)$ con $i < t$. Es claro que esta elección subdivide el conjunto $P_{i-1}(x_t)$ en dos partes no vacías, luego $P_i(x_t)$ queda estrictamente contenido en $P_{i-1}(x_t) \rightarrow \leftarrow$

□

Teorema 6.10. *El Algoritmo 10 es fuertemente $3\sqrt{3}$ -competitivo.*

DEMOSTRACIÓN. Sea $x_t \in \text{OPT}$ con $t > s$ y consideremos el conjunto A antes de la llegada de x_t . Tenemos que:

$$\mathbb{P}(x_t \text{ sea aceptado}) = \mathbb{P}(|P(x_t) \cap A| = 0).$$

Gracias al lema anterior podemos acotar esta probabilidad:

$$\begin{aligned} \mathbb{P}(|P(x_t) \cap A| = 0) &\geq \mathbb{P}\left(\bigcap_{i=s+1}^{t-1} x_i \notin \{v_i^-(t), v_i(t), v_i^+(t)\}\right) \\ &\geq \prod_{i=s+1}^{t-1} \frac{i-3}{i}. \end{aligned}$$

Nuevamente, para la última desigualdad, en cada tiempo i hay i opciones para x_i y de ellas solo prohibimos 3. Esto es lo mismo que obtuvimos en el Teorema 5.4 del capítulo anterior para $k = 3$. Esto justifica la forma de muestrear y nos dice que la probabilidad de aceptar cualquier elemento de OPT es al menos $3\sqrt{3}$. □

Para terminar notemos que los algoritmos que presentamos en este capítulo tampoco necesitan conocer los pesos de los elementos. Para calcular OPT_t nos basta con poder rankear los elementos vistos.

Capítulo 7

Síntesis de los Algoritmos Vistos, Obtención de Bases en el MSP y Algoritmo Cardinal $O(1)$ -competitivo para el MSP.

Este capítulo tiene tres resultados independientes. Primero estudiamos los elementos comunes de los algoritmos online presentados en los cuatro capítulos anteriores y describimos un procedimiento general que llamamos Algoritmo Glotón Mañoso. El segundo resultado muestra como modificar un algoritmo online para el MSP de manera que el conjunto devuelto sea una base. El tercer resultado es un algoritmo cardinal $O(1)$ -competitivo para el MSP en una matroide arbitraria.

7.1. Algoritmo Glotón Mañoso

Si estudiamos las demostraciones de los teoremas que hemos visto concernientes a la competitividad de nuestros algoritmos, podemos ver que hay argumentos que se repiten. Luego, es natural preguntarse si los algoritmos tienen algo en común y la respuesta es que sí. A continuación presentamos un algoritmo genérico que resume la estructura básica común a los Algoritmos 2, 3, 4, 5, 6, 8, 10.

Algoritmo 11 Glotón Mañoso

- 1: Escogemos s tamaño de muestra;
 - 2: Inicializamos la solución: $A \leftarrow \phi$;
 - 3: **Para** $t = s + 1 \dots n$ **hacer**
 - 4: Calculamos OPT_t ;
 - 5: **Si** $x_t \in \text{OPT}_t$ y se cumple una condición extra $q(t)$ **entonces**
 - 6: $A \leftarrow A + x_t$;
 - 7: **Devolver** A ;
-

Además de la elección de s (que siempre tiene como valor esperado una fracción de n), lo que cambia de una matroide a otra es la condición extra $q(t)$.

Sea $t > s$ y consideremos el conjunto A antes de que veamos al elemento x_t . En cada caso que vimos $q(t)$ tenía las siguientes propiedades:

1. Si $A \in \mathcal{I}$ y $q(t)$ se cumple, entonces $A + x_t \in \mathcal{I}$.
2. Existe $k \in \mathbb{N}$ tal que para cada $i \in \{s + 1, \dots, t - 1\}$, dado E_i fijo, existe un conjunto $F_i(x_t)$ con $|F_i(x_t)| \leq k$ tal que:

$$\bigcap_{i=s+1}^{t-1} \{x_i \notin F_i(x_t)\} \Rightarrow q(t).$$

La primera propiedad le da el nombre al algoritmo. Los algoritmos glotones llevan su nombre porque cada vez que pueden agregar un elemento manteniendo factibilidad, lo agregan. Aquí en cambio, $q(t)$ implica factibilidad, pero puede ser más restrictivo. De este modo existe la posibilidad de que el algoritmo rechace elementos que podría aceptar, he ahí la razón del nombre. La condición $q(t)$ así como el k varía entre una matroide y otra, siendo lo que diferencia los algoritmos que hemos presentado hasta ahora.

