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

PROBLEMA DE LA SECRETARIA SIMPLE Y MATROIDAL CON CONSEJOS ORDINALES

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO

FELIPE ANDRÉS VALDEVENITO WERNER

PROFESOR GUÍA:
JOSÉ SOTO SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
MARCOS KIWI KRAUSKOPF
VÍCTOR VERDUGO SILVA

Este trabajo ha sido parcialmente financiado por CMM ANID BASAL FB210005, FONDECYT REGULAR 1181180 y FONDECYT REGULAR 1231669

SANTIAGO DE CHILE
2023

RESUMEN TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS Y MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO
POR: FELIPE ANDRÉS VALDEVENITO WERNER
FECHA: 2023
PROF. GUÍA: JOSÉ SOTO SAN MARTÍN

PROBLEMA DE LA SECRETARIA SIMPLE Y MATROIDAL CON CONSEJOS ORDINALES

Examinamos variaciones del Problema de la Secretaria Simple (con una sola selección) y del Problema de la Secretaria Matroidal en un nuevo escenario, en el cual el tomador de decisiones (el *jugador*) puede solicitar uno (o más) bits de información ordinal de los candidatos antes de que lleguen.

Específicamente, cada candidato tiene información privada (su "peso") que es revelada al momento que el candidato llega. Como es usual, el jugador debe decidir irrevocablemente si seleccionar al candidato o no después de este paso. Adicionalmente, consideramos modelos en los que el jugador puede realizar preguntas ordinales binarias a los candidatos que aún no han llegado, y los candidatos pueden responder basándose solo en su propia información privada más la información públicamente disponible hasta el momento. Llamamos a esta información adicional *consejo ordinal*. Por ejemplo, el jugador puede preguntar a cada candidato que aún no llega si su peso oculto es mayor que todos los pesos de candidatos que ya han llegado, pero no le puede preguntar si su peso es el mayor de la lista completa.

En la versión del problema con única selección, el objetivo es seleccionar al candidato de mayor peso de la lista. Consideramos diferentes maneras en que los candidatos llegan, tales como en orden aleatorio, en orden adversarial, o con una muestra aleatoria de candidatos que se revelan de antemano. En el último caso, el jugador decide la cantidad de candidatos incluidos en la muestra, y el resto de los candidatos llegan en un orden elegido por un adversario ya sea antes o después de la realización de la muestra. Exploramos el impacto de variar la cantidad de consejos ordinales que se pueden solicitar, considerando los casos en que se puede consultar cero, una o ilimitadas veces. Presentamos algoritmos que alcanzan competitividades óptimas para casi todos los escenarios considerados.

Finalmente, investigamos la versión matroidal del problema, en la cual los candidatos son elementos de una matroide conocida, y el jugador solo puede seleccionar elementos incrementalmente tal que se forme un conjunto independiente de la matroide. Notablemente, probamos que incluso una sola ronda de consejos ordinales es suficiente para alcanzar competitividades constantes (independientes del rango de la matroide) si se dispone de una muestra aleatoria. También presentamos algoritmos con competitividades óptimas para varios de los escenarios presentados.

A la memoria de mi tía Heidi.

Tabla de Contenido

Introducción

En el Problema de la Secretaria, un conjunto de n candidatos para un puesto de trabajo es entrevistado uno a uno en un orden aleatorio uniforme. Tras cada entrevista, el empleador (a quien llamaremos el "jugador") aprende el "peso" de cada candidato, pudiendo comparar este peso con los pesos de los candidatos previamente entrevistados. Al finalizar una entrevista y luego de haber aprendido el peso del candidato, el jugador debe decidir en el momento si contratar al candidato en su oficina, finalizando el proceso de entrevistas, o rechazarlo de manera irrevocable y llamar al siguiente candidato. El objetivo del jugador es seleccionar al candidato de mayor peso oculto. Este problema ha sido ampliamente estudiado y ya es sabido (ver por ejemplo Lindley [22] y Dynkin [12]) que para cada n existe una estrategia para el jugador que le permite elegir al mejor candidato con probabilidad al menos $1/e$, y que esta probabilidad de selección es óptima asintóticamente.

Distintas generalizaciones de este problema han sido estudiadas. Algunas de ellas consideran distintos órdenes de llegada de los candidatos, por ejemplo, imponiendo que los candidatos lleguen en un orden seleccionado por un adversario que no quiere que el jugador elija al mejor candidato. Otras variaciones relajan las condiciones de contratación, permitiendo al jugador llamar y contratar a un candidato ya rechazado con alguna restricción y/o penalización. De gran interés son las variaciones en las que más de un candidato puede ser seleccionado, siempre que el conjunto seleccionado satisfaga una restricción combinatorial previamente definida. Por ejemplo, en el k-Problema de la Secretaria (ver por ejemplo [1]) se permite contratar a lo más k candidatos del total de n, y el jugador busca ya sea obtener el mayor peso total (suma de los pesos, suponiendo que éstos son números reales positivos) o maximizar la probabilidad de elegir al mejor candidato, entre otras funciones objetivo.

El Problema de la Secretaria Matroidal ([5],[4]) es una generalización en la que el conjunto de candidatos es el conjunto de referencia de una matroide, los candidatos tienes pesos reales positivos, y el objetivo del jugador es seleccionar un subconjunto de candidatos que forme un conjunto independiente de la matroide y que tenga el mayor peso total posible. Similar al problema original, después de procesar cada candidato, el jugador debe decidir si añadirlo a la solución actual o rechazarlo de manera irrevocable, con la restricción que el conjunto de candidatos seleccionados sea un conjunto independiente de la matroide durante todo el proceso. El rendimiento de un algoritmo para el Problema de la Secretaria Matroidal se mide con su competitividad, definida como la razón entre el peso esperado del conjunto independiente devuelto por el algoritmo y el peso del conjunto independiente de peso máximo de toda la instancia. En [5], Babaioff et al. conjeturan que, para cualquier matroide, debe existir un algoritmo con competitividad constante para el Problema de la Secretaria Matroidal asociado. A pesar del interés por el problema y la investigación realizada, la conjetura

permanece abierta, pero existen importantes resultados para casos específicos. Algoritmos competitivos para varias clases de matroides han sido desarrollados en los últimos años, y en muchos casos, estos algoritmos usan solo información ordinal de los pesos, permitiéndole al jugador comparar los pesos de los candidatos a medida que son revelados, similar al Problema de la Secretaria Clásico (ver por ejemplo [25] y las referencias mencionadas en la publicación). Para matroides generales, los mejores algoritmos obtenidos hasta el momento alcanzan competitividades de $\Omega(1/\log\log\rho)$ (ver por ejemplo [15]), donde ρ es el rango de la matroide.

Una característica interesante tanto del Problema de la Secretaria Clásico como del Problema de la Secretaria Matroidal es que el jugador no tiene absolutamente ninguna información de los pesos de los candidatos antes de que sean procesados. Existen variantes naturales del problema en las cuales se conoce información adicional del peso de un candidato antes de procesarlo. Por ejemplo, en el Problema del Profeta-Secretaria, cada candidato selecciona su peso de una distribución sobre los reales positivos, y todas las distribuciones son conocidas por el empleador por adelantado antes de ver los pesos reales de los candidatos (los cuales, de nuevo, llegan en orden uniformemente aleatorio). Usando esta información distribucional extra, el jugador puede alcanzar mejores competitividades. Para el Problema del Profeta-Secretaria con una sola selección, esta aumenta de $1/e \approx 0{,}3678$ (Problema de la Secretaria Clásico) a aproximadamente 0,669 [10]. Para el Problema del Profeta-Secretaria Matroidal, el actual mejor algoritmo alcanza una competitividad de $1 - 1/e \approx 0{,}63212$ [13], lo cual es objetivamente mucho mejor que el mejor algoritmo para el caso sin información distribucional.

Información adicional sobre candidatos aún no vistos también puede ser provista al jugador de otras maneras, por ejemplo, usando muestras [3, 8, 9] o historia basada en datos [17] (ver también [11]). Estos modelos son estadísticos en naturaleza. Otros modelos usan información obtenida a través de consejos aprendidos con aprendizaje de máquinas [2]. La idea principal en estos modelos es que, de antemano, una predicción numérica sobre la solución óptima concreta es dada, por ejemplo, en el caso con única selección, el máximo peso a ser visto. La predicción es conocida al principio del experimento y puede ser usada para guiar las decisiones. Si la predicción es siempre exacta, el problema se vuelve trivial, por lo cual se asume que hay un error de predicción, la cual es conocida por el jugador. El objetivo del jugador en este caso es tener una estrategia con una competitividad que crezca a medida que el error de predicción disminuye, mientras se mantenga un rendimiento comparable a lo que se puede obtener sin predicción en el caso donde el error de predicción es grande.

En esta tesis exploramos un método diferente para aprender información sobre candidatos aún no vistos que es más cercana a modelos de incertidumbre explorable, donde más información de la instancia puede ser obtenida a través de consultas [14]. En nuestros modelos, los pesos de cada candidato son información privada que se vuelve pública al momento que el algoritmo la procesa. Sin embargo, permitimos al jugador realizar cierta cantidad de preguntas binarias (con respuestas "sí" o "no") a los candidatos que aún no llegan, con las cuales el jugador puede obtener información parcial de los pesos privados. Las preguntas ha realizarse tienen que poder ser respondidas por el candidato usando solo información que se ha observado, por ejemplo, no se le puede preguntar a un candidato si su peso oculto es el mayor entre todos los candidatos, porque el candidato no tiene acceso a la información privada de los candidatos que aún no han sido procesados.

Adicionalmente, restringimos nuestros modelos al caso ordinal, donde consideramos que los pesos de los candidatos tienen un orden total asociado, pero no son necesariamente valores numéricos reales. De tal manera, no tiene sentido preguntarle a un candidato si su peso es mayor a cierto valor real, pero sí tiene sentido preguntarle si su peso es mayor que el peso de un candidato previamente visto. La razón por la que nos restringimos a modelos ordinales es, en primer lugar, por simplicidad: los modelos ordinales restringen fuertemente la variedad de algoritmos posibles para resolver un problema, lo cual facilita los cálculos de las cotas superiores de las garantías que se pueden alcanzar en cada modelo. En segundo lugar, porque se sabe que el problema cardinal (con valores reales) no es más fácil que el problema ordinal en el caso del Problema de la Secretaria Clásico (ver por ejemplo [22]). En tercer lugar, porque gran parte de los algoritmos propuestos en la literatura para el Problema de la Secretaria Matroidal son de naturaleza ordinal, es decir, solo usan información del ranking relativo de los pesos observados.

Nuestros Resultados

En esta tesis estudiamos variaciones del Problema de la Secretaria en el caso simple (donde se selecciona un único candidato) y en el caso matroidal. Específicamente, consideramos diferentes órdenes de llegada de los elementos/candidatos, considerando modelos con Orden Aleatorio (RO), modelos con Orden Adversarial (AO), modelos con Muestra Aleatoria y Orden Adversarial Online (SBOnAO), y modelos con Muestra Aleatoria y Orden Adversarial Offline (SBOffAO). Además, le damos la habilidad adicional al algoritmo de consultar a un oráculo que le entrega un "consejo" al jugador, el cual corresponde a un conjunto de respuestas a preguntas binarias previamente definidas como las mencionadas en el párrafo anterior. En el caso del problema simple y para cada orden mencionado, consideramos tres modelos de consejo: sin acceso al oráculo, con una llamada permitida al oráculo (lo cual simbolizamos con un subíndice 1 en las siglas de los modelos), y con ilimitadas llamadas permitidas al oráculo (lo cual simbolizamos con un subíndice ∞ en las siglas de los modelos). Esto produce 12 modelos para el caso simple.

Competitividades ajustadas para modelos sin consejo en el Problema de la Secretaria Simple eran conocidas para los modelos AO, RO y SBOnAO. En esta tesis presentamos algoritmos y cotas superiores de competitividad para las 9 otras posibilidades, las cuales son todas ajustadas, excepto en el modelo SBOffAO_1 donde existe una diferencia entre la cota inferior y la cota superior encontradas.

Para el Problema de la Secretaria Matroidal, consideramos dos tipos de consejo: consejo básico (B) y consejo completo (C). En esta tesis mostramos algoritmos para el modelo SBOnAO_1 con consejo básico, para el modelo SBOnAO_1 con consejo completo, y para el modelo RO_∞ con ilimitados consejos completos. En los dos últimos casos, los algoritmos tienen competitividades ajustadas. Cotas superiores e inferiores para otros modelos del problema matroidal se deducen trivialmente de estos resultados.

En las dos siguientes tablas se presentan las competitividades ajustadas o sus cotas para los diferentes modelos. Cabe señalar que usualmente, en otros trabajos, se dice que la competitividad del problema con orden adversarial (AO) y otros similares es $1/n$ en vez de 0 (donde

n es el número de elementos), pero en la definición que se dará en esta tesis, la competitividad se definirá como el ínfimo sobre las competitividades para cada n. Los teoremas asociados con cada nuevo resultado son referenciados en las tablas.

	Sin Consejo	Único Consejo	Ilimitados Consejos
RO	$= 1/e$	$\approx 0{,}4477$ (teorema 1.15)	$= 1/2$ (teorema 1.16)
SBOffAO	$= 1/4$ (teorema 1.13)	$\in [1/e, (20\sqrt{10} - 29)/81]$	$= (20\sqrt{10} - 29)/81$ (teorema 1.14)
SBOnAO	$= 1/4$	$= 1/e$ (teorema 1.12)	$= 1/e$ (teorema 1.12)
AO	$= 0$	$= 0$ (teorema 1.11)	$= 0$ (teorema 1.11)

Tabla 1: Valores aproximados/exactos, cotas inferiores y cotas superiores de las competitividades de modelos del Problema de la Secretaria Simple.

	Sin Consejo	Único Consejo B	Único Consejo C	Ilimitados Consejos C
RO	$\geq \Omega(1/\log\log\rho)$	$\geq 1/4*$	$\geq 1/e*$	$= 1/2$ (teorema 2.20)
SBOffAO	$\geq \Omega(1/\log\log\rho)$	$\geq 1/4*$	$\geq 1/e*$	$\geq 1/e*$
SBOnAO	$\geq \Omega(1/\log\log\rho)$	$\geq 1/4$ (teorema 2.18)	$= 1/e$ (teorema 2.19)	$= 1/e*$
AO	$= 0$	$= 0*$	$= 0*$	$= 0*$

Tabla 2: Valores aproximados/exactos, cotas inferiores y cotas superiores de las competitividades de modelos del Problema de la Secretaria Matroidal restringido a la clase de matroides de rango ρ. Entradas con * corresponden a cotas o valores que se deducen trivialmente de teoremas referenciados en las tablas.

Por un lado, el trabajo realizado en las variaciones del Problema de la Secretaria Simple resultan relevantes, dado el interés de la comunidad en estudiar variaciones del problema original en las que el tomador de decisiones obtiene información adicional futura. Varios modelos en la literatura consideran información futura incierta, por lo que el estudio de modelos con consejos (no inciertos) puede ser un punto de partida para establecer las limitaciones que pueden tener otros modelos de más interés, donde los consejos recibidos podrían no ser ciertos con alguna probabilidad positiva.

Por otro lado, al momento de la escritura de esta tesis no se conocen algoritmos con competitividad constante para el Problema de la Secretaria Matroidal original, por lo que los resultados obtenidos en las variaciones del Problema de la Secretaria Matroidal son relevantes, dado que nos muestran que se pueden alcanzar competitividades constantes si se da un poco más de información al jugador que toma las decisiones. Eventualmente, las ideas usadas en esta tesis podrían ser usadas para encontrar algoritmos con competitividades constantes para el problema original o en otros problemas relacionados (problema de la secretaria en matchings, en k-sistemas, entre otros).

Capítulo 1

Problema de la Secretaria Simple

En este capítulo estudiamos las variaciones del Problema de la Secretaria Simple que se obtienen al considerar dos modificaciones:

1. Cambiar el orden de llegada de los elementos.
2. Permitir al jugador consultar a un *oráculo de consejos* que le entrega cierta información ordinal de los pesos de los candidatos aún no vistos.

En la primera sección definimos el problema general, describimos los diferentes órdenes y el oráculo de consejos a considerar, y enunciamos y demostramos lemas a usarse en el capítulo. En la segunda sección revisamos los modelos previamente estudiados y para los cuales ya se conocían algoritmos óptimos. En la tercera sección estudiamos modelos que no se han estudiado previamente, y presentamos cotas inferiores y superiores para la competitividad de estos.

1.1. Preliminares

1.1.1. Definición del Problema

En primer lugar, describimos lo que en esta tesis se entenderá como el Problema de la Secretaria (Simple). Cabe señalar que, en la literatura, el Problema de la Secretaria se define usualmente como el problema que en esta tesis llamamos el Problema de la Secretaria Simple con Orden Aleatorio sin Consejo, ya que en esta tesis la definición incluye las variantes en las que el orden de llegada de los candidatos no es aleatorio uniforme. Hacemos luego la distinción entre el problema sin muestra aleatoria y el problema con muestra aleatoria.

Consideremos conjuntos finitos E y W con $n := |E| = |W|$, y $w : E \to W$ una función biyectiva, donde $(W, \leq)$ es un orden total. E es el conjunto de candidatos, w es la función de pesos, y, para cada $x \in E$, $w(x)$ se denomina el peso de x.

Problema de la Secretaria Simple sin Muestra Aleatoria. En el Problema de la Secretaria Simple sin Muestra Aleatoria, consideramos un jugador que conoce el conjunto E, pero desconoce $(W, \leq)$ y la asignación de pesos w, los cuales son asignados de manera adversarial. El jugador recibe de uno en uno los elementos de E en un orden $\pi : [n] \to E$ que dependerá del modelo. Aquí, para $i \in [n]$, $\pi(i)$ corresponde el i-ésimo elemento que llega cuando el orden es π. Cada vez que llega un elemento $x \in E$, su peso $w(x)$ es revelado al jugador, en cuyo punto el jugador debe decidir si seleccionar x o no. Si el elemento es seleccionado, el proceso termina. Si el elemento no es seleccionado (decimos que el jugador *rechaza* el elemento), el jugador recibe el siguiente elemento en el orden, perdiendo irrevocablemente la oportunidad de seleccionar en el futuro el elemento saltado. Una vez que se rechaza el último elemento en llegar, el proceso termina. Suponemos que el jugador puede verificar en cualquier momento si $w_1 \leq w_2$ o $w_2 \leq w_1$ para $w_1, w_2 \in W$ pesos ya observados por él, es decir, el jugador sabe cómo están ordenados los pesos de los elementos que ya ha observado.

Problema de la Secretaria Simple con Muestra Aleatoria. En el Problema de la Secretaria Simple con Muestra Aleatoria, el jugador primero decide un tamaño de muestra $s \in \{0, 1, ..., n\}$ (posiblemente aleatorio) antes de recibir el primer elemento. El jugador luego pasa a una *fase de muestreo*, en la cual una muestra aleatoria uniforme $S \subseteq E$ de tamaño s se revela al jugador, mostrándole $w(x)$ para cada $x \in S$. Los elementos en esta muestra no pueden ser seleccionados por el jugador. Una vez recibida la muestra, el jugador pasa a la *fase de selección*, en la cual el resto de los elementos (aquellos en $E \setminus S$) llegan en un orden que dependerá del modelo, y se aplican las mismas reglas de selección que en el modelo sin muestra. Un algoritmo para un Problema de la Secretaria con Muestra Aleatoria se puede pensar como una tupla $(A', (\mathcal{D}_n)_{n\in\mathbb{N}})$, donde cada $\mathcal{D}_n$ es una distribución con soporte en $\{0, ..., n\}$ y A' es el *algoritmo post-muestra*. Entendemos la tupla como sigue: Si $|E| = n$, el tamaño de muestra s es una realización de una variable aleatoria con distribución $\mathcal{D}_n$ y A' es la parte del algoritmo que se hace cargo de los elementos que llegan después de la muestra, tratando de elegir el más pesado.

Usualmente, el Problema de la Secretaria se define asignando la función de pesos adversarial $w : E \to \mathbb{R}_+$, lo cual se conoce como la versión cardinal del Problema de la Secretaria. En esa versión, el jugador podría eventualmente usar la información numérica de los pesos de los elementos para tomar decisiones (por ejemplo, podría realizar operaciones aritméticas con los valores de los pesos), sin embargo, las competitividades de los modelos no cambian al pasar del caso ordinal al caso cardinal para el problema simple (ver por ejemplo [22]). Enfocándonos en el caso ordinal, describir algoritmos generales para un modelo resulta ser mucho más sencillo, razón por la cual omitimos el caso cardinal en nuestros análisis. La definición del Problema de la Secretaria dada en los párrafos anteriores es de naturaleza ordinal, en el sentido que el jugador no ve necesariamente valores numéricos en los pesos de los elementos. Es más, es fácil notar que como la elección de W es adversarial, entonces se puede suponer sin pérdida de generalidad que los algoritmos solo son capaces de comparar los pesos entre ellos a través de la relación de orden de $(W, \leq)$, no pueden sacar información de los elementos de W por sí solos ni operarlos. Si nos restringimos a este tipo de algoritmos, la elección de W es irrelevante, podemos pensar que W es fijo si nos limitamos a considerar solo algoritmos que toman decisiones en base a la comparación de pesos vistos. Notemos además que los elementos de E solo son distinguibles por sus pesos para efectos del Problema de la

Secretaria Simple, por lo que podemos asumir que el jugador solo conoce $n = |E|$ antes de iniciar el proceso.

Muchas veces usaremos la palabra *algoritmo* para referirnos al jugador, considerando que las estrategias de selección a presentar se escriben como algoritmos. En ambos problemas descritos, decimos que el jugador *gana* si el elemento seleccionado es el de mayor peso de E, es decir, si el elemento seleccionado es $\operatorname{argmax}_{x \in E} w(x)$, donde el máximo se define según la relación de orden de W. El objetivo del jugador es ganar, por lo cual la siguiente métrica resulta relevante para evaluar qué tan bueno es un algoritmo:

Definición 1.1 (Competividad (Problema de la Secretaria Simple)) *Sea $\mathcal{M}$ un modelo del Problema de la Secretaria Simple, A un algoritmo para este modelo y $\alpha \in [0, 1]$. Definimos la competitividad de A para el modelo $\mathcal{M}$ como:*

$$\alpha_{\mathcal{M}}(A) := \inf_{n \in \mathbb{N}^*} \alpha^n_{\mathcal{M}}(A)$$

donde $\alpha^n_{\mathcal{M}}(A)$ denota la probabilidad de que A gane en instancias de tamaño n. Decimos que A es α-competitivo para el modelo $\mathcal{M}$ si $\alpha \geq \alpha_{\mathcal{M}}(A)$. Definimos la competitividad del modelo $\mathcal{M}$ en instancias de tamaño n como:

$$\alpha^n_{\mathcal{M}} := \max_{A'} \alpha^n_{\mathcal{M}}(A'),$$

donde el máximo se toma sobre los algoritmos para el problema, y definimos la competitividad del modelo $\mathcal{M}$ como:

$$\alpha_{\mathcal{M}} := \inf_{n \in \mathbb{N}^*} \alpha^n_{\mathcal{M}}.$$

Es claro de las definiciones anteriores que un algoritmo es α-competitivo si el algoritmo gana con probabilidad mayor o igual a α en cualquier instancia del problema, y que la competitividad de un algoritmo es igual al mínimo α tal que el algoritmo es α-competitivo. También se puede ver que $\alpha_{\mathcal{M}} = \max_A \alpha_{\mathcal{M}}(A)$.

En esta tesis no nos interesará estudiar la complejidad computacional de los algoritmos ni de los modelos presentados, sino que nos interesará estudiar sus competitividades, lo cual es usual en el contexto de problemas de optimización *online*.

A lo largo de esta tesis, usamos además las siguientes notaciones. Decimos que un elemento fue *procesado* si fue seleccionado/agregado o rechazado. Decimos que $x \in E$ llega en el *paso* i si es el i-ésimo elemento en llegar. Decimos que un elemento es un *récord* si es el elemento más pesado observado hasta el paso en el que llega. Llamamos $\Pi(E)$ al conjunto de todos los órdenes de E, los cuales describimos como funciones $\pi : [n] \to E$.

Escribiremos $E = \{x^1, x^2, ..., x^n\} = \{x_1, x_2, ..., x_n\}$, donde $w(x^1) \geq w(x^2) \geq ... \geq w(x^n)$ y $x_i := \pi(i)$ para cada $i \in [n]$, es decir, usaremos superíndices para ordenar a los elementos en orden decreciente de pesos y subíndices para ordenar a los elementos según el orden de llegada. Claramente, la notación de superíndices es un recurso que usamos para las demostraciones, pero no es información que se revela al jugador cuando se revela el peso de un elemento. Para cada $i \in [n]$, definiremos $E^i = \{x^1, ..., x^i\}$, el conjunto de los i elementos más pesados, y $E_i := \{x_1, ..., x_i\}$ el conjunto de los primeros i elementos en llegar. También escribimos $E^0 = E_0 := \emptyset$.