La segunda propiedad es la clave para el análisis de la competitividad, como ya hemos hecho los cálculos un par de veces no los repetimos aquí. Repasemos solo las ideas principales. Queremos acotar la probabilidad de que un elemento $x \in \text{OPT}$ sea aceptado. El Lema 1.16 nos dice que si $x_t \in \text{OPT}$ entonces $x_t \in \text{OPT}_t$. Suponemos que $x_t = x \in \text{OPT}$, por la observación anterior $\mathbb{P}(x_t \text{ sea aceptado}) = \mathbb{P}(q(t))$. Usamos la segunda propiedad para acotar la probabilidad de $q(t)$. Ahora pensamos que elegimos los primeros $t - 1$ elementos del siguiente modo:

1. Elegimos E_{t-1} sacando al azar $t - 1$ elementos de $E - x$.
2. Elegimos el orden de estos elementos de atrás para adelante al azar. Como si tuviéramos una urna con los elementos de E_{t-1} y los fuéramos sacando sin reposición y los colocáramos desde el último en llegar hasta el primero.

De este modo, para que $q(t)$ se cumpla, cada vez que sacamos un elemento x_i de la urna tenemos $i - k$ casos favorables dentro de i opciones. Finalmente, se usan probabilidades totales para calcular la probabilidad de que x sea aceptado condicionando en su tiempo de llegada.

7.2. Obtener Bases en el MSP

Pensemos primero en el problema clásico de la secretaria como problema de contratación. Estamos entrevistando candidatos para un puesto y nuestro objetivo es elegir al mejor, pero ¿qué ocurre si no hemos elegido a nadie y llegamos a entrevistar al último candidato y descubrimos que no es el mejor? Si nuestro único objetivo era elegir al mejor, podríamos rendirnos y dejar el puesto vacante sin elegir a nadie. Pero es probable que en muchas ocasiones elijamos al último candidato para así llenar el puesto porque necesitamos que alguien haga el

trabajo. Podemos generalizar esto a matroides. Llenar el cupo disponible en el caso clásico es como buscar una base en el caso general, en el sentido de buscar no solo algo factible, si no además maximal. Veamos un ejemplo más para motivar la búsqueda de bases. Supongamos que tenemos r sucursales de cierta franquicia y necesitamos contratar un gerente para cada una ellas. Entrevistamos n candidatos con las hipótesis del MSP, es decir, después de cada entrevista decidimos si contratar o no al entrevistado y esta decisión es irrevocable. Cada candidato está dispuesto a trabajar solo en ciertas sucursales (podemos pensar por ejemplo que algunas le quedan muy lejos de su casa). El problema anterior corresponde al MSP en una matroide transversal. Si bien es lógico querer contratar buenos gerentes, también es importante en este caso poner la mayor cantidad posible de sucursales en marcha, es decir, queremos elegir una base de la matroide transversal.

Consideremos el MSP usual donde la matroide es conocida de antemano y los pesos son no negativos. Supongamos que tenemos un algoritmo online ALG fuertemente c -competitivo para este problema (para c -competitivo también funciona). Vamos a modificar ALG para obtener un algoritmo ALG_{Base} que entregue una base sin afectar su competitividad. El algoritmo modificado es el siguiente:

Algoritmo 12 : ALG_{Base}

- 1: $A \leftarrow \phi$;
 - 2: **Para** $t = 1 \dots n$ **hacer**
 - 3: **Si** x_t es aceptado por ALG **entonces**
 - 4: $A \leftarrow A + x_t$;
 - 5: $C \leftarrow A \cup \overline{E}_t$;
 - 6: **Si** $x_t \notin \text{Span}(C)$ **entonces**
 - 7: $A \leftarrow A + x_t$;
 - 8: **Devolver** A ;
-

Lo que estamos haciendo es lo siguiente, en primer lugar aceptamos cada elemento que ALG acepta. Además, si x_t no es generado por la unión de los elementos ya aceptados con los elementos por venir, también lo aceptamos. Se puede ver que el conjunto devuelto por ALG está contenido en el conjunto devuelto por ALG_{Base} , por lo tanto, la competitividad no puede empeorar.

Lema 7.1. ALG_{Base} devuelve una base.