1.1.2. Órdenes de Llegada

Consideramos modelos para 4 órdenes de llegada de los candidatos: Orden Aleatorio (RO por *Random Order*), Orden Adversarial (AO por *Adversarial Order*), Orden Adversarial Online con Muestra (SBOnAO por *Sample-Based Online Adversarial Order*), Orden Adversarial Offline con Muestra (SBOffAO por *Sample-Based Offline Adversarial Order*). Los primeros dos se usan en el contexto del problema sin muestra, mientras que los últimos dos se usan en el contexto del problema con muestra. Llamemos π al orden de llegada.

En los modelos con Orden Aleatorio, π se selecciona uniformemente al azar entre todos los órdenes de E. En particular, para un algoritmo A, $\alpha_{\text{RO}} := \mathbb{P}_{\pi\sim\text{Unif}(\Pi(E))}[A \text{ gana}]$, donde la probabilidad es tomada sobre la aleatoriedad intrínseca de A y sobre la aleatoriedad de π.

En los modelos con Orden Adversarial, π es elegido por un adversario que conoce el algoritmo que se usará, pero no conoce las realizaciones de variables aleatorias dentro del algoritmo. Por ejemplo, un algoritmo muy simple puede seleccionar un índice $i \sim \text{Unif}([n])$ y elegir el i-ésimo elemento de E que llegue; en tal caso, consideramos que un adversario conoce esta regla de selección, pero que no puede descifrar la realización de i de antemano. Entonces, para un algoritmo A, $\alpha_{\text{AO}}(A) := \text{mín}_{\sigma\in\Pi(E)}\,\mathbb{P}[A \text{ gana}|\pi = \sigma]$.

En los modelos con Muestra Aleatoria y Orden Adversarial Online/Offline trabajamos con el Problema de la Secretaria con Muestra Aleatoria descrito en la sección anterior. Podemos pensar que los elementos de la muestra aleatoria llegan en un orden arbitrario antes que los elementos fuera de la muestra. Usamos la siguiente notación: Si la muestra S tiene tamaño s, $\pi|_{[s]} : [s] \to S$ es una función biyectiva arbitraria. Siguiendo esta notación, si x es el i-ésimo elemento en llegar en la fase de selección, entonces $\pi(s+i) = x$, i.e., x es el $(s+i)$-elemento en llegar.

En los modelos con muestra aleatoria y orden adversarial online, después de que el algoritmo tome la muestra $S \subseteq E$, el conjunto $E \setminus S$ llega de manera adversarial en la fase de selección, y este orden puede depender de S (decimos que éste es un *adversario online* que conoce la realización de S). Es decir, condicionado al valor de $\pi|_{[s]}$, la función biyectiva $\pi|_{[n]\setminus[s]} \to E \setminus \pi|_{[s]}([s])$ se elige de manera adversarial. Por ejemplo, si $\hat{x} \neq x^1$ es el elemento más pesado en la muestra S, el adversario online podría ordenar los elementos más pesados que $\hat{x}$ en orden creciente de pesos y luego entregar el resto de los elementos de $E \setminus S$ en orden decreciente de pesos.

En los modelos con Muestra Aleatoria y Orden Adversarial Offline, un adversario elige un *orden preliminar* $\psi : [n] \to E$ antes de la toma de muestra, luego ψ es independiente de la muestra S (sí podría depender de $\mathcal{D}_n$). Este orden no es conocido por el algoritmo. Cuando el algoritmo entra en la fase de selección, los elementos de $E \setminus S$ llegan en el único orden $\hat{\psi} : [n] \setminus [s] \to E \setminus S$ que es consistente con el orden preliminar ψ, esto es, para todo $x, y \in E \setminus S$, $\hat{\psi}^{-1}(x) < \hat{\psi}^{-1}(y)$ si y solo si $\psi^{-1}(x) < \psi^{-1}(y)$. A este adversario lo llamamos *adversario offline*. Para visualizar mejor estos modelos, le puede resultar útil revisar el ejemplo en la Figura 1.1.

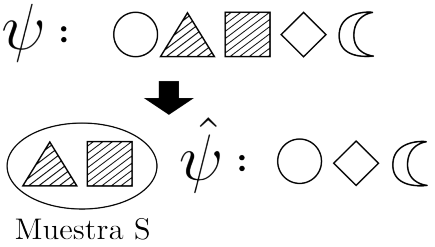

Figura 1.1: Orden Adversarial Offline. Si ψ tiene a las figuras en el orden de arriba y las figuras achuradas corresponden a las que son muestreadas, las figuras fuera de muestra se ordenan según $\hat{\psi}$.

Dentro del análisis a realizar en la tercera sección de este capítulo, nos preguntamos si se puede lograr mejor competitividad ante el adversario offline que ante el adversario online. Intuitivamente, el adversario offline es *más débil* que el adversario online, porque con el adversario offline el algoritmo podría obtener información del orden preliminar a través de la muestra, mientras que el adversario online se adapta a la muestra obtenida por el algoritmo.

1.1.3. Oráculo de Consejos

Además de cambiar el orden de llegada de los elementos, nos interesa estudiar como mejoran (si es que mejoran) las competitividades de los modelos si le damos información ordinal adicional al algoritmo en ciertas etapas del proceso. Imaginemos que después de rechazar un elemento y antes de recibir el siguiente, el jugador puede realizar preguntas ordinales con respuestas "sí" o "no" a los candidatos que aún no revelan sus pesos, llamamos a esta fase entre el rechazo de un elemento y la recepción del siguiente una *ronda de consejos*. Como el jugador no conoce el conjunto W de antemano (solo conoce los pesos que se le han revelado), les puede preguntar a cada uno de los candidatos pendientes si sus pesos son mayores o menores a uno o varios de los pesos que ya se han revelado públicamente. Suponemos que los candidatos responden a las preguntas con sinceridad.

Primero, notemos que en el caso del Problema de la Secretaria Simple existe una pregunta a hacer a los candidatos que es objetivamente mejor que el resto. Supongamos que después de rechazar el último elemento visto, el mayor peso visto hasta el minuto es τ, entonces, la mejor pregunta a hacer a cada candidato es si su peso es mayor a τ o no. En efecto, como el algoritmo solo gana si selecciona el elemento más pesado, hacer otras preguntas ordinales resulta redundante para encontrar el elemento más pesado, el cual tiene peso mayor a τ si aún no ha sido procesado. Estudiaremos variantes de los modelos en las cuales se realiza esa pregunta a cada candidato pendiente en las rondas de consejo, recibimos así un bit de información por cada candidato pendiente en cada ronda de consejos. Podría resultar interesante en trabajos posteriores estudiar variantes en las que solo se pueda hacer preguntas más débiles, por ejemplo, solo preguntar a cada candidato pendiente si su peso es mayor a τ_k, donde τ_k es el k-ésimo mayor peso visto hasta el momento.

Después de realizar una ronda de consejos preguntando a los candidatos si sus pesos son mayores que el mayor peso visto hasta el momento, obtenemos el conjunto de candidatos

que responden "sí" y el conjunto de los que responden "no". Como los candidatos son indistinguibles más allá de sus pesos, la única información que el algoritmo realmente aprende después de la ronda de consejos es la cantidad de respuestas positivas. Definimos el *consejo* para el problema simple de la siguiente manera:

Definición 1.2 (Consejo) *Para un orden de llegada π dado, función de pesos w, e $i \in [n]$, el consejo en el paso i, denotado* adv_i*, se define como*

$$\text{adv}_i := |\{x \in E \setminus E_i : w(x) \geq \max_{j \in [i]} w(x_j)\}|,$$

donde recordamos que $x_j := \pi(j)$ y $E_i := \{x_1, ..., x_i\}$.

En los modelos con consejo, el jugador tendrá acceso a un *oráculo de consejos* que se puede consultar después de rechazar un elemento y antes de recibir el siguiente. Si el jugador consulta el oráculo después de rechazar el i-ésimo elemento, recibe adv_i (es decir, el oráculo le informa la cantidad de elementos no vistos con peso mayor que el más pesado ya visto), tras lo cual el proceso de llegada de elementos continúa normalmente. Dividiremos los modelos según la cantidad permitida de consultas al oráculo: modelos sin acceso al oráculo de consejos (los llamamos *sin Consejo*), modelos en los que se puede consultar al oráculo de consejos una sola vez (*Consejo Único*) y modelos en los que se puede consultar al oráculo de consejos en cada paso (*Consejos Ilimitados*). Si pensamos en las preguntas binarias descritas en los párrafos anteriores, cada consulta al oráculo corresponde a una ronda de consejos.

Para cada modelo de orden $\mathcal{M}$, escribimos $\mathcal{M}_0$ o simplemente $\mathcal{M}$ para referirnos al modelo sin consejo, $\mathcal{M}_1$ para referirnos al modelo con consejo único, y $\mathcal{M}_\infty$ para referirnos al modelo con consejos ilimitados. Considerando los 4 tipos de órdenes de llegada y las 3 cantidades de consejos, tenemos en total 12 modelos por revisar.

1.1.4. Jerarquía de los Modelos

Nos interesa entender como se comparan las competitividades de los modelos a considerar. Se observan dos jerarquías en este sentido. Primero, para el modelo de orden $\mathcal{M}$, claramente $\alpha_{\mathcal{M}} \leq \alpha_{\mathcal{M}_1} \leq \alpha_{\mathcal{M}_\infty}$, porque en modelos con más rondas de consejos el algoritmo puede decidir ignorar los consejos, resultando en modelos con menos rondas de consejos.

Segundo, se tiene que para una cantidad fija de rondas de consejos $k \in \{0, 1, \infty\}$, $\alpha_{\text{AO}_k} \leq \alpha_{\text{SBOnAO}_k} \leq \alpha_{\text{SBOffAO}_k} \leq \alpha_{\text{RO}_k}$. En efecto, el modelo con Orden Adversarial (AO) corresponde al modelo con Orden Adversarial Online con Muestra (SBOnAO) en el que se restringe a que el tamaño de muestra sea $s = 0$, luego el jugador tiene más estrategias posibles en el modelo con Orden Adversarial Online con Muestra, con lo cual $\alpha_{\text{AO}_k} \leq \alpha_{\text{SBOnAO}_k}$. En el modelo con Orden Adversarial Online y Muestra (SBOnAO), un adversario puede seleccionar un orden preliminar $\psi : [n] \to E$ previo al muestreo y luego ordenar los elementos fuera de muestra acorde a ψ como en el modelo con Orden Adversarial Offline con Muestra (SBOffAO), luego en el modelo con Orden Adversarial Offline con Muestra el adversario tiene menos estrategias posibles para que el jugador no gane, con lo cual $\alpha_{\text{SBOnAO}_k} \leq \alpha_{\text{SBOffAO}_k}$. Por último, el modelo con Orden Aleatorio (RO) corresponde a un caso particular del modelo

con Orden Adversarial Offline con Muestra (SBOffAO), en el cual $\psi : [n] \to E$ se elige de manera uniforme entre todos los órdenes de E, con lo cual $\alpha_{\mathrm{SBOffAO}_k} \leq \alpha_{\mathrm{RO}_k}$.

Resulta de interés entender cuando las desigualdades entre las competitividades son estrictas, en particular, nos interesa entender cuando agregar rondas de consejo permite mejorar la competitividad de los modelos. También resulta muy relevante comparar los modelos con Orden Adversarial Online con Muestra (SBOnAO) con los modelos con Orden Adversarial Offline con Muestra (SBOffAO), donde a priori desconocemos si existen diferencias estrictas entre las competitividades.

1.1.5. Lemas Auxiliares

Muchos de los algoritmos que se plantearan en las siguientes secciones partirán tomando una muestra de tamaño s, donde se tomará $s \sim \mathrm{Bin}(n, p)$ con $p \in [0, 1]$. Esto resulta ser útil y facilitará muchos cálculos, según lo que mostraremos en el siguiente lema (el cual se considera folklore):

Lema 1.3 *Sea $p \in [0, 1]$. El experimento de tomar una muestra $S \subseteq E$ uniformemente aleatoria de tamaño $s \sim \mathrm{Bin}(n, p)$ es equivalente al experimento de tomar una muestra $S \subseteq E$ en la que cada elemento de E tiene probabilidad p de aparecer en la muestra de manera independiente (muestreo de Bernoulli).*

Demostración. En efecto, es claro que si se muestrea cada elemento de E con probabilidad p de manera independiente, entonces la muestra obtenida es una muestra aleatoria uniforme de tamaño $s \sim \mathrm{Bin}(n, p)$. Veamos la recíproca, que es lo que nos interesa más. Basta mostrar que, para $X, Y \subseteq E$ con $|X| = m$, $|Y| = l$, $X \cap Y = \emptyset$, al tomar una muestra S de tamaño $s \sim \mathrm{Bin}(n, p)$, se tendrá que

$$\mathbb{P}[X \subseteq S, Y \cap S = \emptyset] = p^m (1 - p)^l.$$

Esto por un lado muestra que cada elemento se muestrea con probabilidad p y que cada elemento se muestrea de manera independiente.

En efecto, para $i \in \{m, ..., n - m - l\}$, $\mathbb{P}[X \subseteq S, Y \cap S = \emptyset | s = i] = \binom{n-m-l}{i-m} / \binom{n}{i}$; y $\mathbb{P}[s = i] = \binom{n}{i} p^i (1 - p)^{n-i}$. Desarrollamos:

$$\begin{aligned}
\mathbb{P}[X \subseteq S, Y \cap S = \emptyset] &= \sum_{i=m}^{n-l} \mathbb{P}[X \subseteq S, Y \cap S = \emptyset | s = i] \cdot \mathbb{P}[s = i] \\
&= \sum_{i=m}^{n-l} \binom{n - m - l}{i - m} p^i (1 - p)^{n-i} \\
&= p^m (1 - p)^l \sum_{j=0}^{n-m-l} \binom{n - m - l}{j} p^j (1 - p)^{n-m-l-j} \\
&= p^m (1 - p)^l \cdot (p + (1 - p))^{n-m-l} = p^m (1 - p)^l,
\end{aligned}$$

donde en la penúltima igualdad usamos el teorema del binomio. □

En general, los algoritmos para los diferentes modelos del Problema de la Secretaria parten tomando una muestra aleatoria de cierto tamaño s posiblemente aleatorio (ya sea porque el modelo les permite tomar muestra, o simplemente el algoritmo parte rechazando s elementos en el caso en que llegan en orden aleatorio). Para cada algoritmo A y cada valor de $n \in \mathbb{N}^*$, existe un tamaño de muestra s_n óptimo para la regla de selección post-muestra de A, por lo que podríamos suponer sin pérdida de generalidad que los algoritmos óptimos tienen un tamaño de muestra determinista dependiente de n. Esto se usa principalmente cuando queremos calcular cotas superiores de las competitividades de los modelos. Para estos cálculos, en particular para los modelos con adversario offline, resultará útil el siguiente lema técnico:

Lema 1.4 *Sea $k \in \mathbb{N}^*$. Sea A un algoritmo para un modelo $\mathcal{M}$ que parte tomando una muestra aleatoria de tamaño s_n (o rechazando los primeros s_n elementos en llegar) y que satisface:*

$$\alpha^n_{\mathcal{M}}(A) \leq \left(\sum_{\substack{i,j \in \mathbb{N} \\ i+j \leq k}} a_{ij} \, \frac{s_n^{\underline{i}}(n-s_n)^{\underline{j}}}{n^{\underline{i+j}}} \right) + o_n,$$

con $o_n = o(1) \geq 0$ y a_{ij} constantes en $[0,1]$, donde $x^{\underline{m}} := \prod_{h=0}^{m-1}(x-h)$ es un factorial descendente. Entonces, se tiene que:

$$\alpha_{\mathcal{M}}(A) \leq \max_{p \in [0,1]} \sum_{\substack{i,j \in \mathbb{N} \\ i+j \leq k}} a_{ij} \, p^i (1-p)^j.$$

Demostración. La sucesión $(s_n/n)_{n \in \mathbb{N}^*}$ está contenida en $[0,1]$, luego, por el teorema de Bolzano-Weierstrass, existe una subsucesión convergente $(s_{l(n)}/l(n))_{n \in \mathbb{N}^*}$. Sean $I := l(\mathbb{N}^*)$ y $\hat{p} := \lim_{n \to \infty}(s_{l(n)}/l(n))$. Acotamos de la siguiente manera:

$$\begin{aligned}
\alpha_{\mathcal{M}}(A) &= \inf_{n \in \mathbb{N}^*} \alpha^n_{\mathcal{M}}(A) \\
&\leq \inf_{n \in I} \alpha^n_{\mathcal{M}}(A) \\
&\leq \inf_{n \in I} \left(\sum_{\substack{i,j \in \mathbb{N} \\ i+j \leq k}} a_{ij} \, \frac{s_n^{\underline{i}}(n-s_n)^{\underline{j}}}{n^{\underline{i+j}}} \right) \\
&\leq \sum_{\substack{i,j \in \mathbb{N} \\ i+j \leq k}} a_{ij} \, \lim_{n \to \infty} \left(\frac{s_{l(n)}^{\underline{i}}(l(n)-s_{l(n)})^{\underline{j}}}{l(n)^{\underline{i+j}}} \right) \\
&= \sum_{\substack{i,j \in \mathbb{N} \\ i+j \leq k}} a_{ij} \, \hat{p}^i (1-\hat{p})^j \\
&\leq \max_{p \in [0,1]} \sum_{\substack{i,j \in \mathbb{N} \\ i+j \leq k}} a_{ij} \, p^i (1-p)^j,
\end{aligned}$$

donde en la penúltima línea se usa que, para todo i, j fijos:

$$\lim_{n\to\infty} \frac{s_{l(n)}^{\underline{i}}}{l(n)^{\underline{i}}} = \prod_{h=0}^{i-1} \lim_{n\to\infty} \frac{s_{l(n)} - h}{l(n) - h} = \prod_{h=0}^{i-1} \hat{p} = \hat{p}^i$$

$$\lim_{n\to\infty} \frac{(l(n) - s_{l(n)})^{\underline{j}}}{(l(n) - i)^{\underline{j}}} = \prod_{h=0}^{j-1} \lim_{n\to\infty} \frac{l(n) - s_{l(n)} - h}{l(n) - i - h} = \prod_{h=0}^{j-1} (1 - \hat{p}) = (1 - \hat{p})^j.$$

□

Notemos que en la demostración anterior es clave que k no dependa de n para poder meter el límite en la suma y el producto. Expresiones como $\sum_{\substack{i,j\in\mathbb{N} \\ i+j\leq k}} a_{ij} \frac{s_n^{\underline{i}}(n-s_n)^{\underline{j}}}{n^{\underline{i+j}}}$ son comunes, dado que, si se toma una muestra S de tamaño s_n y $X, Y \subseteq E$ con $|X| = i$, $|Y| = j$ y $X \cap Y = \emptyset$, entonces $\mathbb{P}[X \subseteq S, Y \cap S = \emptyset] = \frac{s_n^{\underline{i}}(n-s_n)^{\underline{j}}}{n^{\underline{i+j}}}$.

Para calcular cotas superiores de competitividades, también resultarán útiles las siguientes definiciones y el lema que sigue:

Definición 1.5 (Orden Montaña (Desequilibrada)) *Diremos que un orden σ de E es un orden montaña si los pesos de los elementos son crecientes hasta la aparición del elemento más pesado, es decir, $w(\sigma(i)) \leq w(\sigma(j))$ para todo $i \leq j \leq \sigma^{-1}(x^1)$, donde recordamos que $x^1 = \operatorname{argmax}_{x\in E} w(x)$. Diremos que σ es un orden montaña desequilibrada si es un orden montaña y además los elementos que llegan antes que el más pesado son los menos pesados de E, es decir, si $\sigma([\sigma^{-1}(x^1) - 1])$ es el conjunto de los $\sigma^{-1}(x^1) - 1$ elementos menos pesados de E.*

Notemos que si σ es un orden montaña desequilibrada con $\sigma^{-1}(x^1) = i$, entonces $\sigma(j) = x^{n-j+1}$ para todo $j \leq i$. En la Figura 1.2 se presentan ejemplos de órdenes montaña.

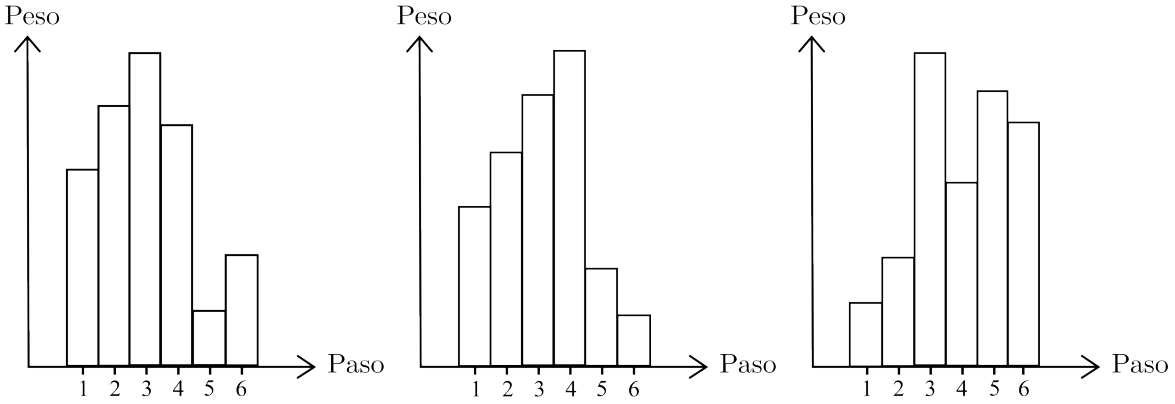

Figura 1.2: Órdenes Montaña. La columna sobre el paso i corresponde al elemento que llega en el paso i, donde la altura de la columna representa el peso del elemento. El gráfico de más a la derecha corresponde a un orden montaña desequilibrada, ya que los elementos antes del más pesado son los menos pesados de todo el conjunto.

Definición 1.6 (Problema de la Secretaria en Montaña (Desequilibrada)) *El Problema de la Secretaria en Montaña (resp. en Montaña Desequilibrada) corresponde al Problema de la Secretaria en el que el orden de llegada π es una variable aleatoria con distribución conocida por el algoritmo y tal que todo $\sigma \in \text{soporte}(\pi)$ es un orden montaña (resp. un orden montaña desequilibrada).*

Lema 1.7 *Sea π una variable aleatoria con valores en los órdenes montaña de E y sea $i := \text{argmax}_{j\in[n]} \mathbb{P}[\pi(j) = x^1]$ la posición más probable de x^1 en π. Entonces, el algoritmo que se salta los primeros $i-1$ elementos en llegar y selecciona el i-ésimo es óptimo para el Problema de la Secretaria en Montaña (sin Consejo) con orden de llegada π. Si π tiene valores solo en los órdenes montaña desequilibradas, entonces el algoritmo anterior es óptimo incluso en el modelo con Consejos Ilimitados*

Demostración. Busquemos el algoritmo óptimo para el problema. Notemos primero que basta considerar solo los algoritmos deterministas. En efecto, sea $\mathcal{A}$ un algoritmo aleatorio, es decir, $\mathcal{A}$ es una variable aleatoria con valores en los algoritmos deterministas. Usando probabilidades totales, la probabilidad de que $\mathcal{A}$ gane está dada por

$$\mathbb{P}[\mathcal{A} \text{ gana}] = \sum_{A \text{ alg determ}} \mathbb{P}[A \text{ gana}] \cdot \mathbb{P}[\mathcal{A} = A],$$

es decir, está dada por una combinación convexa de las probabilidades de que algoritmos deterministas ganen, con lo cual es claro que existe un algoritmo determinista que maximiza la probabilidad de ganar.

Consideremos un algoritmo determinista A, y supongamos sin pérdida de generalidad que A siempre devuelve un elemento de E (si rechaza los primeros $n-1$ elementos, selecciona el n-ésimo). Sea $\hat{\sigma} \in \Pi(E)$ definido como $\hat{\sigma}(i) = x^{n-i+1}$ para cada $i \in [n]$ (es decir, el orden creciente de pesos, partiendo en el elemento menos pesado), y sea k el paso en el que el algoritmo A se detiene si E llega en el orden $\hat{\sigma}$.