DEMOSTRACIÓN. Primero veamos que A es independiente. Llamemos A^* al conjunto de elementos aceptados por ALG y definamos $B_t = (A \setminus A^*) \cap E_{t-1}$. Es decir, el conjunto B_t contiene a los elementos aceptados por ALG_{Base} y no por ALG justo antes de decidir si incluir o no a x_t . Sabemos que $A^* \in \mathcal{I}$, veamos por inducción en t que $A^* \cup B_t \in \mathcal{I}$. Notemos que $B_1 = \phi$, así que tenemos el caso base. Suponemos entonces $A^* \cup B_t \in \mathcal{I}$. Si x_t no es elegido por ALG_{Base} o si es elegido por ALG no hay nada que probar. Luego, SPG $x_t \notin A^*$ y es elegido por ALG_{Base} . En este caso $x_t \notin \text{Span}(A^* \cup B_t \cup \overline{E}_t)$ y por monotonía del Span, $x_t \notin \text{Span}(A^* \cup B_t)$. Por lo tanto:

$$r(A^* \cup B_t + x_t) = r(A^* \cup B_t) + 1 = |A^* \cup B_t| + 1 = |A^* \cup B_t + x_t|.$$

Luego, $A^* \cup B_t + x_t = A^* \cup B_{t+1} \in \mathcal{I}$ (recordemos que $X \in \mathcal{I}$ si y solo si $r(X) = |X|$). Por lo tanto $A^* \cup B_{n+1} = A \in \mathcal{I}$.

Ahora veamos que A genera, basta mostrar que $r(A) = r$. Definimos $C_t = A \cup \overline{E}_t$ y $C_0 = E$. Usamos inducción para probar que $r(A) = r(C_n) = r$.

Si $t = 0$, $C_0 = E$ y $r(E) = r$. Suponemos $r(C_t) = r$, notemos que hay dos posibilidades para C_{t+1} . La primera es $C_{t+1} = C_t$ en cuyo caso no hay nada que probar. La segunda es $C_{t+1} = C_t - x_{t+1}$ en cuyo caso x_{t+1} no es aceptado por ALG_{base} . Luego, $x_{t+1} \in \text{Span}(C_{t+1})$ lo que significa que $r(C_{t+1}) = r(C_{t+1} + x_{t+1}) = r(C_t) = r$. Lo que concluye la demostración. \square

7.3. Algoritmo Cardinal $O(1)$ -competitivo para el MSP

En esta sección mostramos un algoritmo muy simple para el MSP y probamos que es cardinal $O(1)$ -competitivo para cualquier matroide.

Algoritmo 13

- 1: $A \leftarrow \phi$;
 - 2: $s \leftarrow \lceil pn \rceil$;
 - 3: **Para** $t = s + 1 \dots n$ **hacer**
 - 4: Calculamos OPT_t ;
 - 5: **Si** $x_t \in \text{OPT}_t$ y $A + x_t \in \mathcal{I}$ **entonces**
 - 6: $A \leftarrow A + x_t$;
 - 7: **Devolver** A ;
-

Para el análisis del Algoritmo 13 necesitamos definir dos conjuntos, el conjunto B de elementos buenos y el conjunto C de candidatos. Formalmente:

- $B = \{x_t : x_t \in \text{OPT} \text{ y } t > s\}$.
- $C = \{x_t : x_t \in \text{OPT}_t \text{ y } t > s\}$.

Tenemos que tanto A como B están contenidos en C , por lo tanto:

$$|C| \geq |A \cup B| = |A| + |B| - |A \cap B|.$$

Notemos además que $A \cap B = \text{OPT} \cap A$, despejando y aplicando esperanza obtenemos que:

$$\mathbb{E}(|\text{OPT} \cap A|) \geq \mathbb{E}(|A|) + \mathbb{E}(|B|) - \mathbb{E}(|C|).$$

Teorema 7.2. *El Algoritmo 13 es cardinal $10/3$ -competitivo.*

DEMOSTRACIÓN. Para demostrar esto vamos a acotar los tamaños esperado de A , B y C , y vamos a usar estas cotas en la desigualdad anterior. Cada elemento tiene probabilidad $(n - s)/n$ de estar fuera de la muestra, en particular los elementos de OPT , por lo tanto:

$$\mathbb{E}(|B|) = \frac{r(n - s)}{n} = \frac{r(n - \lceil pn \rceil)}{n} \geq \frac{r(n - pn - 1)}{n} = r(1 - p) - \frac{r}{n}.$$

Para acotar el valor esperado de $|A|$ notemos que el conjunto A es maximal bajo inclusión entre los independientes contenidos en C . Luego, $|A| = r(C) \geq r(B) = |B|$ y podemos usar

la cota anterior.