Sea π variable aleatoria con valores en los órdenes montaña de E. Sea σ una realización de π. Afirmamos que A gana si solo si $\sigma(k) = x^1$. En efecto, si $j := \sigma^{-1}(x^1) < k$, el algoritmo rechaza a x^1 en el paso j: en los primeros j pasos el algoritmo observa j elementos en orden creciente de pesos, al igual que en los primeros j pasos cuando A se enfrenta al orden $\hat{\sigma}$. Como consideramos solo algoritmos ordinales y A rechaza el j-ésimo elemento cuando el orden es $\hat{\sigma}$, sigue que A rechaza el j-ésimo elemento cuando el orden es σ, con lo cual A pierde (ver Figura 1.3). Por otro lado, si $i := \sigma^{-1}(x^1) > k$, el algoritmo selecciona el elemento que llega en el paso k: en los primeros k pasos el algoritmo observa k elementos en orden creciente de pesos, al igual que en los primeros k pasos cuando A se enfrenta al orden $\hat{\sigma}$. Como consideramos solo algoritmos ordinales y A selecciona el elemento k-ésimo cuando el orden es $\hat{\sigma}$, sigue que A selecciona el elemento k-ésimo cuando el orden es σ (ver Figura 1.4). Siguiendo el mismo razonamiento, es claro que A gana cuando $\sigma(k) = x^1$.

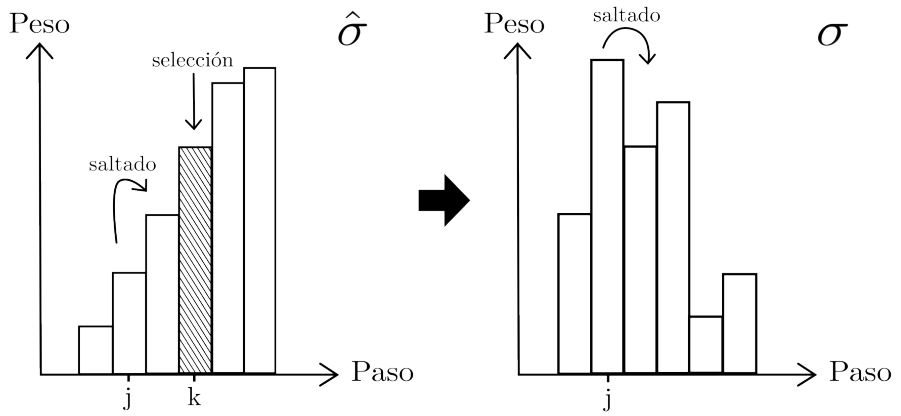

Figura 1.3: Si en $\hat{\sigma}$ (izquierda) se salta el elemento j-ésimo, en σ (derecha, donde $\sigma(j) = x^1$) también se debe saltar el elemento j-ésimo, porque un algoritmo ordinal no ve diferencias entre ambos órdenes hasta el paso j (solo vé que los pesos van creciendo).

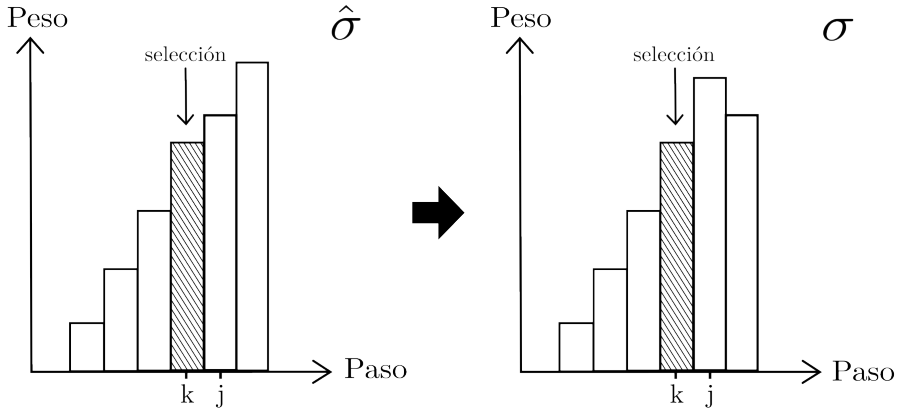

Figura 1.4: Si en $\hat{\sigma}$ (izquierda) se selecciona el elemento k-ésimo, en σ (derecha, donde $\sigma(j) = x^1$) también se debe seleccionar el elemento k-ésimo, porque un algoritmo ordinal no ve diferencias entre ambos órdenes hasta el paso k (solo ve que los pesos van creciendo).

Con lo anterior, tenemos que $\mathbb{P}[A \text{ gana}] = \mathbb{P}[\pi(k) = x^1]$. De esto, queda claro que, para todo algoritmo determinista A',

$$\mathbb{P}[A' \text{ gana}] \leq \max_{j \in [n]} \mathbb{P}[\pi(j) = x^1],$$

con lo cual se concluye la primera parte del lema.

Para la segunda parte del lema, notemos que en el caso de montaña desequilibrada y cuando $\pi(i) = x^1$, el oráculo de consejos entrega al algoritmo

$$\text{adv}_j = \begin{cases} n - j & \text{si } j < i, \\ 0 & \text{si } j \geq i \end{cases}$$

tras rechazar el elemento j-ésimo. Es decir, para $j < i$, después de haber rechazado j elementos, el oráculo solo informa que el elemento más pesado está entre los $n - j$ elementos

restantes, lo cual no ayuda al algoritmo. Por otro lado, para $j \geq i$, después de haber rechazado j elementos, el oráculo informa que el elemento más pesado está entre los rechazados, lo cual no ayuda al algoritmo, el cual no puede seleccionar un elemento que ya rechazó. Entonces, el oráculo no entrega ninguna información útil del orden antes de que se rechace el elemento más pesado. Se concluye entonces la segunda parte del lema. □

Este lema se usará para obtener cotas superiores de la competitividad de problemas que involucren órdenes adversariales (ya sean adversarios comunes, adversarios online o adversarios offline). La idea es que podemos hacer a esos problemas *más fáciles* para el algoritmo, reemplazando el orden adversarial por un orden que tenga una distribución conocida, a la cual llamaremos *distribución adversarial*. Es claro que el segundo problema es más fácil para el algoritmo, por lo cual una cota superior de la competitividad del segundo es también una cota superior de la competitividad del primero.

Los dos últimos lemas que se presentaron fueron desarrollados para esta tesis.

1.2. Resultados Conocidos

1.2.1. Problema con Orden Aleatorio y sin Consejo (RO)

Primero estudiamos el Problema de la Secretaria Clásico estudiado por Lindley y Dynkin, en el cual los elementos llegan en orden uniformemente aleatorio y no hay acceso al oráculo de consejos. El siguiente algoritmo se conoce como el algoritmo de Dynkin:

Algoritmo 1: Algoritmo de Dynkin

Parámetros: $p \in [0, 1]$.
$s \leftarrow \text{Bin}(n, p)$;
Fase de muestreo: rechazar primeros s elementos;
$\tau \leftarrow \text{máx}_{i \in [s]}\, w(x_i)$; `// calcular máximo peso actual`
Fase de selección: devolver primer x tal que $w(x) > \tau$.

Cuando $s = 0$ en el algoritmo anterior, usamos que $\text{máx}_{i \in \emptyset}\, w(x_i)$ es un peso menor que cualquiera que se vea en el futuro, o equivalentemente, en el caso $s = 0$ el algoritmo elige el primer elemento que observa.

Teorema 1.8 (Lindley [22] y Dynkin [12]) *Tomando $p = 1/e$, el Algoritmo 1 es $1/e$-competitivo y esta competitividad es óptima para el Problema de la Secretaria con Orden Aleatorio y sin Consejo* (RO).

Solo incluimos la demostración de que el algoritmo es $1/e$-competitivo, dado que la demostración de la optimalidad del algoritmo puede ser tediosa y no tiene gran relevancia para el desarrollo de esta tesis. Sí se asumirá la optimalidad del algoritmo para demostrar resultados posteriores.

Demostración. Llamemos A al Algoritmo 1. Mostraremos que, para $p \in (0,1]$, el algoritmo A gana con probabilidad mayor a $-p\ln(p)$. Maximizando $-p\ln(p)$ sobre $p \in (0,1]$, se obtiene que el algoritmo es $1/e$-competitivo con $p = 1/e$.

En vez de pensar que los elementos de E llegan en pasos discretos $i \in [n]$, podemos pensar equivalentemente que cada elemento $x \in E$ llega en un tiempo $t(x) \sim \text{Unif}([0,1])$, donde $\{t(y)\}_{y \in E}$ es un conjunto de variables aleatorias independientes. Podemos pensar que los primeros $s \sim \text{Bin}(n,p)$ elementos saltados en A corresponden a una muestra S que toma el algoritmo. Por el Lema 1.3, cada elemento de E se encuentra en S con probabilidad p. Viéndolo desde la perspectiva continua, los elementos de S corresponden a aquellos $x \in E$ tales que $t(x) \leq p$, i.e., los elementos que llegan antes del tiempo p. Entonces, lo que hace el algoritmo es saltarse los elementos que llegan antes que el tiempo p y luego elige el primer récord que aparezca después del tiempo p (si es que existe).

Sea $t^1 = t(x^1)$ el tiempo de llegada de x^1. Si $t^1 \leq p$, x^1 es rechazado y el algoritmo pierde. Si $t^1 > p$, el algoritmo elige a x^1 si y solo si es el primer récord que llega después del tiempo p, esto es, si y solo si el elemento más pesado que llega antes que x^1 llega antes del tiempo p o x^1 es el primer elemento en llegar. Sea $t \in (p,1]$. Condicionado a que x^1 llega en tiempo $t^1 = t$ y que existe un elemento x que llega antes que x^1, el tiempo de llegada de x distribuye uniforme en el intervalo $[0,t]$. En particular, el elemento más pesado que llega antes que x^1 distribuye uniforme en el intervalo $[0,t]$, por lo cual $\mathbb{P}[A \text{ gana} | t^1 = t,\ \exists x \in E :\ t(x) < t^1] = p/t$, porque esta es la probabilidad de que el elemento más pesado anterior llegue antes del tiempo p. Por otro lado, $\mathbb{P}[A \text{ gana} | t^1 = t,\ \forall x \in E :\ t(x) \geq t^1] = 1$, porque corresponde al caso en el que x^1 es el primer elemento en llegar. Para $t > p$, se tiene usando probabilidades totales que:

$$\begin{aligned}
\mathbb{P}[A | t^1 = t] &= (1-(1-t)^{n-1}) \cdot \mathbb{P}[A \text{ gana} | t^1 = t,\ \exists x \in E :\ t(x) < t^1] \\
&\quad + (1-t)^{n-1} \cdot \mathbb{P}[A \text{ gana} | t^1 = t,\ \forall x \in E :\ t(x) \geq t^1] \\
&= (1-(1-t)^{n-1}) \cdot \frac{p}{t} + (1-t)^{n-1} \\
&= \frac{p}{t} + (1-t)^{n-1} \cdot \left(1 - \frac{p}{t}\right) \\
&\geq \frac{p}{t},
\end{aligned}$$

donde en la primera igualdad se usa la independencia de los tiempos de llegada de los elementos para obtener que la probabilidad de que todos los elementos distintos de x^1 lleguen después de un tiempo t es $(1-t)^{n-1}$, y en la desigualdad usamos que el segundo sumando es no-negativo. Integrando en $t \in [0,1]$, obtenemos:

$$\begin{aligned}
\mathbb{P}[A \text{ gana}] &= \int_0^1 \mathbb{P}[A \text{ gana} | t^1 = t] dt \\
&= \int_p^1 \mathbb{P}[A \text{ gana} | t^1 = t] dt \\
&\geq \int_p^1 \left(\frac{p}{t}\right) dt \\
&= 1 \cdot \ln(1) - p\ln(p) \\
&= -p\ln(p).
\end{aligned}$$

Tomando $p = 1/e$, tenemos que A es $1/e$-competitivo. □

	n											
	1	2	3	4	5	6	7	8	9	10	11	12
t_n	0	0	1	1	2	2	2	3	3	3	3	4
q_n	1	0,5	0,5	0,4583	0,4333	0,4278	0,4143	0,4098	0,406	0,3987	0,3984	0,3955

Tabla 1.1: Tamaños de muestra óptimos y competitividades para el Problema Clásico con n secretarias, $n \in \{1, 2, ..., 12\}$ ([6]).

Resulta de interés preguntarse también por la competitividad del Problema de la Secretaria Clásico para valores de $n \in \mathbb{N}^*$ fijos (los valores de α^n_{RO}). Para $n \in \mathbb{N}^*$ y $t \in [0..n-1]$ fijos, consideremos el algoritmo A que se salta los primeros t elementos y elige el primer récord que llegue después del paso t. Si x^1 llega en el paso $i > t$, será seleccionado por el algoritmo si y solo si el elemento de mayor peso en E_{i-1} está en E_t. Condicionado a $x_i = x^1$ y al conjunto E_{i-1}, E_t es un subconjunto de E_i de tamaño t equiprobable, por lo cual:

$$\mathbb{P}[A \text{ gana} | x_i = x^1] = \frac{t}{i-1}.$$

Se tiene entonces que:

$$\begin{aligned}\mathbb{P}[A \text{ gana}] &= \sum_{i=1}^{n} \mathbb{P}[A \text{ gana} | x_i = x^1] \cdot \mathbb{P}[x_i = x^1] \\ &= \frac{t}{n} \sum_{i=t+1}^{n} \frac{1}{i-1}.\end{aligned}$$

Llamamos t_n al valor de $t \in [0..n-1]$ que maximiza la expresión anterior para $n \in \mathbb{N}^*$ fijo, y q_n al valor de $\mathbb{P}[A \text{ gana}]$ cuando $|E| = n$ y $t = t_n$:

$$q_n = \frac{t_n}{n} \sum_{i=t_n+1}^{n} \frac{1}{i-1} = \max_{t \in \{0,...,n-1\}} \frac{t}{n} \sum_{i=t+1}^{n} \frac{1}{i-1}.$$

Usando t_n como tamaño de muestra, el algoritmo descrito es óptimo para el Problema de la Secretaria Clásico con n candidatos, es decir, $\alpha^n_{\text{RO}} = q_n$; y se tiene que $q_n > 1/e$ para todo $n \in \mathbb{N}^*$ y que $(q_n)_n$ es una sucesión no-creciente que converge a $1/e$ (ver por ejemplo [6]). En la Tabla 1.1 mostramos los valores de t_n y q_n para los primeros valores de $n \in \mathbb{N}^*$.

Más adelante, aprovecharemos que $q_n > 1/e$ para obtener una mejor competitividad en el Problema de la Secretaria con Orden Aleatorio y Consejo Único.

1.2.2. Problema con Orden Adversarial y sin Consejo (AO), y Problema con Muestra Aleatoria, Orden Adversarial Online y sin Consejo (SBOnAO)

Mencionamos brevemente dos otros modelos, los cuales han sido estudiados previamente en la literatura. Primero, veamos el Problema de la Secretaria con Orden Adversarial y sin Consejo. Consideramos el siguiente algoritmo para el problema:

Algoritmo 2: Algoritmo de Selección Uniforme

$i \leftarrow \text{Unif}([n])$;
devolver x_i.

El siguiente resultado se considera folklore en la literatura:

Teorema 1.9 *El Algoritmo 2 gana con probabilidad* $1/n$ *y no existe algoritmo que gane con mayor probabilidad para el Problema de la Secretaria con Orden Adversarial y sin Consejo* (AO).

Es evidente que el Algoritmo 2, que elige un elemento al azar de manera uniforme entre los n elementos, elige a x^1 con probabilidad $1/n$. Se verá en la siguiente subsección que no existe algoritmo que gane con mayor probabilidad en el contexto más general en el que se le permite al algoritmo pedir consejos. Por la definición de competitividad dada, se tiene que $\alpha_{AO} = 0$. Este teorema es interesante porque nos indica que en orden adversarial se le debe dar información adicional al algoritmo (por ejemplo, una muestra aleatoria de los pesos) para ganar con probabilidad constante.

Segundo, mencionamos el Problema de la Secretaria con Muestra Aleatoria, Orden Adversarial Online y sin Consejo. Este problema se le ha llamado el Problema de la Secretaria Inconsciente del Orden (*Order Oblivious Secretary Problem* en inglés) [3]. Se tiene que el algoritmo óptimo para este problema corresponde al algoritmo de Dynkin con $p = 1/2$:

Teorema 1.10 (ver por ejemplo [18]) *Tomando* $p = 1/2$*, el Algoritmo 1 es* $1/4$*-competitivo y esta competitividad es óptima para el Problema de la Secretaria con Muestra Aleatoria, Orden Adversarial Online y sin Consejo* (SBOnAO).

Demostramos brevemente que el Algoritmo 1 gana con probabilidad al menos $1/4$. Se concluirá directamente que $\alpha_{\text{SBOnAO}} \leq 1/4$ cuando veamos que $\alpha_{\text{SBOffAO}} \leq 1/4$ en la siguiente sección.

Demostración. Para $p = 1/2$, en virtud del Lema 1.3, el Algoritmo 1 muestrea cada elemento de E con probabilidad $1/2$ de manera independiente y luego selecciona el primer récord fuera de muestra. Notemos que si x^1 se encuentra fuera de la muestra y x^2 se encuentra dentro de la muestra, entonces x^1 es el único récord fuera de muestra, por lo que el algoritmo lo elige. Como este evento ocurre con probabilidad $1/4$ (dado que $\mathbb{P}[x^1 \notin S, x^2 \in S] = \mathbb{P}[x^1 \notin S] \cdot \mathbb{P}[x^2 \in S]$), el algoritmo es $1/4$-competitivo. □

De la comparación de los modelos AO y SBOnAO notamos que basta que el jugador pueda observar una muestra aleatoria de los pesos para poder ganar con probabilidad constante. Sin embargo, de la comparación de los modelos SBOnAO y RO notamos que el jugador puede ganar con mayor probabilidad si además los elementos después de la muestra aleatoria llegan en orden aleatorio uniforme. En la sección siguiente veremos como la habilidad de obtener información adicional a partir del oráculo de consejos permite en algunos casos mejorar las garantías del jugador.

1.3. Resultados Nuevos

1.3.1. Problema con Orden Adversarial y Consejos ($\mathbf{AO_1}$ y $\mathbf{AO_\infty}$)

Retomamos ahora el Problema de la Secretaria con Orden Adversarial, pero ahora permitimos al jugador consultar al oráculo de consejos. Veremos que incluso cuando el jugador puede consultar al oráculo después de rechazar cada elemento, no puede ganar con probabilidad mayor a $1/n$.

Algoritmo: Algoritmo de Selección Uniforme (Algoritmo 2)

$i \leftarrow \text{Unif}([n])$;
devolver x_i.

Teorema 1.11 *El Algoritmo 2 gana con probabilidad* $1/n$ *y no existe algoritmo que gane con mayor probabilidad para el Problema de la Secretaria con Orden Adversarial y Consejos Ilimitados* (AO_∞).

Demostración. Previamente mencionamos que el Algoritmo 2 elige a x^1 con probabilidad $1/n$, falta ver que ningún algoritmo para el modelo gana con probabilidad mayor.

Consideraremos el problema más fácil en el que el orden de llegada π es una variable aleatoria con valores en los órdenes montaña desequilibrada, con distribución adversarial conocida por el algoritmo. Fijemos órdenes $\pi_i : [n] \to E$ para $i \in [n]$ tales que $\pi_i(j) := x^{n-j+1}$ si $j < i$ y $\pi_i(i) = x^1$. En palabras, π_i presenta primero los $i-1$ elementos menos pesados en orden creciente de pesos $(x^n, x^{n-1}, ..., x^{n-i+2})$, luego presenta a x^1, y finalmente presenta los demás elementos en un orden arbitrario. Consideremos π que toma como valor uno de los π_i de manera uniforme, esto es, si $\mathbb{P}[\pi = \pi_i] = 1/n$ para cada $i \in [n]$.

Así, nos estamos reduciendo al Problema de la Secretaria en Montaña Desequilibrada con orden π, luego, en virtud del Lema 1.7, ningún algoritmo (incluso con ilimitados consejos) gana con probabilidad mayor a $\max_{j\in[n]} \mathbb{P}[\pi(j) = x^1] = \max_{j\in[n]} \mathbb{P}[\pi = \pi_j] = 1/n$, con lo cual se concluye que el Algoritmo 2 es óptimo. □

Este teorema nos permite concluir que $\alpha_{\text{AO}} = \alpha_{\text{AO}_1} = \alpha_{\text{AO}_\infty} = 0$, confirmándonos que el oráculo de consejos por sí solo no permite mejorar la competitividad de los modelos si no se le permite al jugador anteriormente tomar una muestra aleatoria de los pesos.

1.3.2. Problema con Muestra Aleatoria, Orden Adversarial Online y Consejos ($\mathbf{SBOnAO_1}$ y $\mathbf{SBOnAO_\infty}$)

Estudiamos ahora la versión del Problema de la Secretaria con Muestra Aleatoria y Orden Adversarial Online en la que además permitimos al jugador consultar al oráculo de consejos. En la subsección anterior notamos que los consejos no le permitían al jugador obtener una mejor garantía al enfrentarse a un orden adversarial, en este caso observaremos que consultar

al oráculo después de la toma de muestra aleatoria le permite mejorar la competitividad de $1/4$ a $1/e$, pero cualquier consejo posterior no permite mejorar la garantía (intuitivamente, los consejos no entregan información nueva una vez iniciada la llegada adversarial de los elementos). Consideremos el siguiente algoritmo para el problema:

Algoritmo 4: Algoritmo de Muestra y Selección Uniforme de Récord

Parámetros: $p \in [0,1]$.
$s \leftarrow \text{Bin}(n,p)$;
Fase de muestreo: observar una muestra uniforme $E_s \subseteq E$ de tamaño s;
$\tau \leftarrow \text{máx}_{i\in[s]}\, w(x_i)$; `// calcular máximo peso actual`
$r \leftarrow \text{adv}_s$; `// recibir consejo del oráculo`
$i \leftarrow \text{Unif}([r])$;
Fase de selección: devolver el i-ésimo elemento x que satisfaga $w(x) > \tau$.

Veremos que se cumple lo siguiente:

Teorema 1.12 *Tomando $p = 1/e$, el Algoritmo 4 es $1/e$-competitivo y esta competitividad es óptima para el Problema de la Secretaria con Muestra Aleatoria, Orden Adversarial Online y Consejos Ilimitados* (SBOnAO_∞).

Demostración. Primero mostramos que, para $p \in (0,1]$, el Algoritmo 4 es $(-p\ln(p))$-competitivo. Tomando $p = 1/e$, se obtiene un algoritmo $1/e$-competitivo.

Notemos que r denota la cantidad de elementos fuera de la muestra que son más pesados que todos los elementos de la muestra. Por el Lema 1.3, se tiene que cada elemento de E se encuentra en la muestra de manera independiente con probabilidad p. Se tiene que r es una variable aleatoria con

$$\mathbb{P}[r = k] = \begin{cases} p(1-p)^k & \text{si } k \in \{0,1,...,n-1\}, \\ (1-p)^n & \text{si } k = n. \end{cases}$$

En efecto, si $k \in \{0,1,...,n-1\}$, $r = k$ si y solo si $\{x^1,...,x^k\}$ no intersecta con la muestra y x^{k+1} está en la muestra. Por otro lado, $r = n$ si y solo si ningún elemento fue muestreado.

Llamemos A al Algoritmo 4. Cuando $r = 0$, A ya perdió, dado que x^1 se encontraría en la muestra. Cuando $r > 0$, A selecciona un elemento al azar de manera uniforme del conjunto

$$B = \{x \in E \setminus E_s : w(x) > \max_{y\in E_s} w(y)\},$$

el cual tiene r elementos. Debido a que $r > 0$, se tiene que $x^1 \in B$. Sigue que

$$\mathbb{P}[A \text{ gana} | r = k] = \frac{1}{k}, \text{ para cada } k \in [n].$$

Usando las expresiones anteriores, acotamos la probabilidad de que A gane:

$$\begin{aligned}\mathbb{P}[A \text{ gana}] &= \sum_{k=1}^{n} \mathbb{P}[A \text{ gana} | r = k] \cdot \mathbb{P}[r = k] \\ &= p \sum_{k=1}^{n-1} \frac{(1-p)^k}{k} + \frac{(1-p)^n}{n} \\ &\geq p \sum_{k=1}^{\infty} \frac{(1-p)^k}{k} = -p \ln(p).\end{aligned}$$

En la desigualdad usamos el hecho que $\frac{(1-p)^n}{n} = p \sum_{k=n}^{\infty} \frac{(1-p)^k}{n} \geq p \sum_{k=n}^{\infty} \frac{(1-p)^k}{k}$, y en la última igualdad usamos la serie de Maclaurin de la función $f(t) = -\ln(1-t) = \sum_{k=1}^{\infty} t^k/k$. Con esto, tomando $p = 1/e$, tenemos que A es $1/e$-competitivo.

Notemos que mostramos que $\mathbb{P}[A \text{ gana}] \geq 1/e$ al tomar $p = 1/e$, pero no hemos demostrado $\mathbb{P}[A \text{ gana}] \leq 1/e$, luego no basta demostrar que A es el algoritmo óptimo para concluir que $\alpha_{\text{SBOnAO}_\infty} \leq 1/e$ (a priori, la competitividad del algoritmo podría ser mayor estricta que $1/e$). Para mostrar que $\alpha_{\text{SBOnAO}_\infty} \leq 1/e$, realizaremos una reducción al Problema de la Secretaria con Orden Aleatorio y sin Consejo (Problema de la Secretaria Clásico), el cual sabemos que tiene competitividad $\alpha_{\text{RO}} = 1/e$ (Teorema 1.8).

Sea $A = (A', \mathcal{D})$ un algoritmo para SBOnAO_∞ con $\alpha := \alpha_{\text{SBOnAO}_\infty}(A)$ su competitividad. Esto es, cuando A es aplicado en una instancia de n elementos, primero selecciona un tamaño de muestra s de la distribución $\mathcal{D}_n$, luego recibe y observa un subconjunto aleatorio $E_s \subseteq E$ de tamaño s, y finalmente ejecuta el algoritmo A' sobre el conjunto restante $E \setminus E_s$ que llega en algún orden elegido de manera adversarial. Sin pérdida de generalidad, asumimos que A' realiza una llamada al oráculo después de recibir la muestra, aprendiendo $r = \text{adv}_s$. Consideremos nuevamente el conjunto $B = \{x \in E \setminus E_s : w(x) > \max_{y \in E_s} w(y)\}$ de los r elementos más pesados fuera de muestra. Mientras A' corre, el conjunto B llega (no necesariamente todos juntos) en orden adversarial. Cada vez que llega un elemento fuera de B, el algoritmo no gana información alguna sobre los elementos de B; en particular, la respuesta del oráculo de consejos en estas iteraciones no cambia con respecto a la iteración anterior. Por lo tanto, podemos asumir que A' simplemente ignora todos los elementos fuera de B. Esto implica que A' se comporta como un algoritmo para el Problema de la Secretaria con Orden Adversarial y Consejos Ilimitados (AO_∞) sobre el conjunto B con r elementos. Usando el Teorema 1.11, concluimos que $\mathbb{P}[A \text{ gana} | r = k] \leq 1/k$ para cada $k \in [n]$.

Modifiquemos el algoritmo anterior variando el algoritmo post-muestra. Sea $A_1 = (A_1', \mathcal{D})$ en el que el algoritmo post-muestra A_1' parte realizando una llamada al oráculo de consejos (para computar $r = \text{adv}_s$ como antes), luego elige un número t del conjunto $[r]$ uniformemente al azar, y finalmente elige el t-ésimo elemento en llegar con peso mayor a todos los pesos en E_s. Entonces, A_1' retorna un elemento uniformemente al azar de B definido arriba, y luego

$\mathbb{P}[A_1 \text{ gana}|r = k] = 1/k$ para cada $k \in [n]$. Se tiene entonces que

$$\begin{aligned}\mathbb{P}[A_1 \text{ gana}] &= \sum_{k=1}^{n} \mathbb{P}[A_1 \text{ gana}|r = k] \cdot \mathbb{P}[r = k] \\ &\geq \sum_{k=1}^{n} \mathbb{P}[A \text{ gana}|r = k] \cdot \mathbb{P}[r = k] \\ &= \mathbb{P}[A \text{ gana}] \\ &= \alpha,\end{aligned}$$

donde usamos que la distribución de r (que es una variable aleatoria) es igual en ambos algoritmos, dado que solo depende de la distribución del tamaño de la muestra aleatoria. Sigue que el algoritmo A_1 tiene competitividad mayor o igual a α.

Por último, sea A_2 el siguiente algoritmo para el Problema de la Secretaria Clásico: Seleccione un número $s \sim \mathcal{D}_n$, observe y rechace los primeros s elementos en llegar, compute el valor umbral τ igual al máximo peso observado hasta el momento, y elija el primer elemento con peso mayor a τ. Es fácil ver que la probabilidad de que A_2 gane (bajo orden de llegada aleatorio) es igual a la probabilidad de que el algoritmo A_1 gane (bajo cualquier orden adversarial online). Entonces, la competitividad de A_2 satisface $\alpha \leq \alpha_{\text{RO}}(A_2) \leq \alpha_{\text{RO}} = 1/e$, con lo cual concluimos. □

Como el Algoritmo 4 solo realiza una llamada al oráculo de consejos, concluimos también que el resultado aplica para el Problema de la Secretaria con Muestra Aleatoria, Orden Adversarial Online y Consejo Único (SBOnAO_1), con lo cual $\alpha_{\text{SBOnAO}_1} = \alpha_{\text{SBOnAO}_\infty} = 1/e$.

Tras la demostración de la cota superior de $\alpha_{\text{SBOnAO}_\infty}$, no resulta ser extraño que la competitividad de este problema coincida con la del Problema de la Secretaria Clásico, ya que el Algoritmo 4 es básicamente una versión del algoritmo de Dynkin (Algoritmo 1) adaptada al modelo SBOnAO_1 . En efecto, el algoritmo de Dynkin también toma una muestra de tamaño $s \sim \text{Bin}(n, p)$ (si los elementos llegan en orden aleatorio, s elementos seguidos rechazados pueden ser considerados como una muestra aleatoria) y luego elige un elemento al azar de $B = \{x \in E \setminus E_s : w(x) > \text{máx}_{y \in E_s} w(y)\}$, la diferencia está en que el algoritmo de Dynkin elige un elemento al azar aprovechando que B llega en orden aleatorio (el primer elemento en llegar de B es un elemento al azar uniforme de B), mientras que el Algoritmo 4 usa la información del tamaño de B para elegir un elemento al azar uniforme de B. De hecho, argumentando correctamente, el análisis hecho de la competitividad del Algoritmo 4 sirve también para demostrar que el algoritmo de Dynkin es $1/e$-competitivo.

1.3.3. Problema con Muestra Aleatoria, Orden Adversarial Offline y sin Consejo (SBOffAO)

En esta subsección y en las dos siguientes tratamos los modelos con Muestra Aleatoria y Orden Adversarial Offline. Recordamos que en estos modelos el adversario primero selecciona un orden preliminar ψ, el jugador toma una muestra de tamaño definido por él, y luego el resto de los elementos llegan en el único orden que es consistente con ψ. En estos escenarios

el adversario tiene menos poder que en los modelos con Muestra Aleatoria y Orden Adversarial Online, en los cuales puede aprovechar la información de la muestra para ordenar los elementos fuera de esta.

Dicho esto, veremos primero que en ausencia de consejos no existen algoritmos con garantías mejores que la mejor garantía del caso con adversario online. Recordamos el algoritmo de Dynkin:

Algoritmo 5: Algoritmo de Dynkin

Parámetros: $p \in [0, 1]$.
$s \leftarrow \text{Bin}(n, p)$;
Fase de muestreo: rechazar primeros s elementos;
$\tau \leftarrow \text{máx}_{i \in [s]}\, w(x_i)$; `// calcular máximo peso actual`
Fase de selección: devolver primer x tal que $w(x) > \tau$.

Demostraremos lo siguiente:

Teorema 1.13 *Tomando $p = 1/2$, el Algoritmo 5 es $1/4$-competitivo y esta competitividad es óptima para el Problema de la Secretaria con Muestra Aleatoria, Orden Adversarial Offline y sin Consejo* (SBOffAO).

Demostración. Ya se demostró que el algoritmo es $1/4$-competitivo (Teorema 1.10), falta ver que $\alpha_{\text{SBOffAO}} \leq 1/4$. Para demostrarlo, consideraremos el problema más fácil en el que el orden preliminar ψ es una variable aleatoria que sigue una distribución adversarial conocida por el algoritmo.

Para cada $i \in [n]$, consideremos un orden fijo $\psi_i : [n] \to E$ tal que $\psi_i(j) = x^{i-j+1}$ para todo $j \leq i$, esto es, cualquier orden que presenta primero los i elementos más pesados en orden creciente de pesos $(x^i, x^{i-1}, ..., x^1)$ y luego presenta el resto de los elementos en cualquier orden. Supongamos que el orden preliminar ψ se selecciona de manera uniforme entre los órdenes preliminares ψ_i.

Sea A un algoritmo para el problema. Como para cada regla de selección post-muestra existe un tamaño de muestra óptimo fijo, supondremos sin pérdida de generalidad que A toma muestra de tamaño fijo s_n dependiente de n. Para visualizar la demostración, supongamos que los elementos de E se colocan en una línea de izquierda a derecha en el orden ψ (ver Figura 1.5). Exactamente s_n de los elementos serán marcados como "muestreados". Sea $\bar{x}$ el elemento no muestreado más a la izquierda en esta línea que además tenga peso mayor al de todos los elementos muestreados. Desde el punto de vista del algoritmo, $\bar{x}$ es el primer récord que aparece fuera de muestra. Sea J la variable aleatoria definida como $J = \psi^{-1}(\bar{x})$, la posición de $\bar{x}$ en el orden ψ si $\bar{x}$ existe.

Supongamos que A posee una habilidad adicional: además de la información ordinal recolectada durante su ejecución, puede acceder y usar el valor de J en cualquier momento después de haber tomado la muestra. Notemos que una cota superior para todas las competitividades de algoritmos $\tilde{A}$ con esta habilidad adicional es también una cota superior para todas las competitividades de algoritmos $\hat{A}$ sin esta habilidad, luego nos basta encontrar la cota superior para el modelo con la habilidad adicional.

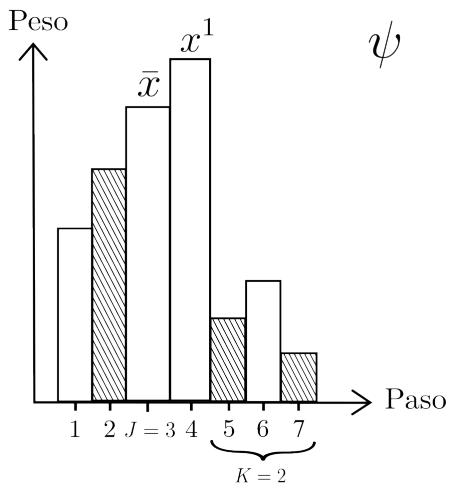

Figura 1.5: Visualización del orden ψ: la columna en la posición i del eje horizontal corresponde al elemento $\psi(i)$. Las columnas achuradas corresponden a los elementos muestreados. $\bar{x}$ corresponde al elemento menos pesado entre los que son más pesados que todos los achurados. J corresponde a la posición de $\bar{x}$ en esta figura y K corresponde a la cantidad de columnas achuradas a la derecha de x^1, el elemento más pesado.

Supongamos que ejecutamos el algoritmo A contra el orden ψ, que S es la muestra aleatoria, y que $x^1 \notin S$. Más aún, supongamos que A no selecciona elementos que no sean récords (esto porque solo nos interesa seleccionar x^1). Entonces, el primer elemento en el que A podría detenerse es $\bar{x}$. Por la definición de los ψ_i, partiendo de $\bar{x}$, el algoritmo verá una secuencia de una cantidad desconocida ℓ de elementos de pesos crecientes (todos récords) que terminará cuando llegue x^1, seguido por un no-récord. Aunque ℓ es desconocido para el algoritmo, se tiene que $\psi = \psi_i \Leftrightarrow i = \ell + J - 1$. Sea ϕ el orden de los elementos que llegan desde $\bar{x}$ en adelante. Lo dicho antes significa que ϕ toma valores en los órdenes montaña de los elementos fuera de la muestra. Entonces, al llegar a $\bar{x}$, el problema se reduce al Problema de la Secretaria en Montaña con orden ϕ y, por el Lema 1.7, la probabilidad de que A gane condicionada a la información obtenida hasta la llegada de $\bar{x}$ está acotada superiormente por $\max_{i\in[m]} \mathbb{P}[\phi(i) = x^1]$, donde m es la cantidad de elementos que quedan por llegar a partir de $\bar{x}$ (incluyéndolo).

La distribución de ϕ que el algoritmo deberá considerar corresponde a la distribución de ψ restringida a los elementos que llegan a partir de $\bar{x}$ y condicionada a toda la información ordinal obtenida antes de la llegada de $\bar{x}$ más el valor de J. Por la naturaleza ordinal del problema, es claro que la distribución de ϕ está totalmente determinada dadas dos cantidades: J y

$$K := |\{x \in S : \psi^{-1}(x) > \psi^{-1}(x^1)\}|,$$

la cantidad de elementos en la muestra que aparecen después de x^1 en el orden preliminar (ver Figura 1.5). En efecto, la información de qué elementos a la izquierda de $\bar{x}$ están dentro o fuera de la muestra es redundante al conocerse J, y lo único que se sabe de los elementos muestreados que están a la derecha de x^1 (todos de pesos menores a los de la izquierda) es su cantidad K.

Para $k \in \{0, ..., n-1\}$ y $j \in [n-k]$, mostraremos que el máximo de $\mathbb{P}[\psi = \psi_i | J = j, K = k]$ como función de i se alcanza en $i = j$. Computaremos primero $\mathbb{P}[J = j, K = k | \psi = \psi_i]$, para luego usar el teorema de Bayes. Condicionemos en $\psi = \psi_i$. Para que el evento $(J = j, K = k)$ pueda ocurrir, deben satisfacerse las siguientes condiciones:

- $j \leq i$: el primer récord no puede encontrarse después de x^1 en el orden preliminar
- $k \leq n - i$: hay $n - i$ elementos *a la derecha* de x^1, por lo que no se pueden muestrear más
- $k \leq s_n$: la cantidad de elementos muestreados *a la derecha* de x^1 es a lo más la cantidad total de elementos muestreados
- $s_n - k \leq j - 1$: se muestrean $s_n - k$ elementos *a la izquierda* de x^1, los cuales se encuentran en $\{\psi(1), ..., \psi(j-1)\}$

Con *a la izquierda* y *a la derecha* nos referimos a la izquierda y derecha de la Figura 1.5 descrita antes.

De aquí en adelante supongamos que i, j, k, n, s_n satisfacen las condiciones anteriores, es decir, el evento en que $J = j$, $K = k$ y $\psi = \psi_i$ tiene probabilidad positiva. El evento $J = 1$ condicionado a $\psi = \psi_i$ corresponde al evento en el que $\psi(1), \psi(2), ..., \psi(i) \notin S$, mientras que el evento $J = j$ con $j \geq 2$ condicionado a $\psi = \psi_i$ corresponde al evento en el que $\psi(j-1) \in S$ y $\psi(j), \psi(j+1), ..., \psi(i) \notin S$ (ver Figura 1.5). El evento $K = k$ corresponde al evento en el que exactamente k de los $n - i$ elementos a la derecha de x^1 aparecen en la muestra. Calculamos:

$$\mathbb{P}[J = j, K = k | \psi = \psi_i] = \begin{cases} \binom{n-i}{k} / \binom{n}{s_n} & \text{si } j = 1, \\ \binom{n-i}{k} \binom{j-2}{s_n-k-1} / \binom{n}{s_n} & \text{si } j \geq 2. \end{cases}$$

Lo anterior se obtiene dividiendo casos favorables por casos totales de la siguiente manera. En ambos casos, la cantidad de muestras posibles de E de tamaño s_n es $\binom{n}{s_n}$. En el caso $j = 1$, la muestra se encuentra contenida en el lado derecho de x^1 y hay $\binom{n-i}{k}$ muestras posibles, una por cada subconjunto de $\{x^{i+1}, ..., x^n\}$ de tamaño $k = s_n$, luego se obtiene lo escrito. En el caso $j \geq 2$, también hay $\binom{n-i}{k}$ intersecciones posibles de la muestra con el lado derecho de x^1, pero además hay que considerar la intersección de la muestra con el lado izquierdo de x^1. Como $\psi(j-1)$ está en la muestra y $\psi(j), \psi(j+1), ..., \psi(i)$ no están, quedan por colocar en la muestra $s_n - k - 1$ elementos del lado izquierdo de x^1, los cuales solo pueden ser seleccionados del conjunto $\{\psi(1), ..., \psi(j-2)\}$, con lo cual se agrega el factor $\binom{j-2}{s_n-k-1}$ y se obtiene lo escrito.

Usando el teorema de Bayes y el hecho que $\mathbb{P}[\psi = \psi_i] = 1/n$, calculamos:

$$\begin{aligned} \mathbb{P}[\psi = \psi_i | J = j, K = k] &= \frac{\mathbb{P}[J = j, K = k | \psi = \psi_i] \cdot \mathbb{P}[\psi = \psi_i]}{\mathbb{P}[J = j, K = k]} \\ &= \begin{cases} C_{njk} \binom{n-i}{k} [[j \leq i \leq n-k]] & \text{si } j = 1 \\ \tilde{C}_{njk} \binom{n-i}{k} [[j \leq i \leq n-k]] & \text{si } j \geq 2. \end{cases} \end{aligned}$$

donde $C_{njk}, \tilde{C}_{njk}$ son constantes que no dependen de i y $[[\cdot]]$ denota el corchete de Iverson, el cual vale 1 si el argumento es verdadero y 0 si es falso. Como $\binom{n-i}{k}$ decrece con i, el máximo de $\mathbb{P}[\psi = \psi_i | J = j, K = k]$ se alcanza cuando i tiene el menor valor posible sujeto a $i \geq j$, esto es, cuando $i = j$. Concluimos que la mejor decisión para A es seleccionar el primer récord fuera de muestra $\bar{x}$ que vea.

Hemos probado entonces que la competitividad del problema está acotada superiormente por la competitividad del algoritmo $\bar{A}$ que toma los valores óptimos de s_n, toma muestra de tamaño s_n y elige el primer récord fuera de muestra que observa. Si $\psi = \psi_1$, $\bar{A}$ gana si y solo si x^1 está fuera de la muestra; si $\psi \neq \psi_1$, el algoritmo gana si y solo si x^1 está fuera de la muestra y x^2 está dentro. Entonces:

$$\begin{aligned}
\alpha^n_{\text{SBOffAO}} &= \alpha^n_{\text{SBOffAO}}(\bar{A}) \\
&= \frac{1}{n} \cdot \mathbb{P}[\bar{A} \text{ gana} | \psi = \psi_1] + \frac{n-1}{n} \cdot \mathbb{P}[\bar{A} \text{ gana} | \psi \neq \psi_1] \\
&= \frac{1}{n} \cdot \frac{n - s_n}{n} + \frac{n-1}{n} \cdot \frac{s_n(n-s_n)}{n(n-1)} \\
&= \frac{s_n(n-s_n)}{n(n-1)} + \left(\frac{n-s_n}{n^2} - \frac{s_n(n-s_n)}{n^2(n-1)} \right),
\end{aligned}$$

donde el sumando entre paréntesis es $o(1)$. Usando el Lema 1.4, tenemos que:

$$\alpha_{\text{SBOffAO}} = \alpha_{\text{SBOffAO}}(\bar{A}) \leq \max_{p \in [0,1]} p(1-p) = \frac{1}{4},$$

máximo que se alcanza en $p = 1/2$. □

Dada la jerarquía de los modelos, lo anterior también demuestra que $\alpha_{\text{SBOnAO}} \leq 1/4$, lo cual teníamos pendiente. Tenemos así que $\alpha_{\text{SBOnAO}} = \alpha_{\text{SBOffAO}} = 1/4$.

1.3.4. Problema con Muestra Aleatoria, Orden Adversarial Offline y Consejos Ilimitados (SBOffAO$_\infty$)

Estudiamos ahora la versión del problema con adversario offline e ilimitados llamadas permitidas al oráculo de consejos. Curiosamente, al permitir consultas al oráculo en cada paso del proceso, los análisis para cotas superiores resultan más sencillos que en el caso con consejo único, porque podemos suponer sin pérdida de generalidad que los algoritmos consultan al oráculo en cada paso.

Consideremos el Algoritmo 6. En palabras, el algoritmo primero toma una muestra y luego consulta al oráculo de consejos y recibe r. Dependiendo del valor de r obtenido, el algoritmo sigue distintas reglas de selección. Esta idea no tendría sentido en el modelo con adversario online, en el cual el adversario ordena los elementos fuera de muestra adaptándose al valor de r. Sin embargo, antes de definir el orden preliminar ψ un adversario offline no sabe que valor de r se obtendrá, por lo cual no puede ordenar los elementos fuera de muestra de la manera más complicada dado r. Para hacer más cómodo el análisis, consideramos el Algoritmo 7, el

Algoritmo 6: Algoritmo de Múltiples Casos

Parámetros: $p, q, h \in [0, 1]$.
$s \leftarrow \text{Bin}(n, p)$;
Fase de muestreo: observar una muestra uniforme $E_s \subseteq E$ de tamaño s;
$\tau \leftarrow \text{máx}_{i \in [s]}\, w(x_i)$; `// calcular máximo peso actual`
$r \leftarrow \text{adv}_s$; `// recibir consejo del oráculo`
Fase de selección:
  caso $r \in \{1, 2\}$ **hacer devolver** primer x tal que $w(x) > \tau$;
  caso $r = 3$ **hacer**
    con probabilidad q: **devolver** primer x tal que $w(x) > \tau$;
    en otro caso
      rechazar primer x_t tal que $w(x_t) > \tau$;
      $\tau \leftarrow \text{máx}_{x \in E_t}\, w(x)$;
      $r \leftarrow \text{adv}_t$;
      si $r = 1$ **entonces devolver** primer x tal que $w(x) > \tau$;
      en otro caso devolver primer o segundo x tal que $w(x) > \tau$ con probabilidad $\frac{1}{2}$ cada uno;
  caso $r \geq 4$ **hacer**
    mientras $r \geq 3$ **hacer**
      rechazar cada elemento x_i y recalcular $r \leftarrow \text{adv}_i$ después de cada rechazo;
    fin
    $\tau \leftarrow \text{máx}_{x \in E_t}\, w(x)$, donde x_t fue el último elemento rechazado;
    si $r = 2$ **entonces**
      con probabilidad h: **devolver** primer x tal que $w(x) > \tau$;
      en otro caso devolver segundo x tal que $w(x) > \tau$;
    si $r = 1$ **entonces devolver** primer x tal que $w(x) > \tau$.

Algoritmo 7: Algoritmo de Múltiples Casos Modificado

si $n = 1$ **entonces**
  devolver único elemento $x \in E$;
si $n \in \{2, 3\}$ **entonces**
  Fase de muestreo: observar un único elemento aleatorio y;
  $\tau \leftarrow w(y)$;
  $r \leftarrow \text{adv}_1$;
  Fase de selección:
  si $r = 1$ **entonces devolver** único x con $w(x) > \tau$;
  si $r = 2$ **entonces devolver** primer o segundo x con $w(x) > \tau$ equiprobablemente;
si $n \geq 4$ **entonces**
  ejecutar Algoritmo 6.

cual separa el caso $n \leq 3$ del caso $n \geq 4$. Veremos que, efectivamente, el Algoritmo 7 tiene mejor garantía que el algoritmo óptimo del caso con adversario online (Algoritmo 4).

Teorema 1.14 *Tomando valores de p, q, h adecuados, el Algoritmo 7 es $(20\sqrt{10} - 29)/81$-competitivo ($\approx 0{,}4228$-competitivo) y óptimo para el Problema de la Secretaria con Muestra Aleatoria, Orden Adversarial Offline y Consejos Ilimitados* (SBOffAO$_\infty$).

Demostración. Si $n = 1$, el algoritmo gana siempre. Si $n = 2$, con probabilidad $1/2$ ocurre que x^2 está en la muestra y el algoritmo gana. Si $n = 3$, con probabilidad $1/3$ se tiene que x^2 está en la muestra y el algoritmo gana, mientras que con probabilidad $1/3$ el elemento x^3 está en la muestra y, condicionado a este evento, el algoritmo gana con probabilidad $1/2$, con lo cual el algoritmo gana con probabilidad $\frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{2}$. Concluimos así que el Algoritmo 7 gana con probabilidad mayor a $(20\sqrt{10} - 29)/81$ cuando $n \leq 3$.

Consideremos ahora $n \geq 4$ y sea A el Algoritmo 6. Primero mostraremos que A es $(20\sqrt{10} - 29)/81$-competitivo optimizando sobre los valores de $p, q, h \in [0, 1]$. Sea $r_s = \text{adv}_s$ con $s \sim \text{Bin}(n, p)$. Es claro que $\mathbb{P}[r_s = k] = p(1-p)^k$ para $k \in [3]$ y $\mathbb{P}[r_s \geq 4] = (1-p)^4$ (ver por ejemplo el argumento en la demostración del Teorema 1.12).

El desempeño del algoritmo depende únicamente del ordenamiento de los tres elementos más pesados (x^1, x^2, x^3), ya que en el caso $r_s \leq 3$ se ignoran los elementos que no son los tres más pesados, y en el caso $r_s \geq 4$ el algoritmo recalcula en cada paso r hasta que $r \leq 2$. Para $a, b, c \in \{x^1, x^2, x^3\}$ distintos, llamemos $((a, b, c))$ al conjunto de órdenes preliminares ψ tales que $\psi^{-1}(a) < \psi^{-1}(b) < \psi^{-1}(c)$, es decir, los órdenes preliminares en los que a aparece antes que b y b aparece antes que c.