Finalmente, para acotar $\mathbb{E}(|C|)$ notemos en primer lugar que:

$$\mathbb{P}(x_t \in \text{OPT}_t) = \frac{r(E_t)}{t} \leq \frac{r}{t}$$

ya que si condicionamos en cualquier E_t , hay $r(E_t) \leq r$ elementos en OPT_t y x_t es un elemento elegido uniformemente al azar dentro de E_t . Luego:

$$\begin{aligned} \mathbb{E}(C) &= \sum_{t=s+1}^n \mathbb{P}(x_t \in \text{OPT}_t) \\ &\leq \sum_{t=s+1}^n \frac{r}{t} \\ &= r \int_{\lceil pn \rceil}^n \frac{1}{\lceil t \rceil} dt \\ &\leq r \int_{pn}^n \frac{1}{t} dt \\ &= r \ln(1/p). \end{aligned}$$

□

Juntando todo tenemos que:

$$\frac{\mathbb{E}(|\text{OPT} \cap A|)}{r} \geq 2 - 2p + \ln(p) - \frac{2}{n}.$$

Derivando obtenemos que $p = 1/2$ es óptimo y el lado derecho queda como $1 - \ln(2) - o(1)$ que (para n grande) es aproximadamente $0,3$. Dado que podemos inflar n y puesto que $1 - \ln(2) > 3/10$, podemos asegurarnos de elegir n suficientemente grande de modo que nuestro algoritmo sea cardinal $10/3$ -competitivo.

Capítulo 8

Problema de la Secretaria con Despidos Limitados

Otra variante del problema de la secretaria es el $j - k$ Secretary Problem, donde se pueden elegir j candidatos, pero sólo se obtiene beneficio por los mejores k , podemos pensar esto como la selección de un equipo deportivo con j jugadores y sólo k titulares, o como un problema de contratación donde hay que llenar k puestos y está permitido hacer $j - k$ despidos. En este capítulo estudiamos una generalización del problema anterior. Supongamos que tenemos una matroide cualquiera $\mathcal{M} = (E, \mathcal{I})$ con rango r , durante la ejecución del algoritmo se nos permite elegir cr elementos con c alguna constante positiva, y al final podemos botar elementos hasta obtener el mejor conjunto independiente contenido en los elementos preseleccionados. Para ver que este problema generaliza el $j - k$ Secretary Problem basta tomar una matroide k -uniforme y $c = j/k$.

La estrategia que usamos para este problema es muy sencilla, comenzamos como siempre tomando una muestra y a continuación aceptamos cualquier elemento que mejore lo ya visto hasta haber elegido un máximo de cr elementos. Para analizar la competitividad, la clave es ver que con alta probabilidad no hay muchos elementos que mejoren lo ya visto. El tamaño de la muestra lo elegimos más adelante. A continuación presentamos el algoritmo:

Algoritmo 14

- 1: Fijamos s ;
 - 2: $A \leftarrow \phi$;
 - 3: **Para** $t = s + 1 \dots n$ **hacer**
 - 4: **Si** $x_t \in \text{OPT}_t$ y $|A| < cr$ **entonces**
 - 5: $A \leftarrow A + x_t$
 - 6: **Devolver** $\text{OPT}(A)$
-

Para estudiar la competitividad de este algoritmo, calculemos la probabilidad de que un elemento de OPT sea aceptado. Como ya sabemos, si $x \in \text{OPT}$, entonces x es un elemento que mejora lo visto sin importar el momento en que llegue. Además si x queda en A , queda también en $\text{OPT}(A)$. Por lo tanto, para que un elemento del óptimo sea aceptado se necesitan solo dos cosas: que llegue después de la muestra y que el número de elementos que mejoran

desde que termina el muestreo hasta su llegada sea menor que cr .