Definimos $\eta(a, b, c) := \mathbb{P}[A \text{ gana} | \psi \in ((a, b, c))]$. Notemos que:

$$\begin{aligned}
\eta(a, b, c) &= \sum_{i=1}^{3} \mathbb{P}[A \text{ gana} | \psi \in ((a, b, c)), r_s = i] \cdot \mathbb{P}[r_s = i] \\
&\quad + \mathbb{P}[A \text{ gana} | \psi \in ((a, b, c)), r_s \geq 4] \cdot \mathbb{P}[r_s \geq 4] \\
&= \sum_{i=1}^{3} \mathbb{P}[A \text{ gana} | \psi \in ((a, b, c)), r_s = i] \cdot p(1-p)^i \\
&\quad + \mathbb{P}[A \text{ gana} | \psi \in ((a, b, c)), r_s \geq 4] \cdot (1-p)^4.
\end{aligned}$$

Afirmamos que la probabilidad que A gane ante cada $((a, b, c))$ está dada por las siguientes expresiones:

$$\begin{aligned}
\eta(x^1, x^2, x^3) &= \eta(x^1, x^3, x^2) = p(1-p) + p(1-p)^2 + qp(1-p)^3, \\
\eta(x^2, x^3, x^1) &= \eta(x^2, x^1, x^3) = p(1-p) + (1-q)p(1-p)^3 + (1-p)^4, \\
\eta(x^3, x^1, x^2) &= p(1-p) + p(1-p)^2 + \frac{1}{2}(1-q)p(1-p)^3 + h(1-p)^4, \\
\eta(x^3, x^2, x^1) &= p(1-p) + \frac{1}{2}(1-q)p(1-p)^3 + (1-h)(1-p)^4.
\end{aligned}$$

Analicemos cada una de las expresiones. La primera línea corresponde a los casos en que x^1 es el primer elemento en aparecer entre los tres más pesados. Si $r_s = 1$, A siempre gana.

Si $r_s = 2$, A elige el primer récord que ve, el cual es x^1, luego también gana. Si $r_s = 3$, A elige el primer récord que ve (de nuevo x^1) con probabilidad q, luego gana con probabilidad q. Si $r_s \geq 4$, A pierde, porque seguirá llamando al oráculo y no realizará una selección antes de observar $r \leq 2$, pero eso ocurre solo después de rechazar x^1. Esto prueba la primera línea.

La segunda línea corresponde a los casos en que x^2 es el primer elemento en aparecer entre los tres más pesados. Si $r_s = 1$, A siempre gana. Si $r_s = 2$, A elige el primer récord que ve (x^2) y pierde. Si $r_s = 3$, A gana si rechaza el primer récord (x^2), lo cual ocurre con probabilidad $(1 - q)$. Después de rechazar este primer récord, A llama al oráculo y recibe $r = 1$, tras lo cual A gana. Si $r_s \geq 4$, A gana, porque seguirá llamando al oráculo y no realizará una selección antes de observar $r \leq 2$, lo cual ocurre después de rechazar a x^2. Tras rechazar a x^2, el oráculo entrega $r = 1$ y luego A gana. Esto prueba la segunda línea.

En la tercera línea, nuevamente, si $r_s = 1$, A gana. Si $r_s = 2$, A elige el primer récord que ve, el cual es x^1 (notemos que x^3 apareció en la muestra en este caso), luego también gana. Si $r_s = 3$, para que A gane tiene que rechazar el primer récord que ve (x^3), lo cual ocurre con probabilidad $(1 - q)$; tras lo cual recibe $r = 2$ y gana con probabilidad $1/2$. Es decir, condicionado a $r_s = 3$, A gana con probabilidad $(1 - q)/2$. Si $r_s \geq 4$, A rechaza a x^3 y luego llama al oráculo, recibiendo $r = 2$. Con probabilidad h elige el siguiente récord, el cual es x^1, con lo cual A gana. Esto prueba la tercera línea.

En la cuarta línea, nuevamente, si $r_s = 1$, A gana. Si $r_s = 2$, A elige el primer récord que ve, el cual es x^2, luego A pierde. Si $r_s = 3$, para que A gane tiene que rechazar el primer récord que ve (x^3), lo cual ocurre con probabilidad $(1 - q)$; tras lo cual recibe $r = 2$ y gana con probabilidad $1/2$. Si $r_s \geq 4$, A rechaza a x^3 y luego llama al oráculo, recibiendo $r = 2$. Con probabilidad $(1 - h)$ rechaza el siguiente récord, el cual es x^2. Tras ese evento, A elige el siguiente récord, el cual es x^1, con lo cual gana. Esto prueba la cuarta línea.

Habiendo demostrado la exactitud de estas cuatro expresiones, continuamos con la demostración. Por la definición de competitividad:

$$\alpha_{\text{SBOffAO}\infty}(A) = \text{mín}\left(\left\{\eta(x^1, x^2, x^3), \eta(x^2, x^3, x^1), \eta(x^3, x^1, x^2), \eta(x^3, x^2, x^1)\right\}\right).$$

Es fácil de verificar que $\eta(x^2, x^3, x^1) \geq \eta(x^3, x^2, x^1)$ para cualquier valor de $p, q, h \in [0, 1]$, por lo cual se puede omitir $\eta(x^2, x^3, x^1)$ dentro del mínimo. Una técnica potencialmente buena para intentar maximizar ese mínimo (sobre $p, q, h \in [0, 1]$) corresponde a imponer $\eta(x^1, x^2, x^3) = \eta(x^3, x^1, x^2) = \eta(x^3, x^2, x^1)$. Con esto, obtendremos expresiones $q = q(p)$ y $h = h(p)$. Imponemos:

$$\begin{aligned} \eta(x^3, x^1, x^2) &= \eta(x^3, x^2, x^1) \\ \Leftrightarrow p(1-p)^2 + h(1-p)^4 &= (1-h)(1-p)^4 \\ \Leftrightarrow h &= \frac{1}{2} - \frac{p}{2(1-p)^2} \end{aligned}$$

Reemplazando este valor de h obtenemos:

$$\eta(x^3, x^1, x^2) = p(1-p) + \frac{1}{2}p(1-p)^2 + \frac{1}{2}(1-q)p(1-p)^3 + \frac{1}{2}(1-p)^4$$

Ahora imponemos:

$$\begin{aligned}\eta(x^1,x^2,x^3) &= \eta(x^3,x^1,x^2)\\ \Leftrightarrow p(1-p)^2+qp(1-p)^3 &= \frac{1}{2}p(1-p)^2+\frac{1}{2}(1-q)p(1-p)^3+\frac{1}{2}(1-p)^4\\ \Leftrightarrow q &= \frac{1-2p}{3p(1-p)}\end{aligned}$$

Reemplazando, tenemos que:

$$\eta(x^1,x^2,x^3)=\eta(x^3,x^1,x^2)=\eta(x^3,x^2,x^1)=p(1-p)+\frac{2}{3}p(1-p)^2+\frac{1}{3}(1-p)^3.$$

Maximizando la expresión anterior en $p\in[0,1]$, se obtiene $p=(4-\sqrt{10})/3\approx 0{,}2792$, $q=q(p)\approx 0{,}7312$, $h=h(p)\approx 0{,}2312$. Con tales valores, $\alpha_{\mathrm{SBOffAO}_\infty}(A)=(20\sqrt{10}-29)/81$. Cabe señalar que el desarrollo anterior no es una demostración rigurosa de que los valores obtenidos para p,q,h son los óptimos para el algoritmo A, pero veremos que lo son al mostrar la cota superior de la competitividad.

Para la cota superior de la competitividad, consideraremos el problema más fácil en el que el orden preliminar ψ es una variable aleatoria que sigue una distribución adversarial conocida por el algoritmo. Usaremos la siguiente notación: Para $a_1,...,a_i\in E$ distintos, escribimos $\phi=(a_1,...,a_i,\#)$ para decir que ϕ es un orden de E arbitrario que satisface $\phi(j)=a_j$ para $j\in[i]$. Definimos los siguientes órdenes de E usando esta notación:

$$\psi_1=(x^1,x^2,x^3,\#);\quad \psi_2=(x^3,x^1,x^2,\#);\quad \psi_3=(x^3,x^2,x^1,\#)$$

Supongamos que el orden preliminar ψ que selecciona el adversario corresponde a uno de ψ_1,ψ_2,ψ_3 elegido uniformemente al azar. Afirmamos que no existe algoritmo que seleccione a x^1 con probabilidad mayor a $(20\sqrt{10}-29)/81$ contra tal distribución adversarial en el límite $n\to\infty$.

En efecto, sea A un algoritmo para $\mathrm{SBOffAO}_\infty$. Sin pérdida de generalidad, suponemos que A pide consejo inmediatamente después de obtener la muestra y después de rechazar cada elemento. Suponemos sin pérdida de generalidad que A toma muestra de tamaño $s=s_n$ fijo dependiente de n (esto porque, para cada regla de selección post-muestra, existe un tamaño óptimo de muestra que maximiza la probabilidad de que el algoritmo gane). Sea $r_s=\mathrm{adv}_s$. Por definición de probabilidad, $\mathbb{P}[A\text{ gana}|r_s=1]\leq 1$. Verifiquemos que ante la distribución adversarial descrita, $\mathbb{P}[A\text{ gana}|r_s=2]\leq 2/3$ y $\mathbb{P}[A\text{ gana}|r_s\geq 3]\leq 1/3$.

Escribamos ϕ para referirnos al orden de los elementos de $\{x^1,x^2,x^3\}$ que están fuera de la muestra, es decir, el único orden de $\{x^1,x^2,x^3\}\setminus S$ que es consistente con ψ. Notemos que, cuando x^1 no está en la muestra, ϕ toma valores en los órdenes montaña desequilibrada: en los tres casos los elementos que aparecen antes que x^1 son los menos pesados (de $\{x^1,x^2,x^3\}\setminus S$) y estos aparecen en orden creciente de pesos. Entonces, cuando x^1 no está en la muestra, el problema se reduce después de la muestra al Problema de la Secretaria en Montaña Desequilibrada con orden ϕ. Condicionado a $r_s=2$, x^3 está en la muestra, y x^1 y x^2 están afuera, luego:

$$\begin{aligned}\mathbb{P}[\phi(1)=x^1|r=2]&=\mathbb{P}[\psi\in\{\psi_1,\psi_2\}]=\frac{2}{3},\\ \mathbb{P}[\phi(2)=x^1|r=2]&=\mathbb{P}[\psi=\psi_3]=\frac{1}{3}.\end{aligned}$$

Por la segunda parte del Lema 1.7 (sobre la variable aleatoria $(\phi|r = 2)$), se tiene que $\mathbb{P}[A \text{ gana}|r = 2] \leq 2/3$. Condicionado a $r \geq 3$, los elementos x^1, x^2 y x^3 están fuera de la muestra, luego:

$$\begin{aligned}\mathbb{P}[\phi(1) = x^1|r \geq 3] &= \mathbb{P}[\psi = \psi_1] = \frac{1}{3},\\ \mathbb{P}[\phi(2) = x^1|r \geq 3] &= \mathbb{P}[\psi = \psi_2] = \frac{1}{3},\\ \mathbb{P}[\phi(3) = x^1|r \geq 3] &= \mathbb{P}[\psi = \psi_3] = \frac{1}{3}.\end{aligned}$$

Por el Lema 1.7 (sobre la variable aleatoria $(\phi|r \geq 3)$), se tiene que $\mathbb{P}[A \text{ gana}|r \geq 3] \leq 1/3$. Así,

$$\begin{aligned}\alpha^n_{\text{SBOffAO}_\infty}(A) &\leq 1 \cdot \mathbb{P}[r_s = 1] + \frac{2}{3} \cdot \mathbb{P}[r_s = 2] + \frac{1}{3} \cdot \mathbb{P}[r_s \geq 3]\\ &= 1 \cdot \frac{s_n(n - s_n)}{n^{\underline{2}}} + \frac{2}{3} \cdot \frac{s_n(n - s_n)^{\underline{2}}}{n^{\underline{3}}} + \frac{1}{3} \cdot \frac{(n - s_n)^{\underline{3}}}{n^{\underline{3}}}.\end{aligned}$$

Usando el Lema 1.4 obtenemos:

$$\alpha_{\text{SBOffAO}_\infty}(A) \leq \max_{p\in[0,1]} \left(p(1 - p) + \frac{2}{3}p(1 - p)^2 + \frac{1}{3}(1 - p)^3 \right) = \frac{20\sqrt{10} - 29}{81}.$$

Como el algoritmo A es arbitrario, se concluye que $\alpha_{\text{SBOffAO}_\infty} \leq (20\sqrt{10} - 29)/81$. □

Este resultado es interesante primero porque $\alpha_{\text{SBOffAO}_\infty} > \alpha_{\text{SBOnAO}_\infty}$, lo que muestra que, con consejos, el problema con adversario offline es más fácil que el problema con adversario online, lo cual no se observa para el caso sin consejos, donde $\alpha_{\text{SBOffAO}} = \alpha_{\text{SBOnAO}}$.

1.3.5. Problema con Muestra Aleatoria, Orden Adversarial Offline y Consejo Único (SBOffAO_1)

Mencionamos brevemente la versión del problema con adversario offline con una única consulta permitida al oráculo de consejos. Este problema resulta ser difícil de analizar, siendo el único de los 12 modelos en el que no obtuvimos un valor exacto de la competitividad del problema. En particular, encontrar cotas superiores de la competitividad de este problema se hace más complicado que en otros casos, debido a que se requiere considerar el caso más general en el que el algoritmo puede pedir el consejo en un momento que no sea inmediatamente después de tomar la muestra (pedir consejo más tarde no es útil por ejemplo con adversario online, donde el adversario controla completamente el orden de aparición de los elementos fuera de muestra).

Por jerarquía de los modelos, tenemos que $\alpha_{\text{SBOnAO}_1} = 1/e \leq \alpha_{\text{SBOffAO}_1}$ (el Algoritmo 4 es $1/e$-competitivo en este modelo), pero hasta el momento de la escritura de esta tesis no sabemos si la desigualdad es estricta. Si la desigualdad fuera una igualdad, significaría que cambiar el adversario online por uno offline no le da oportunidad al jugador de mejorar su probabilidad de ganar, tal como se vio en el problema con adversario offline sin consejos

(SBOffAO). La dificultad de encontrar un algoritmo que gane con probabilidad mayor a $1/e$ nos hace conjeturar que $\alpha_{\text{SBOffAO}_1} = 1/e$.

Por otro lado, sabemos que $\alpha_{\text{SBOffAO}_1} \leq \alpha_{\text{SBOffAO}_\infty} = (20\sqrt{10}-29)/81$. Como el Algoritmo 6 consulta el oráculo de consejos más de una vez y no hemos hallado algoritmos con igual garantía que requieran menos consejos, creemos que $\alpha_{\text{SBOffAO}_1} < \alpha_{\text{SBOffAO}_\infty}$, lo cual marcaría otra diferencia con las versiones con adversarios online, donde el problema con un consejo es igual de difícil que el problema con ilimitados consejos.

1.3.6. Problema con Orden Aleatorio y Consejo Único (RO_1)

En esta subsección estudiamos la modificación del Problema de la Secretaria Clásico en la que se le permite al algoritmo hacer una única consulta al oráculo de consejos. Recordemos de la subsección 1.2.1 las definiciones:

$$q_n = \frac{t_n}{n} \sum_{i=t_n+1}^{k} \frac{1}{i-1} = \max_{t\in[n-1]} \frac{t}{n} \sum_{i=t+1}^{n} \frac{1}{i-1},$$

donde q_n resulta ser la competitividad del Problema de la Secretaria Clásico con n secretarias, competitividad que se obtiene al saltarse los primeros $t_n = \operatorname{argmax}_{t\in[n-1]} \sum_{i=t+1}^{n} \frac{t}{i-1}$ elementos en el algoritmo de Dynkin. El siguiente algoritmo resulta ser una mejora natural al algoritmo de Dynkin cuando se permite hacer una consulta al oráculo:

Algoritmo 8: Algoritmo de Dynkin con Doble Muestra

Parámetros: $p \in [0,1]$.
$s \leftarrow \text{Bin}(n,p)$;
Primera fase de muestreo: rechazar primeros s elementos;
$\tau \leftarrow \max_{i\in[s]} w(x_i)$; `// calcular máximo peso actual`
$r \leftarrow \text{adv}_s$; `// recibir consejo del oráculo`
Segunda fase de muestreo: rechazar primeros t_r elementos x tales que $w(x) > \tau$;
$j \leftarrow$ paso del último elemento rechazado;
$\tau \leftarrow \max_{i\in[j]} w(x_i)$;
Fase de selección: devolver primer x tal que $w(x) > \tau$.

En palabras, el algoritmo primero se salta s elementos, tras lo cual pedirá el consejo. Tras pedir el consejo y recibir $r = \text{adv}_s$, el problema se reduce al Problema de la Secretaria con r secretarias, luego en tal instante el jugador ejecuta el algoritmo de Dynkin óptimo para esa cantidad de secretarias. Tendremos el siguiente resultado:

Teorema 1.15 *Tomando $p \approx 0{,}2029$, el Algoritmo 8 es $\approx 0{,}4477$-competitivo y esta competitividad es óptima para el Problema de la Secretaria con Orden Aleatorio y Consejo Único (RO_1).*

Demostración. Primero mostramos que el Algoritmo 8 (llamémoslo A) es 0,4477-competitivo. En efecto, cuando $r > 0$, la probabilidad de ganar es q_r, dado que los elementos con peso

mayor a τ llegan en un orden uniformemente aleatorio. Por lo tanto, usando que la secuencia $(q_k)_{k\in\mathbb{N}^*}$ es no-creciente, obtenemos:

$$\mathbb{P}[A \text{ gana}] = \sum_{i=1}^{n} \mathbb{P}[A \text{ gana}|r=k] = p\sum_{i=1}^{n-1} q_i(1-p)^i + q_n(1-p)^n \geq p\sum_{i=1}^{\infty} q_i(1-p)^i.$$

Maximizando numéricamente la función más a la derecha sobre $p \in [0,1]$, obtenemos que $\mathbb{P}[A \text{ gana}] \geq 0{,}4477$ cuando $p \approx 0{,}2029$.

Para mostrar que el Algoritmo 8 es óptimo en el modelo RO_1, necesitamos verificar las siguientes dos afirmaciones que aplican para un algoritmo óptimo para el modelo RO_1:

1. Después de recibir $r = \mathrm{adv}_s$ para algún $s \in [0..n]$, el algoritmo no puede tener mejor garantía que el algoritmo de Dynkin (Algoritmo 1) con tamaño de muestra t_r.
2. Sin pérdida de generalidad, ningún elemento es seleccionado antes de que el algoritmo haya realizado la consulta al oráculo.

Para verificar la primera afirmación, notamos que después de recibir el consejo r, el problema se convierte en el Problema de la Secretaria Clásico con r elementos "buenos" (los elementos con pesos mayores al mayor peso en muestra) y $(n-s-r)$ elementos "malos". Lo mejor que puede hacer un algoritmo en esta situación es ignorar los elementos malos y ejecutar el algoritmo de Dynkin sobre el conjunto de elementos buenos; si no, podríamos tener un mejor algoritmo para el Problema de la Secretaria con r elementos, dado que podemos transformar la instancia del problema clásico con r elementos en una nueva instancia con $(n-s)$ elementos, sumando $(n-s-r)$ elementos de peso 0. Verificamos así la primera afirmación.

Para verificar la segunda afirmación, consideremos un algoritmo óptimo A_1 para el problema. Sea s el paso tras el cual el algoritmo pide consejo al oráculo. Siguiendo la primera afirmación, A_1 ejecuta el algoritmo de Dynkin saltándose t_{adv_s} elementos "buenos" después de pedir el consejo. Supongamos que A_1 tiene la posibilidad de elegir un récord antes de pedir el consejo, digamos, a partir del paso $i \leq s$. De argumentos estándar del problema clásico (ver por ejemplo [6]), si A_1 está dispuesto a seleccionar un récord en el paso i, entonces está dispuesto a elegir un récord en cualquier paso j con $i \leq j \leq s$. Además, sin pérdida de generalidad i, s no dependen de los pesos observados. Consideremos ahora un segundo algoritmo A_2, el cual no hace una selección antes de pedir el consejo, pide el consejo después de rechazar el elemento $(i-1)$-ésimo y luego ejecuta el algoritmo de Dynkin saltándose $t_{\mathrm{adv}_{i-1}}$ elementos "buenos". Veamos que $\alpha^n_{\mathrm{RO}_1}(A_2) \geq \alpha^n_{\mathrm{RO}_1}(A_1)$.

Consideremos σ el orden aleatorio uniforme de E. Si en σ no aparecen récords en los pasos en $[i,s]$ (es decir, si el máximo peso entre los primeros s elementos se encuentra entre las primeras $i-1$ posiciones), entonces A_1 y A_2 hacen exactamente lo mismo. En efecto, cuando no hay récords en los pasos en $[i,s]$, $\mathrm{adv}_{i-1} = \mathrm{adv}_s$. A_1 se saltará los primeros s elementos y luego aplicará el algoritmo de Dynkin saltándose adv_s elementos "buenos", mientras que A_2 se saltará los primeros $i-1$ elementos y luego aplicará el algoritmo de Dynkin saltándose adv_s elementos "buenos". Sin embargo, ninguno de los elementos "buenos" que se salta A_2

aparece en $[i, s]$, luego A_2 hace lo mismo que A_1. Consideremos ahora el caso en el que σ tiene un récord en $[i, s]$. Sea $h \in [n - i + 1]$ y condicionemos en $\text{adv}_{i-1} = h$. El primer récord que aparece en $[i, s]$ es el elemento más pesado con probabilidad $1/h$, luego A_1 gana con probabilidad $1/h$, mientras que A_2 gana con probabilidad $q_h \geq 1/h$. La desigualdad se debe a que el algoritmo de Dynkin sobre h elementos tiene mejor garantía que el algoritmo que selecciona un elemento al azar de manera uniforme entre los h elementos. Tenemos así que $\alpha^n_{\text{RO}_1}(A_2) \geq \alpha^n_{\text{RO}_1}(A_1)$. Por lo tanto, para cada algoritmo que pueda seleccionar un elemento antes de pedir consejo, existe uno al menos igual de bueno que no selecciona elementos antes del consejo. Verificamos así la segunda afirmación.

Consideremos ahora un algoritmo $\bar{A}$ óptimo para el problema. Sin pérdida de generalidad (por las afirmaciones demostradas), $\bar{A}$ primero rechaza una cantidad fija de elementos s_n, tras lo cual consulta el oráculo, recibiendo $r = \text{adv}_{s_n}$, y finalmente ejecuta el algoritmo de Dynkin saltándose t_r elementos "buenos". La competitividad de $\bar{A}$ para el problema con n secretarias es:

$$\begin{aligned}
\alpha^n_{\text{RO}_1}(\bar{A}) &= \sum_{i=1}^{n-s_n} q_i \cdot \mathbb{P}[r = i] \\
&= \sum_{i=1}^{n-s_n} q_i \cdot \mathbb{P}[\{x^1, x^2, ..., x^i\} \cap E_{s_n} = \emptyset,\ x^{i+1} \in E_{s_n+1}] \\
&= \sum_{i=1}^{n-s_n} q_i \cdot \frac{s_n (n - s_n)^{\underline{i}}}{n^{\underline{i+1}}}.
\end{aligned}$$

Como $\bar{A}$ es óptimo, se tiene que $(n - s_n) = \Omega(n)$ y $s_n = \Omega(n)$. Para ver lo primero, supongamos por absurdo que existe $\hat{n} \in \mathbb{N}$ tal que $s_{\hat{n}} > \hat{n} \cdot (1 - 1/e)$. En tal caso, el elemento más pesado aparece en la muestra con probabilidad mayor a $(1 - 1/e)$, por lo cual $\alpha^{\hat{n}}_{\text{RO}_1}(\bar{A}) < 1/e = \alpha_{\text{RO}} \leq \alpha_{\text{RO}_1}$, lo cual contradice la optimalidad de $\bar{A}$ para el problema RO_1. Entonces, para todo $n \in \mathbb{N}^*$, $(n - s_n) > n/e$, luego $(n - s_n) = \Omega(n)$. Para ver lo segundo, razonamos también por absurdo. Supongamos que $\liminf_{n\to\infty}(s_n/n) = 0$ (lo cual es equivalente a $s_n \neq \Omega(n)$). Entonces, se tendría que para cualquier $i \in \mathbb{N}^*$ fijo,

$$\liminf_{n\to\infty} \mathbb{P}[r = i] = \liminf_{n\to\infty} \frac{s_n (n - s_n)^{\underline{i}}}{n^{\underline{i+1}}} = 0.$$

Luego, para cualquier $j \in \mathbb{N}^*$ fijo:

$$\begin{aligned}
\alpha_{\text{RO}_1}(\bar{A}) &\leq \liminf_{n\to\infty} \alpha^n_{\text{RO}_1}(\bar{A}) \\
&= \liminf_{n\to\infty} \left(\sum_{i=1}^{\infty} q_i \cdot \mathbb{P}[r = i] \right) \\
&\leq \liminf_{n\to\infty} \left(\sum_{i=1}^{j-1} q_i \cdot \mathbb{P}[r = i] \right) + q_j \\
&\leq 0 + q_j = q_j,
\end{aligned}$$

donde en la segunda desigualdad usamos que la secuencia $(q_i)_i$ es no-creciente para acotar $\sum_{i=j}^{\infty} q_i \cdot \mathbb{P}[r = i] \leq q_j$. En particular, $\alpha_{\text{RO}_1}(\bar{A}) \leq q_5 = 0{,}4\bar{3}$, lo cual contradice la optimalidad de $\bar{A}$, ya que vimos que el Algoritmo 8 es $\approx 0{,}4477$-competitivo. Entonces, $s_n = \Omega(n)$.