Definimos las variables aleatorias $\{X_i\}_{i \in [n]}$, donde

$$X_i = \begin{cases} 1, & \text{si } x_i \in \text{OPT}_i. \\ 0, & \text{si no.} \end{cases}$$

Sea $x \in \text{OPT}$, usando las variables X_i podemos escribir lo siguiente:

$$\begin{aligned} \mathbb{P}(x \in \text{OPT}(A)) &= \mathbb{P}\left(t(x) > s \wedge \sum_{i=s+1}^{t(x)-1} X_i < cr\right) \\ &= 1 - \mathbb{P}\left(t(x) \leq s \vee \sum_{i=s+1}^{t(x)-1} X_i \geq cr\right) \\ &= 1 - \left(\mathbb{P}(t(x) \leq s) + \mathbb{P}\left(\sum_{i=s+1}^{t(x)-1} X_i \geq cr\right)\right) \\ &= \frac{n-s}{n} - \mathbb{P}\left(\sum_{i=s+1}^{t(x)-1} X_i \geq cr\right). \end{aligned}$$

El valor de X_i depende de E_i y de x_i , cuyas elecciones pueden verse como el resultado de elegir i elementos al azar de E (los primeros en llegar) y luego seleccionar uno de estos para ser x_i . Ahora, sea quien sea E_i , $|\text{OPT}_i| = r(E_i) \leq r$, de donde se deduce que :

$$\mathbb{E}(X_i) = \mathbb{P}(X_i = 1) \leq \frac{r}{i}.$$

Pues $X_i = 1$ si y solo si $x_i \in \text{OPT}_i$.

Queremos ver ahora que $\mathbb{P}(\sum_{i=s+1}^{t(x)-1} X_i \geq cr)$ no es muy grande. Una opción sería acotar lo anterior usando directamente la desigualdad de Markov, pero para obtener un buen resultado nos interesa usar una cota tipo Chernoff, para lo cual se requiere independencia. En pro de la fluidez del argumento solucionamos ese asunto más adelante y suponemos de momento que las variables $\{X_i\}_{i \in [n]}$ son una familia de Bernoullis independientes con parámetro r/i o 1 en caso que $r/i \geq 1$. Definimos además:

$$Z_i = \sum_{j=s+1}^i X_j \quad \text{y} \quad Z = \sum_{j=s+1}^{t(x)-1} X_j.$$

Así, Z_i corresponde al número de elementos que mejoran después de la muestra y antes del tiempo i . Por otro lado, Z es el número de elementos que mejoran después de la muestra y antes que x . Recordemos que como el orden de los elementos es aleatorio, $t(x)$ toma valores en $[n]$ de manera equiprobable.

Sea $a > 0$, tenemos que:

$$\mathbb{P}(Z \geq cr) = \mathbb{P}(e^{aZ} \geq e^{acr}) \leq \mathbb{E}(e^{aZ})e^{-acr}.$$

Lo anterior corresponde a la forma estándar en que se usa la desigualdad de Markov para obtener cotas de tipo Chernoff. Para acotar $\mathbb{E}(e^{aZ})$ separamos los cálculos en tres lemas.

Lema 8.1. $\mathbb{E}(e^{aZ}) = \frac{1}{n} \sum_{i=0}^{n-1} \mathbb{E}(e^{aZ_i}).$

DEMOSTRACIÓN.

$$\begin{aligned} \mathbb{E}(e^{aZ}) &= \sum_{k=0}^{n-1} e^{ak} \mathbb{P}(Z = k) \\ &= \sum_{k=0}^{n-1} e^{ak} \sum_{i=0}^{n-1} P(Z = k | t(x) - 1 = i) \mathbb{P}(t(x) - 1 = i) \\ &= \frac{1}{n} \sum_{i=0}^{n-1} \sum_{k=0}^{n-1} e^{ak} P(Z_i = k) \\ &= \frac{1}{n} \sum_{i=0}^{n-1} \mathbb{E}(e^{aZ_i}). \end{aligned}$$

□

Ahora acotamos $\mathbb{E}(e^{aZ_i})$, si $i \leq s$ es claro que $Z_i = 0$ y sigue que $\mathbb{E}(e^{aZ_i}) = 1$. Nos interesa el caso $i > s$.

Lema 8.2. Si $i > s$, $\mathbb{E}(e^{aZ_i}) \leq \left(\frac{i}{s}\right)^{r(e^a - 1)}$.