Para simplificar la notación, supongamos que s_n/n converge (si no fuera así, bastaría intercambiar los límites de los desarrollos que vienen por límites inferiores, los cuales sí existen, y la demostración es completamente análoga) y sea $\hat{p} := \lim_{n\to\infty} s_n/n$. Definamos para cada $n, i \in \mathbb{N}^*$:

$$f_n(i) = q_i \cdot \frac{s_n(n-s_n)^{\underline{i}}}{n^{\underline{i+1}}} \cdot [[i \leq n - s_n]],$$
$$f(i) = q_i \cdot \hat{p}(1-\hat{p})^i,$$

donde $[[\cdot]]$ denota el corchete de Iverson. Notemos que, como $(n - s_n) = \Omega(n)$, para cada $i \in \mathbb{N}^*$, $f_n(i) \to f(i)$ cuando $n \to \infty$. Además:

$$\alpha^n_{\mathrm{RO}_1}(\bar{A}) = \sum_{i=1}^{\infty} f_n(i).$$

Notemos que si se tuviera que:

$$\lim_{n\to\infty} \sum_{i=1}^{\infty} f_n(i) = \sum_{i=1}^{\infty} f(i),$$

tendríamos que:

$$\alpha_{\mathrm{RO}_1}(\bar{A}) \leq \lim_{n\to\infty} \alpha^n_{\mathrm{RO}_1} = \lim_{n\to\infty} \sum_{i=1}^{\infty} f_n(i) = \sum_{i=1}^{\infty} q_i \cdot \hat{p}(1-\hat{p})^i \leq \max_{p\in[0,1]} \left(\sum_{i=1}^{\infty} q_i \cdot p(1-p)^i \right),$$

con lo cual concluiríamos que el Algoritmo 8 es óptimo.

El intercambio del límite con la serie no se puede hacer de manera directa, por lo cual haremos uso del teorema de convergencia dominada. Como $s_n = \Omega(n)$, debe existir $c \in (0,1)$ tal que $(n - s_n) \leq c \cdot n$. Definamos $g(i) = c^i$ para cada $i \in \mathbb{N}^*$. Veamos que $f_n(i) \leq g(i)$ para todo $i \in \mathbb{N}^*$ y para todo $n \in \mathbb{N}^*$ suficientemente grande. En efecto, para n suficientemente grande, se tiene que $s_n/(n-i) \leq 1$, porque $(n - s_n) = \Omega(n)$. Entonces, usando que $q_i \leq 1$, se tiene:

$$\begin{aligned} f_n(i) &= q_i \cdot \frac{s_n}{(n-i)} \cdot \frac{(n-s_n)^{\underline{i}}}{n^{\underline{i}}} \\ &\leq \frac{(cn)^{\underline{i}}}{n^{\underline{i}}} \\ &= \prod_{k=0}^{i-1} \frac{(cn-k)}{(n-k)} \\ &= c^i \prod_{k=0}^{i-1} \frac{(n-k/c)}{(n-k)} \\ &\leq c^i = g(i). \end{aligned}$$

Como $f_n(i) \leq g(i)$ para n suficientemente grande y además $\sum_{i=1}^{\infty} g(i) < \infty$, se tiene por el teorema de convergencia dominada que se puede realizar el intercambio del límite con la serie deseado, con lo cual concluimos. □

Notamos que este algoritmo se puede generalizar a más cantidad de consultas al oráculo de consejos: se parte tomando una muestra, se consulta al oráculo para reducir la cantidad de elementos buenos, luego se toma de nuevo muestra sobre los elementos buenos, se pide consejo nuevamente para reducir aún más el número de elementos buenos, y así sucesivamente.

1.3.7. Problema con Orden Aleatorio y Consejos Ilimitados (RO_∞)

Finalmente, en esta subsección permitimos ilimitados accesos al oráculo de consejos en el contexto de orden aleatorio. El algoritmo que proponemos resulta coincidir con la última idea planteada en el párrafo anterior, llevada al límite de ilimitados consejos: después de rechazar un elemento y recibir un consejo, si la cantidad de elementos buenos es mayor a 1, tomamos una nueva muestra de tamaño 1 y volvemos a consultar al oráculo, repitiendo hasta que el problema se haya reducido al problema con un solo elemento bueno:

Algoritmo 9: Algoritmo Primero después del Segundo

$r \leftarrow n$;
$i \leftarrow 1$;
mientras $r \neq 1$ **hacer**
  rechazar x_i;
  $r \leftarrow \mathrm{adv}_i$;
fin
$t \leftarrow$ paso del último elemento rechazado;
$\tau \leftarrow \max_{i \in [t]} w(x_i)$;
devolver primer x tal que $w(x) > \tau$.

Se tiene el siguiente resultado:

Teorema 1.16 *El Algoritmo 9 es* 1/2*-competitivo y esta competitividad es óptima para el Problema de la Secretaria con Orden Aleatorio y Consejos Ilimitados* (RO_∞)

Demostración. Es claro que el algoritmo selecciona a x^1 si y solo si x^2 llegó antes, lo cual ocurre con probabilidad 1/2 en modelos con orden aleatorio. Para verificar que el algoritmo es óptimo, es suficiente considerar el caso $n = 2$: x^1 es el primer elemento con probabilidad 1/2 y el segundo con probabilidad 1/2, por lo cual un algoritmo que elige el primer elemento con probabilidad p y el segundo con probabilidad $(1 - p)$ gana con probabilidad $p/2 + (1 - p)/2 \leq 1/2$. El oráculo es inútil en el caso $n = 2$, dado que no da información antes del rechazo del primer elemento y, tras rechazarse el primer elemento, elegir el segundo es la única opción que queda. □

La competitividad obtenida es estrictamente mayor que la del resto de los modelos que estudiamos antes. Sin embargo, esta competitividad es menor o igual a la de los problemas con información distribucional conocida para cada elemento. Por ejemplo, en el Problema de Desigualdad del Profeta, cada elemento tiene un peso positivo que es la realización de una variable aleatoria con distribución conocida, e incluso cuando los elementos llegan en orden adversarial se logra una competitividad de 1/2 [20]. Esto es notable, porque nos muestra que

el problema más fácil con consejos ordinales es igual de difícil que el problema más difícil con información distribucional.

Capítulo 2

Problema de la Secretaria Matroidal

En este capítulo estudiamos algunas variaciones del Problema de la Secretaria Matroidal que se obtienen al considerar distintos órdenes de llegada de los elementos y permitiendo al jugador consultar a un oráculo de consejo. A diferencia de lo realizado en el capítulo anterior en el problema con selección única, aquí consideraremos dos tipos de oráculos que entregan distinta cantidad de información ordinal.

En la primera sección damos definiciones y propiedades generales asociadas a matroides, definimos el problema a considerar, y describimos los oráculos de consejos. En la segunda sección mencionamos algunos resultados que se han obtenido previamente para el problema de la secretaria matroidal general y mencionamos algoritmos para clases específicas de matroides. En la tercera sección estudiamos tres modelos con consejos, los cuales no se han estudiado previamente, y presentamos algoritmos con competitividades no nulas para cada uno.

A lo largo de este capítulo, usaremos las notaciones introducidas en el capítulo anterior.

2.1. Preliminares

2.1.1. Definiciones y Propiedades Básicas de Matroides

Iniciamos el capítulo definiendo las matroides, estructura con la cual estaremos trabajando durante todo el capítulo. El concepto de matroide surge del intento de generalizar la idea de independencia lineal de vectores (por ejemplo, en $\mathbb{R}^m$) y, como veremos, permite generalizar varias definiciones y propiedades. En adelante, usaremos la siguiente notación: para I conjunto y x elemento, denotamos $I + x := I \cup \{x\}$ y $I - x := I \setminus \{x\}$.

Definición 2.1 (Matroide) *La tupla $M = (E, \mathcal{I})$ se dice matroide si E es un conjunto finito y $\mathcal{I} \subseteq 2^E$ es una colección que satisface los dos siguientes axiomas:*

1. *Si $I \subseteq J$ y $J \in \mathcal{I}$, entonces $I \in \mathcal{I}$.*
2. *Si $I, J \in \mathcal{I}$ y $|I| > |J|$, entonces existe $x \in I \setminus J$ tal que $J + x \in \mathcal{I}$.*

El conjunto E se denomina el conjunto de referencia de M y los elementos de $\mathcal{I}$ se denominan los (conjuntos) independientes de M. Un conjunto $B \in \mathcal{I}$ se dice base de M si es un conjunto independiente maximal con respecto a la relación de inclusión.

A continuación, damos tres ejemplos de matroides de uso frecuente:

Definición 2.2 (Matroide Lineal) *Sea V espacio vectorial, $E \subseteq V$ subconjunto finito e $\mathcal{I} = \{I \subseteq E : I$ es linealmente independiente$\}$. $M = (E, \mathcal{I})$ es la matroide lineal asociada a los vectores E del espacio vectorial V.*

Definición 2.3 (Matroide Gráfica) *Sea $G = (V, E)$ grafo e $\mathcal{I} = \{I \subseteq E : (V, I)$ es acíclico$\}$. $M = (E, \mathcal{I})$ es la matroide gráfica asociada al grafo G.*

Definición 2.4 (Matroide Transversal) *Sea $G = (V, E)$ grafo bipartito con partición $V = L \cup R$. Sea $\mathcal{I} = \{I \subseteq L :$ existe un matching en G que cubre a $I\}$. $M = (L, \mathcal{I})$ es la matroide transversal asociada al grafo bipartito G con lado izquierdo L.*

No veremos que las tres definiciones anteriores corresponden efectivamente a matroides. Se tiene la siguiente propiedad en matroides:

Proposición 2.5 *Si B_1 y B_2 son bases de una matroide $M = (E, \mathcal{I})$, entonces $|B_1| = |B_2|$.*

Demostración. Supongamos que $|B_1| \neq |B_2|$, sin pérdida de generalidad, $|B_1| < |B_2|$. Entonces, por la propiedad 2) de la definición de matroides, existe $x \in B_2 \setminus B_1$ tal que $B_1 + x \in \mathcal{I}$, lo cual contradice la maximalidad de B_1. Concluimos así que $|B_1| = |B_2|$. □

La proposición anterior motiva la siguiente definición:

Definición 2.6 (Rango) *El rango de una matroide $M = (E, \mathcal{I})$, denotado* $\mathrm{rank}(M)$*, se define como el cardinal de una base de M. Para $I \subseteq E$, definimos* $\mathrm{rank}_M(I)$ *como el cardinal de un independiente subconjunto de I maximal para la inclusión.*

Esta definición se hacía para conjuntos de vectores en un espacio vectorial antes de la definición de matroides y resulta natural también para otras matroides conocidas previamente. A modo de ejemplo, el rango de una matroide gráfica M asociada a un grafo $G = (V, E)$ corresponde a $|V| - cc(G)$, donde $cc(G)$ es la cantidad de componentes conexas de G. Así como un conjunto de vectores *generan* un subespacio vectorial, podemos definir:

Definición 2.7 (Conjunto generado) *Sea $M = (E, \mathcal{I})$ matroide. Para $I \subseteq E$, el conjunto generado por I se define como el conjunto*

$$\mathrm{span}(I) = \{x \in E : \mathrm{rank}(I + x) = \mathrm{rank}(I)\}.$$

Decimos que $x \in E$ es generado por $I \subseteq E$ si $x \in \mathrm{span}(I)$.

Es claro que las funciones $\mathrm{rank}_M : 2^E \to \{0, ..., \mathrm{rank}(M)\}$ y $\mathrm{span} : 2^E \to 2^E$ son monótonas, esto es, $\mathrm{rank}_M(I) \leq \mathrm{rank}(J)$ y $\mathrm{span}(I) \subseteq \mathrm{span}(J)$ si $I \subseteq J$.

Imaginemos que queremos construir mediante un algoritmo un conjunto independiente I de una matroide M. Si en un instante del proceso tenemos un conjunto independiente I' y queremos agregar un elemento $x \notin I'$ sin sacar elementos de I', podremos hacerlo si $x \notin \text{span}(I')$.

En el contexto de problemas de optimización con restricciones matroidales, si tenemos una función de pesos $w : E \to \mathbb{R}_+$, definiremos de la manera natural el peso de un subconjunto de E:

Definición 2.8 (Extensión función de pesos) *Si $w : E \to \mathbb{R}_+$ es una función de pesos, se define su extensión como $w : 2^E \to \mathbb{R}_+$, donde $w(I) = \sum_{x \in I} w(x)$ para cada $I \subseteq E$.*

Consideremos el siguiente algoritmo que construye una base de una matroide M:

Algoritmo 10: Algoritmo Glotón General

Entrada: Matroide $M = (E, \mathcal{I})$, $\sigma : [|E|] \to E$ orden de E.
Salida: Base ALG de M.
ALG $\leftarrow \emptyset$;
para $i \in [|E|]$ **hacer**
  $x \leftarrow \sigma(i)$;
  si ALG $+x \in \mathcal{I}$ **entonces** ALG $\leftarrow$ ALG $+x$;
fin
devolver ALG.

Para ver que el conjunto devuelto por el algoritmo es una base de M, basta notar que ALG $\in \mathcal{I}$ por construcción y que si ALG no fuera maximal, entonces existiría $x \in E \setminus$ ALG tal que ALG $+x \in \mathcal{I}$ que se debió haber agregado cuando apareció antes.

El algoritmo glotón general (Algoritmo 10) resulta ser muy útil al estudiar problemas de optimización con restricciones matroidales. Llamemos GLOTON(M, σ) a la salida del Algoritmo 10. Las dos siguientes propiedades son resultados clásicos de matroides y se usarán a lo largo del capítulo:

Proposición 2.9 *Sea $M = (E, \mathcal{I})$ matroide, $\sigma : [n] \to E$ orden de E y $x \in E$. Se tiene que $x \in$ GLOTON(M, σ) si y solo si $x \notin \text{span}(\{z \in E : \sigma^{-1}(z) < \sigma^{-1}(x)\})$, es decir, si y solo si x no es generado por el conjunto de elementos que llegan antes que x en el orden σ.*

Demostración. Para cada $i \in [n]$, recordamos que $E_i := \sigma([i])$ es el conjunto de elementos que llegan hasta el paso i (inclusive). Llamemos $G =$ GLOTON(M, σ) y $G_i = G \cap E_i$, lo que ha agregado el algoritmo glotón hasta el paso i. Afirmamos primero que $\text{span}(G_i) = \text{span}(E_i)$ para cada $i \in [n]$.

En efecto, por monotonía de span$(\cdot)$ se tiene que $\text{span}(G_i) \subseteq \text{span}(E_i)$. Supongamos por absurdo que $\text{span}(G_i) \neq \text{span}(E_i)$, luego existe $z \in \text{span}(E_i) \setminus \text{span}(G_i)$. Por definición del conjunto generado, se tiene que:

$$\text{rank}(E_i + z) = \text{rank}(E_i)$$
$$\text{rank}(G_i + z) > \text{rank}(G_i)$$

Por monotonía, $\text{rank}(E_i + z) \geq \text{rank}(G_i + z)$, luego, combinando expresiones, tenemos que $\text{rank}(E_i) > \text{rank}(G_i)$. Esto nos dice que existe $I \subseteq E_i$ tal que $I \in \mathcal{I}$ y $|I| > |G_i|$. Entonces, por el segundo axioma de matroides, existe $x \in E_i \setminus G$ tal que $G_i + x \in \mathcal{I}$, lo cual implica que x debió haber sido agregado por el algoritmo glotón, lo cual es una contradicción.

Se tiene entonces que $\text{span}(G_i) = \text{span}(E_i)$ para cada $i \in [n]$. Sea $j = \sigma^{-1}(x)$. Concluimos con lo siguiente:

$$x \in G \Leftrightarrow G_{j-1} + x \in \mathcal{I} \Leftrightarrow x \notin \text{span}(G_{j-1}) \Leftrightarrow x \notin \text{span}(E_{j-1}),$$

donde las primeras dos equivalencias se tienen del hecho que G_{j-1} corresponde al valor de ALG justo antes de procesar x. □

Proposición 2.10 *Sea $M = (E, \mathcal{I})$ matroide, $w : E \to \mathbb{R}_+$ función de peso inyectiva y $\sigma_w : [n] \to E$ el orden de E decreciente en peso w. Entonces, la única base de peso máximo de M es $\text{OPT} = \text{GLOTON}(M, \sigma_w)$. Más aún, para $x \in E$, $x \in \text{OPT}$ si y solo si x no está generado por el conjunto de elementos más pesados que x.*

Demostración. Para cada $i \in [n]$, recordamos que $E_i := \sigma_w([i]) = E^i$ es el conjunto de elementos que llegan hasta el paso i (inclusive), que en este caso corresponde también al conjunto de los i elementos más pesados de E. Llamemos $G = \text{GLOTON}(M, \sigma_w)$, $G_i = G \cap E_i$ y $\text{OPT}_i = \text{OPT} \cap E_i$. Realizamos el siguiente cálculo:

$$\begin{aligned}
w(G) &= \sum_{i=1}^{n} [[x^i \in G]] \cdot w(x^i) \\
&= \sum_{i=1}^{n} (|G_i| - |G_{i-1}|) w(x^i) \\
&= \sum_{i=1}^{n} |G_i| (w(x^i) - w(x^{i+1})) \\
&= \sum_{i=1}^{n} \text{rank}(E_i) (w(x^i) - w(x^{i+1})) \\
&\geq \sum_{i=1}^{n} |\,\text{OPT}_i\,| (w(x^i) - w(x^{i+1})) \\
&= \sum_{i=1}^{n} (|\,\text{OPT}_i\,| - |\,\text{OPT}_{i-1}\,|) w(x^i) \\
&= \sum_{i=1}^{n} [[x^i \in \text{OPT}]] \cdot w(x^i) \\
&= w(\text{OPT}).
\end{aligned}$$

En el cálculo anterior, $[[\cdot]]$ denota el corchete de Iverson. Además, se definen $G_0 = \text{OPT}_0 = \emptyset$ y $w(x^{n+1}) = 0$. Las primeras tres y las últimas tres igualdades son directas. En la cuarta igualdad usamos que $|G_i| = \text{rank}(E_i)$, lo cual se obtiene del hecho que

G_i es una base de E_i. Por último, en la desigualdad usamos que $\text{rank}(E_i) \geq \text{OPT}_i$, lo cual se tiene porque $\text{OPT}_i \in \mathcal{I}$ y $\text{OPT}_i \subseteq E_i$.

Por definición de OPT y del hecho que $G \in \mathcal{I}$, tenemos que $w(G) \leq w(\text{OPT})$, luego $w(G) = w(\text{OPT})$. Como existe un único independiente de peso máximo (bajo la hipótesis de inyetividad de w), concluimos que $\text{OPT} = G$.

Para ver lo segundo, basta usar la Proposición 2.9 con orden $\sigma = \sigma_w$, más lo recién demostrado. □

La última proposición nos muestra lo fácil que es resolver el problema de encontrar el independiente de peso máximo de una matroide cuando se conoce de antemano la función de pesos, siempre que podamos evaluar fácilmente si un conjunto es independiente o no. En los problemas que estudiaremos en este capítulo, la función de pesos será revelada de manera online como en el capítulo anterior, por lo cual la ejecución del algoritmo glotón no se podrá realizar de igual manera.

2.1.2. Definición del Problema

En esta subsección definimos el Problema de la Secretaria Matroidal. La definición del problema se hace asumiendo que el lector leyó la definición del Problema de la Secretaria Simple en el capítulo 1. Los órdenes de llegada a considerar serán los mismos que se presentaron en ese capítulo.

En el Problema de la Secretaria Matroidal, el conjunto de candidatos E es el conjunto de referencia de una matroide $M = (E, \mathcal{I})$. Consideraremos la versión del problema en la cual el jugador conoce de antemano M, es decir, asumimos que puede evaluar si $I \in \mathcal{I}$ o no para cada $I \subseteq E$ (el tiempo que toma realizar esto no nos interesa en esta tesis). El jugador inicia el proceso con un conjunto ALG inicialmente vacío. Nuevamente, un adversario asigna $w : E \to W$ biyectiva con $(W, \leq)$ un orden total. Los elementos de E llegan de uno en uno en un orden π (o un subconjunto llega primero como muestra aleatoria) y los pesos de los elementos se van revelando tras la llegada de cada uno. Cuando se revela que el elemento $x \in E$ tiene peso $w(x)$, el jugador debe decidir si agregar x a ALG o no. Si se agrega, x queda en ALG durante todo el resto del proceso, mientras que si se rechaza, el jugador pierde irrevocablemente la oportunidad de agregarlo en el futuro. Durante todo el proceso, ALG debe ser un conjunto independiente de la matroide y ALG al final del proceso corresponde a la solución que devuelve el algoritmo.

Para evaluar qué tan bueno es un algoritmo para uno de estos problemas, tenemos que definir la competitividad de manera más general. Consideremos primero la versión cardinal del Problema de la Secretaria Matroidal, la cual corresponde al caso en que $w : E \to W \subseteq \mathbb{R}_+$. En este caso, tiene sentido definir $w(I)$ para $I \subseteq E$ como en la Definición 2.8. Para $\alpha \in [0, 1]$ decimos que un algoritmo es α-competitivo para un modelo del Problema de la Secretaria Matroidal Cardinal si $\mathbb{E}[w(\text{ALG})]/w(\text{OPT}) \geq \alpha$, donde $\text{ALG} \in \mathcal{I}$ es la solución que devuelve el algoritmo y $\text{OPT} \in \mathcal{I}$ es la base de peso máximo de M, la cual se puede obtener con el algoritmo glotón al conocerse w (Proposición 2.10). Esta definición de competitividad del

problema cardinal es la que propone Babaioff al introducir el problema en [5]. Por la misma Proposición 2.10, OPT depende únicamente del orden de los pesos de E y no de los valores reales específicos que tienen, por lo cual se puede extender la definición de competitividad al problema ordinal que nos interesa. Más aún, tiene sentido definir OPT incluso cuando $w : E \to W$ en el caso ordinal, tomándolo como $\text{OPT} := \text{GLOTON}(M, \sigma_w)$. Pensamos ahora en el caso ordinal $w : E \to W$. Las siguientes dos definiciones se hacen por ejemplo en [25].