DEMOSTRACIÓN.

$$\begin{aligned} \mathbb{E}(e^{aZ_i}) &= \mathbb{E}\left(e^{a \sum_{j=s+1}^i X_j}\right) \\ &= \mathbb{E}\left(\prod_{j=s+1}^i e^{aX_j}\right) \\ &= \prod_{j=s+1}^i \mathbb{E}(e^{aX_j}) \\ &= \prod_{j=s+1}^i (e^a \mathbb{P}(X_j = 1) + \mathbb{P}(X_j = 0)) \\ &= \prod_{j=s+1}^i (1 + \mathbb{P}(X_j = 1)(e^a - 1)). \end{aligned}$$

Donde usamos la independencia para separar el valor esperado de un producto en el producto de los valores esperados. Para la siguiente cota usamos que $1 + y \leq e^y$.

$$\begin{aligned}
\prod_{j=s+1}^i (1 + \mathbb{P}(X_j = 1)(e^a - 1)) &\leq \prod_{j=s+1}^i \exp(\mathbb{P}(X_j = 1)(e^a - 1)) \\
&\leq \prod_{j=s+1}^i \exp\left(\frac{r}{j}(e^a - 1)\right) \\
&= \exp\left(\sum_{j=s+1}^i \frac{r}{j}(e^a - 1)\right) \\
&\leq \exp\left(r(e^a - 1) \int_s^i \frac{1}{x} dx\right) \\
&= \exp(r(e^a - 1) \ln(i/s)) \\
&= \left(\frac{i}{s}\right)^{r(e^a - 1)}.
\end{aligned}$$

□

Lema 8.3. $\mathbb{E}(e^{aZ}) \leq 1 + \frac{1}{r(e^a - 1)} \left(\frac{n}{s}\right)^{r(e^a - 1)}$.

DEMOSTRACIÓN.

$$\begin{aligned}
\mathbb{E}(e^{aZ}) &= \frac{1}{n} \sum_{i=0}^{n-1} \mathbb{E}(e^{aZ_i}) \\
&\leq \frac{1}{n} \left(\sum_{i=0}^s 1 + \sum_{i=s+1}^{n-1} \left(\frac{i}{s}\right)^{r(e^a - 1)} \right) \\
&= \frac{1}{n} \left(s + 1 + \frac{1}{s^{r(e^a - 1)}} \sum_{i=s+1}^{n-1} i^{r(e^a - 1)} \right) \\
&\leq \frac{1}{n} \left(s + 1 + \frac{1}{s^{r(e^a - 1)}} \int_0^n y^{r(e^a - 1)} dy \right) \\
&= \frac{1}{n} \left(s + 1 + \frac{1}{s^{r(e^a - 1)}} \left(\frac{n^{r(e^a - 1) + 1}}{r(e^a - 1) + 1} \right) \right) \\
&\leq \frac{s + 1}{n} + \frac{1}{r(e^a - 1)} \left(\frac{n}{s}\right)^{r(e^a - 1)} \\
&\leq 1 + \frac{1}{r(e^a - 1)} \left(\frac{n}{s}\right)^{r(e^a - 1)}.
\end{aligned}$$

□

Antes de continuar, para simplificar un poco las cosas tomamos $p = s/n$, el teorema que presentamos a continuación es válido para todo $a > 0$.

Teorema 8.4. Para cualquier $a > 0$, el Algoritmo 14 es, en nuestro modelo con despidos, fuertemente $\left[1 - p - e^{-acr} \left(1 + \frac{1}{r(e^a - 1)p^{r(e^a - 1)}}\right)\right]^{-1}$ -competitivo.

DEMOSTRACIÓN. Si suponemos primero que las variables X_i son independientes como antes y juntamos lo que tenemos hasta ahora obtenemos directamente el resultado. Lo que vamos a hacer es acoplar las variables X_i a variables Y_i de modo que $X_i \leq Y_i$ y que $\{Y_i\}_{i \in [n]}$ sea una familia de bernoullis independientes de parámetro r/i .

Para simular Y_i hacemos lo siguiente, vemos E_i y creamos una bola blanca por cada elemento de E_i . Si $i \leq r$ pintamos todas las bolas rojas. Si $i > r$ pintamos rojas r bolas partiendo por las que corresponden a OPT_i y eligiendo luego al azar las que hagan falta. Tomamos la bola correspondiente a x_i , notemos que como el orden es aleatorio esto equivale a tomar una de las i bolas al azar. Si la bola correspondiente a x_i es roja, $Y_i = 1$, y si no, $Y_i = 0$. De este modo la variable Y_i domina a X_i , es decir: $X_i = 1 \Rightarrow Y_i = 1$. Además Y_i tiene probabilidad r/i de ser 1, independiente del resto de las variables Y_j . Con esto, tenemos que:

$$\begin{aligned} \mathbb{P}(x \in \text{OPT}(A)) &= 1 - p - \mathbb{P}\left(\sum_{i=s+1}^{t(x)-1} X_i \geq cr\right) \\ &\geq 1 - p - \mathbb{P}\left(\sum_{i=s+1}^{t(x)-1} Y_i \geq cr\right). \end{aligned}$$

Y definiendo Z usando $\{Y_i\}_{i \in [n]}$ en vez de $\{X_i\}_{i \in [n]}$ podemos acotar usando el Lema 8.3 y concluir el teorema. \square

El teorema anterior lamentablemente no nos dice mucho (al menos a primera vista). Para terminar con algo más visual presentamos una tabla con la competitividad aproximada que obtenemos con el Algoritmo 14 para distintos valores de r y c .

r/c	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5
10	0,189	0,413	0,577	0,696	0,782	0,844	0,888	0,920	0,943	0,959
20	0,245	0,474	0,635	0,748	0,826	0,880	0,917	0,943	0,961	0,973
30	0,270	0,502	0,662	0,771	0,845	0,895	0,929	0,952	0,967	0,978
40	0,286	0,518	0,677	0,784	0,856	0,903	0,935	0,957	0,971	0,980
50	0,296	0,530	0,688	0,793	0,863	0,909	0,940	0,960	0,973	0,982
60	0,304	0,539	0,696	0,799	0,868	0,913	0,943	0,962	0,975	0,983
70	0,310	0,545	0,702	0,804	0,872	0,916	0,945	0,964	0,976	0,984
80	0,315	0,551	0,707	0,809	0,875	0,919	0,947	0,965	0,977	0,985
90	0,319	0,555	0,711	0,812	0,878	0,921	0,948	0,966	0,978	0,986
100	0,323	0,559	0,714	0,815	0,880	0,922	0,949	0,967	0,979	0,986

Tabla 8.1: Competitividad para distintos valores de r y c .

Para obtener los valores de la tabla, utilizamos el Teorema 8.4. El valor de p se fijó en función de a , c y r como $\exp[-acr/(r(e^a - 1) + 1)]$. Esto se obtiene de derivar la garantía

con respecto a p e igualar a 0. Si bien es cierto, al fijar s en forma determinista no tenemos derecho de elegir p como queramos, podemos acercarnos tanto como deseemos a cualquier valor entre 0 y 1 tomando n grande y s adecuado. El valor de a se optimizó usando la función *fmincon* de MATLAB, con a positivo y partiendo con un valor inicial $a = \ln(2)$.

Como era de esperar, vemos que la competitividad crece rápidamente cuando c crece, convergiendo a 1. Si fijamos c , vemos que la competitividad aumenta lentamente con r . Algo interesante de destacar es que se obtiene competitividad constante incluso con $c < 1$.

También vale la pena notar que si tomamos $c = 1$, podemos aplicar el Algoritmo 14 en una matroide k -uniforme sin que se descarte al final ningún elemento elegido. Tenemos así un algoritmo fuertemente $O(1)$ -competitivo para el MSP en matroides k -uniformes.

Conclusión

En esta tesis se estudia el problema de la secretaria en matroides (MSP). Probablemente la pregunta abierta más importante en relación a este problema es si se cumple la conjetura planteada por Babaioff et al. [1] que dice que para toda matroide existe un algoritmo $O(1)$ -competitivo. Tratamos de aportar en la búsqueda de una respuesta a esta pregunta avanzando en tres direcciones.

Primero presentamos y analizamos algoritmos fuertemente $O(1)$ -competitivos para el MSP en cuatro clases de matroides distintas: transversales, gráficas, representables k -sparsas y laminares. Para ser precisos, en el Capítulo 3 probamos que el Algoritmo 5 es fuertemente e -competitivo para el MSP en matroides transversales. En el Capítulo 4 probamos que el Algoritmo 6 es fuertemente 4-competitivo para el MSP en matroides gráficas, superando el algoritmo $2e$ -competitivo de Korula y Pál [13]. En el Capítulo 5 probamos que el Algoritmo 8 es fuertemente $\sqrt[k]{k^k}$ -competitivo para matroides representables con representación k -sparsa y mostramos que este resultado supera al algoritmo ke -competitivo presentado por Soto en [19]. En el Capítulo 6 probamos que el Algoritmo 10 es fuertemente $3\sqrt{3}$ -competitivo para el MSP en matroides laminares, superando el algoritmo fuertemente 9,6-competitivo obtenido por Ma et al. [16]. Además mostramos que todos estos algoritmos comparten un esquema similar, el cual llamamos Glotón Mañoso (Algoritmo 11).