Definición 2.11 (Ordinal-Competitividad) *Sea $\mathcal{M}$ un modelo del Problema de la Secretaria Matroidal con matroide $M = (E, \mathcal{I})$, A un algoritmo para este modelo, $\alpha \in [0, 1]$ y $w : E \to W$ asignada adversarialmente. Decimos que A es α-ordinal-competitivo para el modelo $\mathcal{M}$ si $\mathbb{E}[\tilde{w}(\text{ALG})]/\tilde{w}(\text{OPT}) \geq \alpha$ para cada $\tilde{w} : E \to \mathbb{R}_+$ inyectiva consistente con w (esto es, para cada $\tilde{w}$ tal que $\tilde{w}(x) < \tilde{w}(y)$ si y solo si $w(x) < w(y)$), donde la esperanza se toma considerando la aleatoriedad de A y las condiciones del modelo $\mathcal{M}$.*

La ordinal-competitividad de un algoritmo y de un modelo se definen a partir de esta definición análogamente a cómo se hace en la Definición 1.1. Damos a continuación otra definición de competitividad que es más fuerte que la anterior:

Definición 2.12 (Probabilidad-Competitividad) *Sea $\mathcal{M}$ un modelo del Problema de la Secretaria Matroidal con matroide $M = (E, \mathcal{I})$, A un algoritmo para este modelo y $\alpha \in [0, 1]$. Decimos que A es α-probabilidad-competitivo para el modelo $\mathcal{M}$ si $\mathbb{P}[\bar{x} \in \text{ALG}] \geq \alpha$ para $\bar{x} \in \text{OPT}$, donde la probabilidad se toma considerando la aleatoriedad de A y las condiciones del modelo $\mathcal{M}$.*

Proposición 2.13 *Sea A un algoritmo para un modelo del Problema de la Secretaria Matroidal. Si A es α-probabilidad-competitivo, entonces es α-ordinal-competitivo.*

Demostración. Sea w función de pesos asignada adversarialmente y $\tilde{w} : E \to \mathbb{R}_+$ consistente con w. Se tiene que:

$$\begin{aligned}
\mathbb{E}[\tilde{w}(\text{ALG})] &= \sum_{x \in E} \tilde{w}(x) \cdot \mathbb{P}[x \in \text{ALG}] \\
&\geq \sum_{\bar{x} \in \text{OPT}} \tilde{w}(\bar{x}) \cdot \mathbb{P}[\bar{x} \in \text{ALG}] \\
&\geq \alpha \sum_{\bar{x} \in \text{OPT}} \tilde{w}(\bar{x}) \\
&= \alpha \cdot \tilde{w}(\text{OPT}),
\end{aligned}$$

con lo cual concluimos. □

2.1.3. Oráculos de Consejos

Como en el Problema de la Secretaria Simple, nos interesa estudiar cómo mejoran las competitividades de los modelos si le damos información ordinal adicional al algoritmo en ciertas etapas del proceso. El oráculo de consejo que se definió en el capítulo anterior no pareciera ser demasiado relevante en el contexto con restricciones matroidales, por ejemplo, un

elemento en el óptimo offline OPT no necesariamente satisface que es más pesado que todos los elementos vistos antes. Tal como hicimos en el capítulo anterior, pensemos que después de rechazar un elemento y antes de recibir el siguiente, el algoritmo puede hacer preguntas ordinales a los candidatos que aún no revelan sus pesos. Ahora, como los candidatos son elementos del conjunto de referencia de una matroide, podría ser relevante hacer distintas preguntas a cada candidato, a diferencia del caso simple en el que los elementos eran indistinguibles antes de conocer sus pesos. También podría resultar útil hacer más de una pregunta a cada candidato pendiente, con el fin de obtener con mayor precisión la ubicación de los pesos de los elementos pendientes relativa a la ubicación de los pesos ya conocidos. Consideraremos dos oráculos de consejos diferentes para el Problema de la Secretaria Matroidal, dando lugar a dos clases de modelos diferentes.

Primero, consideremos que en cada ronda de consejos el algoritmo puede realizar una sola pregunta ordinal a cada elemento cuyo peso aún no ha sido revelado. Consideremos la siguiente pregunta a hacer a un elemento $x \in E$ que no se ha visto, cuando el conjunto de elementos vistos es $S \subseteq E$: "¿Está x en la base de peso máximo de la matroide restringida al conjunto $S + x$, i.e., $x \in \mathrm{OPT}(S + x)$?" Resulta que esta es una pregunta ordinal válida, como se concluye a partir de la siguiente proposición:

Proposición 2.14 *Sea $M = (E, \mathcal{I})$ matroide, $x \in E$ con $\{x\} \in \mathcal{I}$, $w : E \to W$ función de pesos y $S \subseteq E - x$. Escribamos $\mathrm{OPT}(S) = \{z^1, ..., z^m\}$, donde $w(z^1) > w(z^2) > ... > w(z^m)$. Si $x \notin \mathrm{span}(S)$, entonces $x \in \mathrm{OPT}(S + x)$. Si $x \in \mathrm{span}(S)$ e*

$$i := \text{mín}\left(\{j \in [m] : x \in \mathrm{span}(\{z^1, ..., z^j\})\}\right),$$

entonces $x \in \mathrm{OPT}(S + x)$ si y solo si $w(x) > w(z^i)$.

La demostración de la proposición anterior es directa de la Proposición 2.10. Definimos el *consejo básico* para el Problema de la Secretaria Matroidal de la siguiente manera:

Definición 2.15 (Consejo Básico) *Sea $M = (E, \mathcal{I})$ matroide. Para un orden de llegada $\pi : [n] \to E$, una función de pesos $w : E \to W$, e $i \in [n]$, el consejo básico en el paso i, denotado BAdv_i, se define como el conjunto:*

$$\mathrm{BAdv}_i := \{x \in E \setminus E_i : x \in \mathrm{OPT}(E_i + x)\},$$

esto es, el conjunto de elementos que llegan después del paso i y que mejoran la solución óptima actual hasta el paso i.

Segundo, consideremos que en cada ronda de consejos el algoritmo puede realizar $O(\log(\rho))$ preguntas ordinales a cada elemento pendiente, donde $\rho = \mathrm{rank}(M)$. Con esta cantidad de preguntas, podemos determinar para cada elemento pendiente $x \in E$ entre qué dos pesos de elementos del óptimo actual se encuentra (solo se necesita una cantidad logarítmica de preguntas si se usa búsqueda binaria). Definimos el *consejo completo* para el Problema de la Secretaria Matroidal de la siguiente manera:

Definición 2.16 (Consejo Completo) *Sea $M = (E, \mathcal{I})$ matroide. Para un orden de llegada $\pi : [n] \to E$, una función de pesos $w : E \to W$ e $i \in [n]$, sea $\mathrm{OPT}(E_i) = \{z^1, ..., z^m\}$ el*

óptimo actual hasta el paso i, con $w(z^1) > w(z^2) > ... > w(z^m)$. El consejo completo en el paso i, denotado CAdv_i*, se define como la tupla* $\text{CAdv}_i := (A_0, A_1, ..., A_m)$*, donde:*

$$A_0 := \{x \in E \setminus E_i : w(x) > w(z^1)\},$$
$$A_j := \{x \in E \setminus E_i : w(z^{j+1}) < w(x) < w(z^j)\} \text{ para cada } j \in [m-1],$$
$$A_m := \{x \in E \setminus E_i : w(x) < w(z^m)\},$$

es decir, el consejo completo entrega la partición de $E \setminus E_i$ en $m+1$ conjuntos ordenados según el intervalo en el que se encuentran los pesos de sus elementos. Llamamos cajones a los conjuntos A_i.

En los modelos con consejo básico (respectivamente con consejo completo), el jugador tendrá acceso a un oráculo de consejo que se puede consultar después de procesar un elemento (agregarlo a la solución o rechazarlo) y antes de recibir el siguiente. Si el jugador consulta el oráculo después de procesar el i-ésimo elemento, recibe BAdv_i (resp. CAdv_i), tras lo cual el proceso de llegada de elementos continúa normalmente. Nuevamente, como en el capítulo 1, dividimos los modelos según la cantidad permitida de consultas al oráculo.

Para cada modelo de orden $\mathcal{M}$, escribimos $M\mathcal{M}$ para referirnos a la versión matroidal del problema. Escribimos $M\mathcal{M}$ para referirnos al modelo sin Consejo (o $M\mathcal{M}_0$), $M\mathcal{M}_{1B}$ (resp. $M\mathcal{M}_{1C}$) para referirnos al modelo con Consejo Básico Único (resp. Consejo Completo Único), y $M\mathcal{M}_{\infty B}$ (resp. $M\mathcal{M}_{\infty C}$) para referirnos al modelo con Consejos Básicos Ilimitados (resp. Consejos Completos Ilimitados).

2.1.4. Jerarquía de los Modelos

Al igual que en el Problema de la Secretaria Simple, más rondas de consejos mejoran o mantienen igual la competitividad. Más precisamente, para el modelo de orden $\mathcal{M}$, $\alpha_{\mathcal{M}} \leq \alpha_{\mathcal{M}_{1B}} \leq \alpha_{\mathcal{M}_{\infty B}}$ y $\alpha_{\mathcal{M}} \leq \alpha_{\mathcal{M}_{1C}} \leq \alpha_{\mathcal{M}_{\infty C}}$. Además, el consejo básico es más débil que el consejo completo, por lo cual $\alpha_{\mathcal{M}_{kB}} \leq \alpha_{\mathcal{M}_{kC}}$ para $k \in \{0, 1, \infty\}$. Para una cantidad fija de rondas de consejo (con un mismo tipo de oráculo de consejo), la misma jerarquía del capítulo 1 entre modelos de orden de llegada se mantiene.

Por último, notemos que el Problema de la Secretaria Simple es un caso particular del Problema de la Secretaria Matroidal. Definimos la matroide uniforme de la siguiente manera:

Definición 2.17 (Matroide k-Uniforme) *Sea E conjunto finito ($|E| = n$), $k \in [n]$ e $\mathcal{I} = \{I \subseteq E : |I| \leq k\}$. $U_n^k = (E, \mathcal{I})$ es la matroide k-uniforme de n elementos.*

Es claro que lo anterior define una matroide. Notamos que el Problema de la Secretaria Simple corresponde al Problema de la Secretaria Matroidal restringido a matroides 1-uniformes. Entonces, en los modelos sin consejos, la competitividad del problema matroidal es menor o igual a la competitividad del problema simple.

Para los modelos con consejo, notamos que el consejo básico y el consejo completo entregan la misma información que el consejo del Problema de la Secretaria Simple: los elementos que mejoran la solución óptima actual en el consejo básico corresponden a los elementos más

pesados que todos los anteriores cuando la matroide es U_n^1, y el consejo completo particiona el conjunto de elementos no vistos en el conjunto de los elementos menos pesados que el más pesado visto más el conjunto de los elementos más pesados que el más pesado visto. Como los elementos en la matroide uniforme son indistinguibles (ignorando sus pesos), los conjuntos proporcionados por los consejos del problema matroidal no entregan más información que la cantidad de elementos más pesados que el elemento más pesado ya visto. Concluimos así que, para un orden y una cantidad de rondas de consejos fijas, las competitividades de los problemas matroidales son menores o iguales a la competitividad del problema simple.

2.2. Resultados Conocidos

En esta sección mencionamos brevemente algunos resultados relevantes que se han obtenido para el Problema de la Secretaria Matroidal en investigaciones previas.

Babaioff et al. propusieron el Problema de la Secretaria Matroidal original (la versión cardinal, con orden de llegada aleatorio y sin consejos), y presentan un algoritmo $\Omega(1/\log(\rho))$-competitivo, donde ρ es el rango de la matroide con la cual se trabaja [5]. En el mismo trabajo, conjeturan que debe existir un algoritmo con competitividad constante para el problema, lo cual aún no se confirma hasta el momento de la escritura de esta tesis. En los años siguientes de la presentación del problema, éste ha sido revisado profundamente, encontrándose mejores garantías para el caso general, para subclases de matroides y para variaciones del problema.

En el problema general, luego se propuso un algoritmo $\Omega(1/\sqrt{\log(\rho)})$-competitivo para el problema [7] y después se propusieron algoritmos $\Omega(1/\log\log(\rho)))$-competitivos (primero en [21], después en [15]). Estos algoritmos no se pueden aplicar en el contexto ordinal, pues requieren operar aritméticamente con los pesos observados. Soto et al. propusieron algoritmos $\Omega(1/\log\log(\rho)))$-ordinal-competitivo y $\Omega(1/\log(\rho))$-probabilidad-competitivo para el problema ordinal [25].

Sí se han encontrado algoritmos con competitividades constantes para el problema ordinal restringido a clases específicas de matroides, por ejemplo [25]:

- Algoritmo $1/e$-probabilidad-competitivo para matroides transversales.
- Algoritmo $1/4$-probabilidad-competitivo para matroides gráficas.
- Algoritmo $1/(3\sqrt{3})$-probabilidad-competitivo para matroides laminares (revisar definición por ejemplo en [25]).

En general, estos algoritmos usan información propia de la clase sobre la cual trabajan para lograr tales garantías, por lo cual no nos entregan muchas pistas de cómo resolver el problema general.

También se han alcanzado competitividades constantes en versiones modificadas del problema general. Jaillet et al. proponen un algoritmo $1/4$-competitivo para el Problema de la Secretaria Matroidal en el modelo de Orden Libre [16]. En este modelo, el orden de llegada

de los elementos es elegido por el algoritmo, quien de antemano conoce la matroide sobre la cual se está trabajando. Ese algoritmo resulta ser muy relevante en el contexto de esta tesis, ya que el Algoritmo 11 con el que trabajaremos resulta ser una adaptación de éste al modelo con consejo básico (y sin orden libre). Por otro lado, Soto propone un algoritmo $(1 - 1/e)/16$-competitivo para el Problema de la Secretaria Matroidal en el Modelo de Asignación Aleatoria y Orden Adversarial [24], modelo en el que el adversario elige el orden total $(W, \leq)$, pero la asignación $w : E \to W$ se realiza de manera uniformemente aleatoria (entre todas las biyecciones entre ambos conjuntos).

2.3. Resultados Nuevos

2.3.1. Problema con Consejo Básico Único

Empezamos estudiando la versión del Problema de la Secretaria Matroidal en la que el algoritmo puede realizar una única consulta al oráculo de consejo básico. Recordemos que esto significa que, después de consultar el oráculo de consejos tras rechazar un elemento, el algoritmo es informado del conjunto de elementos no vistos que permiten mejorar la solución de peso máximo del problema restringido a los elementos ya vistos. Específicamente, el algoritmo que proponemos es válido en el modelo con Muestra Aleatoria y Orden Adversarial Online con Único Consejo Básico (SBOnAO_{1B}) y en los modelos más fáciles que éste. Como se mencionó en la sección anterior, el algoritmo es una adaptación del algoritmo propuesto en [16] para el Problema de la Secretaria Matroidal en el modelo de Orden Libre.

Algoritmo 11: Algoritmo de Selección Post-Muestra con Consejo Básico

Entrada: Matroide $M = (E, \mathcal{I})$.
Salida: Conjunto independiente ALG de M.
ALG $\leftarrow \emptyset$;
AUX $\leftarrow \emptyset$;
$s \leftarrow \text{Bin}(n, \frac{1}{2})$;
Fase de muestreo: observar una muestra uniforme $S \subseteq E$ de tamaño s;
$Z = \{z^1, ..., z^m\} \leftarrow \text{OPT}(S)$, donde $w(z^1) > w(z^2) > ... > w(z^m)$;
$G \leftarrow \text{BAdv}_s$;
Fase de preselección: sin observar pesos restantes:;
para $i \in [m]$ **hacer**
  $G_i \leftarrow G \cap \text{span}(\{z^1, ..., z^i\}) \setminus \text{span}(\{z^1, ..., z^{i-1}\})$;
  para $x \in G_i$ *en cualquier orden* **hacer**
    si AUX $+x \in \mathcal{I}$ **entonces** AUX $\leftarrow$ AUX $+x$;
  fin
fin
para $x \in G_{m+1} := E \setminus \text{span}(Z)$ *en cualquier orden* **hacer**
  si AUX $+x \in \mathcal{I}$ **entonces** AUX $\leftarrow$ AUX $+x$;
fin
Fase de selección:
para $x \in E \setminus S$ *en orden adversarial* **hacer**
  si $x \in$ AUX **entonces** ALG $\leftarrow$ ALG $+x$;
fin
devolver ALG.

En el algoritmo, introducimos una *fase de preselección*. Esta fase ocurre después de que el algoritmo consulta al oráculo de consejos, y en ella, el algoritmo selecciona los elementos que se agregarán a la solución ALG. Esta fase se realiza sin ver los pesos de los elementos de $E \setminus S$, por lo cual el algoritmo no está limitado por el orden de llegada $E \setminus S$. La razón por la cual el algoritmo puede realizar una buena selección en esta fase sin conocer los pesos de los elementos, es que el algoritmo no se encuentra limitado por el orden de llegada y el consejo recibido entrega suficiente información de los pesos. Por restricción del modelo, dijimos que el algoritmo solo puede seleccionar un elemento cuando se revela su peso, por lo cual en la fase de preselección agregamos los elementos a un conjunto auxiliar AUX y luego en la fase de selección (cuando sí se revelan los pesos) elegimos los elementos que quedaron en AUX (con lo cual, ALG = AUX).

En la fase de preselección después de tomar la muestra, el algoritmo particiona G en los G_i's y luego agrega los elementos del conjunto G a AUX usando el algoritmo glotón (Algoritmo 10) en un orden construido de la siguiente manera: primero llega el conjunto G_1 en cualquier orden, luego el conjunto G_2 en cualquier orden, y así sucesivamente hasta G_{m+1}. Si llamamos σ a este orden de $E \setminus S$, la salida del Algoritmo 11 corresponde a $\text{GLOTON}(M', \sigma)$, donde M' es la matroide M restringida al conjunto de referencia G. La clave está en que no se consideran elementos fuera de G, lo cual no se puede hacer en el modelo sin consejo si no se conocen los pesos de los elementos pendientes. Se tiene el mismo resultado que en el modelo de Orden Libre:

Teorema 2.18 *El Algoritmo 11 es* 1/4*-probabilidad-competitivo para el Problema de la Secretaria Matroidal con Muestra Aleatoria, Orden Adversarial Online y Consejo Básico Único* (MSBOnAO_{1B}).

Para demostrar este teorema, se puede realizar una reducción al problema en el modelo de orden libre y usar el teorema en [16]. Por completitud, realizaremos la demostración desde cero sin hacer uso de la reducción.

Demostración. Para mostrar que el Algoritmo 11 es 1/4-probabilidad-competitivo, tomamos $\bar{x} \in \text{OPT}$ y mostraremos que $\mathbb{P}[\bar{x} \in \text{ALG}] \geq 1/4$. Escribimos $E = \{x^1, ..., x^n\}$, donde $w(x^i) < w(x^j)$ si $i < j$. Para $j \in [n]$, definimos $E^j = \{x^1, ..., x^j\}$. Definimos las dos siguientes variables aleatorias:

$$j_1 := \text{mín}\left(\{j \in [n] : \bar{x} \in \text{span}((E^j \cap S) - \bar{x})\}\right),$$
$$j_2 := \text{mín}\left(\{j \in [n] : \bar{x} \in \text{span}((E^j \setminus S) - \bar{x})\}\right),$$

donde, si $\bar{x} \notin \text{span}(S - \bar{x})$ tomamos $j_1 = \infty$, y si $\bar{x} \notin \text{span}((E \setminus S) - \bar{x})$ tomamos $j_2 = \infty$.

Podemos visualizar estas variables aleatorias de la siguiente manera: Imaginemos que colocamos todos los elementos de E en una columna desde arriba hacia abajo en orden decreciente de pesos (diremos que x^i está *a profundidad* i). Imaginemos luego que construimos dos nuevas columnas, una a la izquierda y otra a la derecha de la columna original. En la columna de la izquierda copiamos los elementos distintos a $\bar{x}$ que entran en la muestra S, mientras que en la columna de la derecha copiamos los elementos distintos a $\bar{x}$ que quedan fuera de la muestra S. Los elementos copiados los dejamos a la misma altura que en la columna original (ver Figura 2.1). Imaginemos ahora que ejecutamos el algoritmo glotón (Algoritmo 10) sobre la columna izquierda desde arriba hacia abajo (en orden decreciente de pesos). En cada profundidad h de la columna, nos preguntamos si $\bar{x}$ se podría agregar a la solución actual del algoritmo glotón cuando ha tratado de agregar elementos hasta al elemento x^h (si $x^h \in S - \bar{x}$). Si en alguna profundidad la respuesta es "no", la primera profundidad en la que esto ocurre corresponde a j_1. Si no existe tal profundidad, es decir, $\bar{x}$ se puede agregar a la solución después de haber evaluado toda la columna izquierda, entonces $j_1 = \infty$. j_2 se entiende de manera análoga aplicando la misma idea a la columna de la derecha.

Afirmamos que cuando $j_1 \leq j_2$ y $\bar{x} \notin S$, $\bar{x}$ es seleccionado por el Algoritmo 11. Intuitivamente, notemos que solo $\bar{x}$ y los elementos en la columna derecha se consideran en la etapa post-muestra del algoritmo. Por definición de j_2, si cuando llegue $\bar{x}$ no hay elementos en AUX con profundidades $h \geq j_2$, entonces $\bar{x}$ se podrá agregar a AUX manteniendo factibilidad. La clave es que, como veremos, todos los elementos de G que lleguen antes que $\bar{x}$ estarán a profundidades $h < j_1$, por lo que cuando se analice $\bar{x}$ se podrá agregar a la solución.

Formalmente, notemos primero que si $\bar{x} \notin S$, entonces $\bar{x} \in G$. Sea I el conjunto de elementos añadidos antes del momento de considerar a $\bar{x}$, entonces $\bar{x}$ es añadido a la solución si y solo si $I + \bar{x} \in \mathcal{I}$. Nos ponemos en dos casos. Si $j_2 = \infty$, entonces $\bar{x} \notin \text{span}((E \setminus S) - \bar{x})$. Como $I \subseteq E \setminus S$, entonces $I + \bar{x} \in \mathcal{I}$.

Si $j_2 < \infty$, también $j_1 < \infty$. Entonces, existe $i \in [m]$ tal que $\bar{x} \in G_i$. Más aún, i es el menor índice tal que $\bar{x} \in \text{span}(\{z^1, ..., z^i\})$, es decir, $z^i = x^{j_1}$. El conjunto I solo contiene

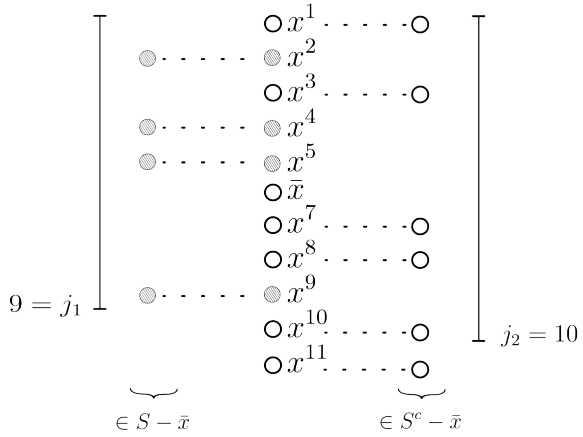

Figura 2.1: En la columna central se colocan todos los elementos de E de arriba a abajo en orden decreciente de pesos. Los elementos distintos de $\bar{x}$ que queden en la muestra (los achurados) se copian en la columna de la izquierda, mientras que los elementos distintos de $\bar{x}$ que queden fuera de la muestra se copian a la columna de la derecha. Las columnas de la izquierda y de la derecha distribuyen igual.

elementos con pesos estrictamente mayores a $w(x^{j_1})$, porque $I \subseteq \bigcup_{k=1}^{i} G_k$ y cada $x \in G_k$ satisface $w(x) > w(z^k)$ por la definición del consejo G y la Proposición 2.14. Entonces, como $I \subseteq E \setminus S$, se tiene que $I \subseteq E^{j_1-1} \setminus S \subseteq E^{j_2-1} \setminus S$. Por la definición de j_2, obtenemos que $I + \bar{x} \in \mathcal{I}$.

Habiendo demostrado la afirmación, basta ver que el evento "$j_1 \leq j_2$ y $\bar{x} \notin S$" ocurre con probabilidad al menos $1/4$. En efecto, para cada $T \subseteq E$, se tiene que $j_1 \leq j_2$ para $S = T$ o para $S = E \setminus T$. Como cada elemento entra en la muestra S con probabilidad $1/2$, los eventos $S = T$ y $S = E \setminus T$ son equiprobables, luego $\mathbb{P}[j_1 \leq j_2] \geq 1/2$. En la Figura 2.1, lo anterior nos dice que las columnas izquierda y derecha distribuyen igual. Además, que $\bar{x}$ esté o no en la muestra es independiente del evento $j_1 \leq j_2$. Se tiene entonces que:

$$\mathbb{P}[\bar{x} \in \text{ALG}] \geq \mathbb{P}[j_1 \leq j_2, \bar{x} \notin S] = \mathbb{P}[j_1 \leq j_2] \cdot \mathbb{P}[\bar{x} \notin S] \geq 1/4.$$

□

Pareciera que el consejo BAdv_s entrega mucha información al algoritmo y, de hecho, hasta el momento no sabemos si se puede realizar una reducción del problema sin consejo al problema con consejo, de tal manera de resolver el problema original con competitividad constante. Sin embargo, pensando el oráculo de consejos como una ronda de consejos en la que el jugador puede realizar preguntas ordinales a los elementos pendientes, el teorema anterior nos dice lo siguiente: si podemos en un solo instante realizar una sola pregunta ordinal a cada elemento pendiente, entonces podemos resolver el problema con competitividad constante.