En segundo lugar mostramos que existe un algoritmo cardinal $O(1)$ -competitivo que funciona en cualquier matroide, en otras palabras, probamos que en el MSP no es muy difícil elegir un conjunto independiente tal que el tamaño esperado de su intersección con el óptimo sea una fracción constante del tamaño de este último.

En tercer y último lugar, proponemos y estudiamos una variante del MSP. En esta variante está permitido preseleccionar hasta cr elementos sin preocuparnos de que los elementos preseleccionados formen o no un conjunto independiente. Al final del proceso debemos elegir entre los elementos preseleccionados un conjunto independiente. Para esta variante presentamos un algoritmo cuya competitividad depende de c y de r . Se observa que la competitividad es creciente en c y en r , llegando rápidamente a valores cercanos a 1 para $c \geq 4$. Además se obtiene competitividad constante incluso para $c = 1/2$.

Como futura línea de trabajo queda tratar de adaptar el esquema glotón mañoso para usarlo en otras clases de matroides o más generalmente en otros problemas de selección online.

Bibliografía

- [1] Moshe Babaioff, Nicole Immorlica, and Robert Kleinberg. Matroids, secretary problems, and online mechanisms. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 2007.
- [2] Sourav Chakraborty and Oded Lachish. Improved competitive ratio for the matroid secretary problem. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1702–1712. Society for Industrial and Applied Mathematics, 2012.
- [3] Nediálko B Dimitrov and C Greg Plaxton. Competitive weighted matching in transversal matroids. In *International Colloquium on Automata, Languages, and Programming*, pages 397–408. Springer, 2008.
- [4] Michael Dinitz and Guy Kortsarz. Matroid secretary for regular and decomposable matroids. *SIAM Journal on Computing*, 43(5):1807–1830, 2014.
- [5] Eugene B Dynkin. The optimum choice of the instant for stopping a markov process. In *Soviet Math. Dokl*, volume 4, 1963.
- [6] Moran Feldman, Ola Svensson, and Rico Zenklusen. A simple $o(\log \log(\text{rank}))$ -competitive algorithm for the matroid secretary problem. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1189–1201. Society for Industrial and Applied Mathematics, 2015.
- [7] Shayán Oveis Gharan and Jan Vondrák. On variants of the matroid secretary problem. *Algorithmica*, 67(4):472–497, 2013.
- [8] Gary Gordon and Jennifer McNulty. *Matroids: a geometric introduction*. Cambridge University Press, 2012.
- [9] Sungjin Im and Yajun Wang. Secretary problems: Laminar matroid and interval scheduling. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1265–1274. SIAM, 2011.
- [10] Patrick Jaillet, José A Soto, and Rico Zenklusen. Advances on matroid secretary problems: Free order model and laminar case. In *International Conference on Integer Pro-*

gramming and Combinatorial Optimization, pages 254–265. Springer, 2013.

- [11] Thomas Kesselheim, Klaus Radke, Andreas Tönnis, and Berthold Vöcking. An optimal online algorithm for weighted bipartite matching and extensions to combinatorial auctions. In *European Symposium on Algorithms*, pages 589–600. Springer, 2013.
- [12] Robert Kleinberg. A multiple-choice secretary algorithm with applications to online auctions. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 630–631. Society for Industrial and Applied Mathematics, 2005.
- [13] Nitish Korula and Martin Pál. Algorithms for secretary problems on graphs and hypergraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 508–520. Springer, 2009.
- [14] Oded Lachish. $O(\log \log \text{rank})$ competitive ratio for the matroid secretary problem. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 326–335. IEEE, 2014.
- [15] Denis V Lindley. Dynamic programming and decision theory. *Applied Statistics*, pages 39–51, 1961.
- [16] Tengyu Ma, Bo Tang, and Yajun Wang. The simulated greedy algorithm for several submodular matroid secretary problems. *Theory of Computing Systems*, 58(4):681–706, 2016.
- [17] Lex Schrijver. Combinatorial optimization-polyhedra and efficiency. *Algorithms and Combinatorics*, 24:1–1881, 2003.
- [18] Paul D Seymour. Decomposition of regular matroids. *Journal of combinatorial theory, Series B*, 28(3):305–359, 1980.
- [19] José A Soto. Matroid secretary problem in the random-assignment model. *SIAM Journal on Computing*, 42(1):178–211, 2013.
- [20] Hassler Whitney. On the abstract properties of linear dependence. *American Journal of Mathematics*, 57(3):509–533, 1935.
- [21] Wikipedia. Rigidity matroid — wikipedia, the free encyclopedia, 2017. [Online; accessed 17-March-2017].