Cabe señalar que, hasta el momento, no sabemos si la cota para el algoritmo y para el problema es ajustada: existe la posibilidad de que este algoritmo tenga competitividad estrictamente mayor a $1/4$ o que otro algoritmo para el problema tenga competitividad estrictamente mayor a $1/4$. En particular, podría ser que exista un mejor algoritmo que, además de usar el consejo, aproveche la información precisa de los pesos de los elementos que

llegan después de la muestra (el Algoritmo 11 toma la decisión de qué conjunto seleccionar antes de ver los pesos precisos). Por jerarquía de los modelos, tenemos que:

$$\alpha_{\mathrm{MRO}_{1B}}, \alpha_{\mathrm{MSBOffAO}_{1B}}, \alpha_{\mathrm{MRO}_{\infty B}}, \alpha_{\mathrm{MSBOffAO}_{\infty B}}, \alpha_{\mathrm{MSBOnAO}_{\infty B}} \geq \alpha_{\mathrm{MSBOnAO}_{1B}} \geq \frac{1}{4}.$$

2.3.2. Problema con Consejo Completo Único

Pasamos ahora al mismo problema, pero en el que cambiamos el oráculo de consejo básico por el oráculo de consejo completo. Recordemos que, con la información que entrega este oráculo al algoritmo, se puede saber entre qué 2 pesos de elementos del óptimo actual $\mathrm{OPT}(S)$ se encuentra cada elemento aún no procesado. El algoritmo que proponemos es válido en el modelo con Muestra Aleatoria y Orden Adversarial Online (SBOnAO_{1C}) y en los modelos más fáciles que éste.

Algoritmo 12: Algoritmo de Selección Post-Muestra con Consejo Completo

Entrada: Matroide $M = (E, \mathcal{I})$.
Salida: Conjunto independiente ALG of M.
Parámetros: $p \in [0, 1]$.
$\mathrm{ALG} \leftarrow \emptyset$;
$\mathrm{AUX} \leftarrow \emptyset$;
$s \leftarrow \mathrm{Bin}(n, p)$;
Fase de muestreo: observar una muestra uniforme $S \subseteq E$ se tamaño s;
$Z = \{z^1, ..., z^m\} \leftarrow \mathrm{OPT}(S)$, donde $w(z^1) > w(z^2) > ... > w(z^m)$;
$(A_0, A_1, ..., A_m) \leftarrow \mathrm{CADV}_s$;
Fase de preselección: sin observar pesos restantes:
para $x \in A_0$ *en orden aleatorio uniforme* **hacer**
  si $\mathrm{AUX} + x \in \mathcal{I}$ **entonces** $\mathrm{AUX} \leftarrow \mathrm{AUX} + x$;
fin
para $i \in [m]$ **hacer**
  si $\mathrm{AUX} + z^i \in \mathcal{I}$ **entonces** $\mathrm{AUX} \leftarrow \mathrm{AUX} + z^i$;
  para $x \in A_i$ *en orden aleatorio uniforme* **hacer**
    si $\mathrm{AUX} + x \in \mathcal{I}$ **entonces** $\mathrm{AUX} \leftarrow \mathrm{AUX} + x$;
  fin
fin
Fase de selección:
para $x \in E \setminus S$ *en orden adversarial* **hacer**
  si $x \in \mathrm{AUX}$ **entonces** $\mathrm{ALG} \leftarrow \mathrm{ALG} + x$;
fin
devolver ALG.

Tal como el Algoritmo 11, el Algoritmo 12 realiza toda la selección de elementos en la fase de preselección, donde tiene la información entregada por el oráculo de consejos después de la muestra, pero no los pesos precisos de cada elemento no procesado. A diferencia del algoritmo de la sección anterior, el Algoritmo 12 agrega a AUX también elementos que aparecían en la muestra S durante la fase de preselección, pero en la fase de selección se agregan a ALG los

elementos de AUX $\setminus S$.

En la fase de preselección después de tomar la muestra, el algoritmo agrega los elementos de $\text{OPT}(S) \cup E \setminus S$ al conjunto AUX usando el algoritmo glotón (Algoritmo 10) en un orden construido de la siguiente manera: primero llega A_0 en orden uniformemente aleatorio, luego llega z^1, luego A_1 en orden uniformemente aleatorio, luego z^2, y así sucesivamente hasta A_m. Si llamamos σ a este orden de $\text{OPT}(S) \cup E \setminus S$, se tiene que $\text{AUX} = \text{GLOTON}(M', \sigma)$, donde M' es la matroide M restringida al conjunto de referencia $\text{OPT}(S) \cup E \setminus S$. El algoritmo es bueno porque imita lo que hace el algoritmo que entrega la base de peso máximo en el problema de optimización offline, pero con incertidumbre de algunos pesos (no podemos ordenar en orden decreciente los pesos de los elementos de un conjunto A_i antes de recibirlos). Se tiene el siguiente resultado:

Teorema 2.19 *Tomando $p = 1/e$, el Algoritmo 12 es $1/e$-probabilidad-competitivo y esta competitividad es óptima para el Problema de la Secretaria Matroidal con Muestra Aleatoria, Orden Adversarial Online e Ilimitados Consejos Completos* ($\text{MSBOnAO}_{\infty C}$).

Notemos que el algoritmo consulta el oráculo de consejos una sola vez y que en el enunciado del teorema decimos que el algoritmo es óptimo en el caso con ilimitados consejos, por lo cual también es óptimo para el caso con único consejo.

Demostración. Para mostrar que el Algoritmo 12 es $1/e$-probabilidad-competitivo, tomamos $\bar{x} \in \text{OPT}$ y mostraremos que $\mathbb{P}[\bar{x} \in \text{ALG}] \geq 1/e$. Sea j la variable aleatoria que satisface que $y \in A_j$ cuando $\bar{x} \notin S$. Cuando $\bar{x} \in S$, anotamos $A_j = \emptyset$. Para cada $y \in E$, definimos:

$$E_{\leq w(y)} := \{x \in E : w(x) \leq w(y)\},$$
$$E_{> w(y)} := \{x \in E : w(x) > w(y)\}.$$

Para cada $i \in [m]$, definimos $Z^i := \{z^1, ..., z^i\}$. Definimos $B := A_j \cap E_{\leq w(\bar{x})} \setminus \text{span}(Z^j)$, el conjunto de los elementos en el mismo cajón que $\bar{x}$ (A_j) con peso menor o igual a $w(\bar{x})$ y que no son generados por elementos de Z más pesados que $\bar{x}$. Intuitivamente, el conjunto de elementos que pueden "bloquear" a $\bar{x}$ de ser elegido está contenido en B. Más precisamente, podemos verificar que si $\bar{x}$ es el primer elemento de B en aparecer en el orden aleatorio de A_j, entonces el algoritmo selecciona a $\bar{x}$. En efecto, si $\bar{x}$ es el primer elemento de B en aparecer en el orden aleatorio de A_j, entonces todos los elementos añadidos por el algoritmo a AUX antes que $\bar{x}$ tienen peso mayor a $w(\bar{x})$, ya que los elementos de conjuntos A_i con $i < j$ tienen peso mayor (por definición de los cajones) y los elementos de $\text{span}(Z^j) \cap A_j$ no pueden ser añadidos, porque cada elemento del conjunto Z^j fue previamente procesado por el algoritmo (por la Proposición 2.9, si un elemento u llega y W es el conjunto de elementos procesados antes que u, u puede ser añadido a la solución si y solo si $u \notin \text{span}(W)$). Como $\bar{x} \in \text{OPT}$, por la Proposición 2.10, $\bar{x} \notin \text{span}(E_{> w(\bar{x})})$, por lo cual $\bar{x}$ es añadido a AUX y luego a ALG.

Como el orden de los elementos de B se toma uniformemente al azar, se tiene que, para $t \in \{0, 1, ..., n\}$:

$$\mathbb{P}[\bar{x} \in \text{ALG} \,||B| = t] \geq \begin{cases} \frac{1}{t}, & \text{si } t > 0 \\ 0, & \text{si } t = 0 \end{cases} .$$

El último paso de la demostración consiste en determinar la distribución de $|B|$. Afirmamos lo siguiente: Condicionado al valor de $S \cap E_{<w(\bar{x})}$,

$$\mathbb{P}[|B| = t] = \begin{cases} p(1-p)^t, \text{ si } t \in \{0, ..., h-1\} \\ (1-p)^h, \text{ si } t = h, \end{cases}$$

donde $h := |E_{\leq w(y)} \setminus \text{span}(Z^j)|$. Procedemos a demostrar esta afirmación

Recordemos que la muestra S contiene cada elemento de E con probabilidad p de manera independiente (Lema 1.3). El caso $t = 0$ ocurre cuando $\bar{x} \in S$, lo cual ocurre con probabilidad $(1-p)$. Supongamos ahora que $t > 0$, i.e., $\bar{x} \notin S$.

Sea $E = \{x^1, ..., x^n\}$ en orden decreciente de pesos. El proceso de muestreo puede ser entendido de la siguiente manera: Lanzamos n monedas enumeradas independientes, cada una con probabilidad p de salir "cara" y probabilidad $(1-p)$ de salir "sello". Sea C^i el resultado de la moneda i-ésima, asociada al elemento x^i. Las monedas se lanzan en orden, y para cada $i \in [n]$, $C^i =$ cara cuando $x^i \in S$ y $C^i =$ sello en caso contrario.

Sea $k \in [n]$ tal que $x^k = \bar{x}$. Luego, el conjunto $E_{\leq w(\bar{x})} \setminus \text{span}(Z^j)$ (y el valor de h) está completamente determinado por los resultados de las primeras $k-1$ monedas. Condicionemos en los valores de las primeras k monedas, con $C^k =$ sello (porque $\bar{x} = x^k \notin S$). Recordemos que $\bar{x} \in A_j = \{x \in E \setminus S : w(z^{j+1}) < w(x) < w(z^j)\}$ (si $j = 0$, podemos inventar un elemento z^0 más pesado que todos los elementos de E; y si $j = m$, podemos inventar un elemento z^{m+1} menos pesado que todos los elementos de E). Notemos que z^{j+1}, el elemento más pesado de $\text{OPT}(S)$ con peso menor a $w(\bar{x})$, satisface $z^{j+1} = \text{argmax}(\{w(x) : x \in S \setminus \text{span}(Z^j)\})$, i.e., z^{j+1} es el elemento más pesado de $E_{\leq w(\bar{x})} \setminus \text{span}(Z^j)$ que además está en S. Entonces, la moneda de z^{j+1} es la primera moneda de $E_{\leq w(\bar{x})} \setminus \text{span}(Z^j)$ con resultado cara. Por lo tanto:

$$\mathbb{P}[|B| = t | \bar{x} \notin S] = \begin{cases} p(1-p)^{t-1}, \text{ si } t \in \{0, ..., h-1\} \\ (1-p)^{h-1}, \text{ si } t = h \end{cases},$$

donde $(1-p)^{t-1}$ corresponde a $(t-1)$ sellos consecutivos en $E_{<w(\bar{x})} \setminus \text{span}(Z^j)$, y el factor p corresponde a una cara después. Esto prueba la afirmación.

Combinando las expresiones, obtenemos:

$$\begin{aligned} \mathbb{P}[\bar{x} \in \text{ALG}] &= \sum_{c^1,...,c^{k-1}} \mathbb{P}[\bar{x} \in \text{ALG} \,|C^1 = c^1, ..., C^{k-1} = c^{k-1}] \cdot \mathbb{P}[C^1 = c^1, ..., C^{k-1} = c^{k-1}] \\ &\geq \sum_{c^1,...,c^{k-1}} \left(\sum_{t=1}^{h-1} \frac{p(1-p)^t}{t} + \frac{(1-p)^h}{h} \right) \cdot \mathbb{P}[C^1 = c^1, ..., C^{k-1} = c^{k-1}] \\ &\geq \sum_{c^1,...,c^{k-1}} \left(\sum_{t=1}^{\infty} \frac{p(1-p)^t}{t} \right) \cdot \mathbb{P}[C^1 = c^1, ..., C^{k-1} = c^{k-1}] \\ &= \sum_{t=1}^{\infty} \frac{p(1-p)^t}{t} = -p \ln(p), \end{aligned}$$

donde en las sumas los c^i toman los valores "cara" y "sello". El condicionamiento en los valores de $C^1, ..., C^{k-1}$ es necesario, porque el valor de h en la segunda línea depende de tales valores. Tomando $p = 1/e$, obtenemos el algoritmo $1/e$-probabilidad-competitivo.

El algoritmo es óptimo para el Problema de la Secretaria Matroidal con Muestra Aleatoria, Orden Adversarial Online y Consejo Completo Único, porque el problema tiene como caso particular el Problema de la Secretaria Simple con Muestra Aleatoria, Orden Adversarial Online y Consejo Único (cuya competitividad es $1/e$), tal como se señala en la subsección 2.1.4. □

Cabe señalar que la técnica de demostración usada acá (específicamente la parte del lanzamiento de monedas) está inspirada por las ideas en [23]. Podemos notar las similitudes de esta demostración con la del Teorema 1.12. Esto es natural, porque resulta que el Algoritmo 4 es el caso particular del Algoritmo 12 cuando la matroide es U_n^1: el consejo completo se convierte en el consejo del problema simple, el conjunto B de elementos que "bloquean" se convierte en el conjunto de elementos más pesados que el más pesado visto en la muestra ($j = 0$ y $E_{\leq w(\bar{x})} = E$, luego $B = A_0$), y el algoritmo elige un elemento de A_0 uniformemente al azar. Por jerarquía de los modelos, tenemos que:

$$\alpha_{\mathrm{MRO}_{1C}}, \alpha_{\mathrm{MSBOffAO}_{1C}}, \alpha_{\mathrm{MRO}_{\infty C}}, \alpha_{\mathrm{MSBOffAO}_{\infty C}} \geq \alpha_{\mathrm{MSBOnAO}_{1C}} = \alpha_{\mathrm{MSBOnAO}_{\infty C}} = \frac{1}{e}.$$

2.3.3. Problema con Orden Aleatorio y Consejos Completos Ilimitados

Por último, nos enfocamos en el problema con consejos completos ilimitados, en el contexto de un orden de llegada uniformemente aleatorio. Así como en la subsección anterior encontramos la generalización a matroides del algoritmo óptimo para $\mathrm{SBOnAO}_{\infty C}$, en esta subsección generalizaremos el Algoritmo 9 para RO_∞ al problema matroidal. Proponemos el siguiente algoritmo:

Algoritmo 13: Algoritmo Primero después del Segundo (Matroidal)

Entrada: Matroide $M = (E, \mathcal{I})$.
Salida: Conjunto independiente ALG de M.
ALG $\leftarrow \emptyset$;
si $x_1 \notin \mathrm{span}(E \setminus \{x_1\})$ **entonces** ALG $\leftarrow$ ALG $+x_1$;
$(A_0, A_1) \leftarrow \mathrm{CADV}_1$;
para $i \in [2..n]$, *recibir* x_i *y* **hacer**
  $j \leftarrow$ índice tal que $x_i \in A_j$;
  si $x_i \notin \mathrm{span}\left((\bigcup_{k=0}^{j} A_k \cup \{x \in E_{i-1} : w(x) > w(x_i)\}) - x_i\right)$ **entonces**
    ALG $\leftarrow$ ALG $+x_i$;
  $(A_0, ..., A_m) \leftarrow \mathrm{CADV}_i$, con $m = |\,\mathrm{OPT}(E_i)|$;
fin
devolver ALG.

Por la Proposición 2.10, sabemos que los elementos $\bar{x} \in \mathrm{OPT}$ no están generados por el conjunto de elementos más pesados que $\bar{x}$. Cuando llega un elemento x, no conocemos el conjunto de elementos más pesados que él de manera precisa, pero sí sabemos qué elementos vistos previamente son más pesados y conocemos la información entregada por el oráculo de consejos completos en el paso anterior. La idea detrás del Algoritmo 13 es verificar en

cada paso si el elemento que llega satisface una condición más restrictiva que la condición de optimalidad de la Proposición 2.10. Se tiene el siguiente resultado:

Teorema 2.20 *El Algoritmo 13 es* $1/2$*-probabilidad-competitivo y esta competitividad es óptima para el Problema de la Secretaria Matroidal con Orden Aleatorio y Consejos Completos Ilimitados* ($\text{MRO}_{\infty C}$).

Demostración. Primero mostramos que ALG $\subseteq$ OPT. En efecto, si $x \in$ ALG llegó en el paso t, sea $(A_0, ..., A_m) = \text{CADV}_{t-1}$ la tupla dada por el oráculo después de haberse rechazado el elemento previo, y sea $j \in \{0, ..., m\}$ tal que $x \in A_j$. Como $x \in$ ALG, se tiene que:

$$x \notin \text{span}\left((\bigcup_{k=0}^{j} A_k \cup \{z \in E_{t-1} : w(z) > w(x)\}) - x \right) \supseteq \text{span}\left(\{z \in E : w(z) > w(x)\}\right),$$

donde la inclusión se debe a que el argumento del segundo span$(\cdot)$ está contenido en el argumento del primer span$(\cdot)$ y span$(\cdot)$ es una función monótona. Se tiene entonces que $x \notin \text{span}\left(\{z \in E : w(z) > w(x)\}\right)$, luego, por la Proposición 2.10, $x \in$ OPT.

Ahora, sea $\bar{x} \in$ OPT. Queremos mostrar que $\mathbb{P}[\bar{x} \in \text{ALG}] \geq 1/2$. $E = \{x^1, ..., x^n\}$, donde los elementos se listan en orden decreciente de pesos. Si $\bar{x} \notin \text{span}(E - \bar{x})$, entonces siempre será añadido por el algoritmo. Si $\bar{x} \in \text{span}(E - \bar{x})$, sea $i \in [n]$ el mínimo índice tal que $\bar{x} \in \text{span}(\{x^1, ..., x^i\} - \bar{x})$. Como $\bar{x} \in$ OPT, por la Proposición 2.10 se tiene que $w(\bar{x}) > w(x^i)$.

Afirmamos que si x^i aparece antes que $\bar{x}$ en el orden aleatorio, entonces $\bar{x}$ es seleccionado por el algoritmo. En efecto, supongamos que x^i llega en el paso s y $\bar{x}$ llega en el paso t, con $s < t$. Entonces, $x^i \in \text{OPT}(E_{t-1})$, si no, $\text{span}(\{x^1, ..., x^i\} - \bar{x}) = \text{span}(\{x^1, ..., x^{i-1}\} - \bar{x})$, lo cual contradice la minimalidad de i. Esto implica que ningún conjunto en CADV_{t-1} tiene elementos u, v tales que $w(u) > w(x^i)$ y $w(v) < w(x^i)$. Si $\text{CADV}_{t-1} = (A_0, ..., A_m)$, sea j el índice tal que $\bar{x} \in A_j$. Como $x^i \in \text{OPT}(E_{t-1})$ y $w(\bar{x}) > w(x^i)$, los pesos de los elementos en los conjuntos $A_0, ..., A_j$ son todos mayores que $w(x^i)$, luego:

$$\left((\bigcup_{k=0}^{j} A_k \cup \{z \in E_{t-1} : w(z) > w(\bar{x})\}) - \bar{x} \right) \subseteq \{x^1, ..., x^{i-1}\}$$

$$\Rightarrow \text{span}\left((\bigcup_{k=0}^{j} A_k \cup \{z \in E_{t-1} : w(z) > w(\bar{x})\}) - \bar{x} \right) \subseteq \text{span}(\{x^1, ..., x^{i-1}\}) \not\ni \bar{x},$$

con lo cual $\bar{x}$ es seleccionado. Concluimos que la competitividad del algoritmo es $1/2$ tras notar que x^i aparece antes que $\bar{x}$ con probabilidad $1/2$ en el orden aleatorio uniforme.

El algoritmo es óptimo para el Problema de la Secretaria Matroidal con Orden Aleatorio y Consejos Completos Ilimitados, porque el problema tiene como caso particular el Problema de la Secretaria Simple con Orden Aleatorio y Consejos Completos Ilimitados (cuya competitividad es $1/2$), tal como se señala en la subsección 2.1.4. □

Nuevamente, como lo hicimos en la subsección 1.3.7, notamos que incluso en el contexto matroidal, el problema más fácil con consejos ordinales es igual de difícil que el problema más difícil con información distribucional, el Problema de Desigualdad del Profeta Matroidal con competitividad $1/2$ [19].

Conclusión

En esta tesis se estudiaron diversas variaciones del Problema de la Secretaria (Simple y Matroidal), con diferentes órdenes de llegada de los candidatos y distinta disponibilidad de consultas permitidas a oráculos de consejos ordinales.

En el caso simple, se hallaron algoritmos óptimos para todos los problemas excepto uno, el Problema de la Secretaria Simple con Muestra Aleatoria, Orden Adversarial Offline y Consejo Único (SBOffAO_1), en el cual solo obtuvimos que la competitividad se encuentra en el intervalo $[1/e, (20\sqrt{10}-29)/81]$, intervalo que se obtiene de la jerarquía entre los modelos. Resulta de interés encontrar un valor más ajustado para esa competitividad, en particular, sería interesante verificar si la competitividad es igual que en el caso de adversario online, es decir, si $\alpha_{\text{SBOffAO}_1} = \alpha_{\text{SBOnAO}_1} = 1/e$. A priori no tenemos claridad si esa conjetura es cierta, dado que las competitividades con adversario online y offline son iguales cuando no hay consejo, pero son distintas cuando hay ilimitados consejos. Dentro de los resultados, es notable el cómo la cantidad de consejos impacta en la garantía de los algoritmos en el caso con adversario offline y en el caso con orden aleatorio. Creemos que todos estos modelos pueden ser una buena adición al repertorio de variaciones del Problema de la Secretaria con información adicional que se estudian, y que podrían servir de base para estudiar otras variaciones más realistas (por ejemplo, variaciones en las que el oráculo entrega respuestas con cierto nivel de error, simulando respuestas de los candidatos menos certeras). Por otro lado, los análisis hechos son bien variados y tienen valor en el contexto del estudio de garantías de algoritmos para problemas de naturaleza online.

En el caso matroidal, se hallaron algoritmos con competitividades constantes cuando el jugador tiene acceso al oráculo de consejos (en los casos distintos a los que tienen orden adversarial sin muestra). Al momento de la escritura de esta tesis, aún no se sabe si existen algoritmos con competitividades constantes (independientes de los rangos de las matroides) en el problema sin consejos, por lo cual es relevante encontrar algoritmos con competitividades constantes en escenarios donde el jugador tiene más libertades y/o habilidades adicionales. Sería interesante revisar si existen reducciones que permitan pasar del problema matroidal sin información adicional a uno con información adicional agregando solo un factor constante a la competitividad, pero de momento no se han obtenido resultados en esta línea. Observamos que las competitividades constantes para los problemas con consejo se alcanzan incluso en el contexto con Muestra Aleatoria y Orden Adversarial Online (MSBOnAO_1), por lo que se podría pensar que el paso a Orden Aleatorio permita mantener una competitividad constante incluso tras sacar los consejos, así como nos dimos cuenta que el Problema Simple con Muestra Aleatoria, Orden Adversarial Online y Consejo Único (SBOnAO_1) es prácticamente el mismo problema que el Problema Simple con Orden Aleatorio (RO). Ahora bien, es importante notar

que el consejo básico matroidal entrega mucha más información que el consejo del problema simple, por lo cual no podemos estar seguros que se pueda realizar una reducción tan sencilla. En estos modelos, aún hay espacio para mejores algoritmos en los casos con consejos básicos y en los casos con orden aleatorio, en los cuales no se obtuvo ninguna cota superior nueva (que no se deduzca directamente de las cotas del problema simple).

Dentro de las futuras líneas de estudio posibles se encuentra el estudio del modelo $\mathrm{SBOffAO}_1$, en el cual solo tenemos cotas posiblemente no ajustadas que se heredan de otros modelos. También resulta relevante el estudio de casos con m accesos al oráculo de consejos, con más limitaciones a las preguntas que se pueden hacer a los candidatos, con oráculos que entregan respuestas con cierta incertidumbre, y sobre sistemas distintos a matroides. Dentro de éste último, podría ser interesante estudiar modelos con consejo para el Problema de la Secretaria en k-sistemas, sistemas de independencia en los que los tamaños de las bases (conjuntos independientes maximales) se diferencian en a lo más un factor k. Las matroides son 1-sistemas y algunas de las ideas que se usan para estudiarlas se aplican a k-sistemas, pero las proposiciones usadas en esta tesis no son válidas, por lo cual se requiere de más trabajo para llegar a nuevos resultados.

Bibliografía

[1] Susanne Albers y Leon Ladewig: *New results for the k-secretary problem.* Theoretical Computer Science, 863:102–119, 2021.

[2] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer y Pavel Kolev: *Secretary and Online Matching Problems with Machine Learned Advice.* En *Advances in Neural Information Processing Systems*, volumen 33, páginas 7933–7944. Curran Associates, Inc., 2020.

[3] Pablo D. Azar, Robert Kleinberg y S. Matthew Weinberg: *Prophet Inequalities with Limited Information.* En *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, páginas 1358–1377. SIAM, 2014.

[4] Moshe Babaioff, Nicole Immorlica, David Kempe y Robert Kleinberg: *Matroid Secretary Problems.* Journal of the ACM, 65(6):1–26, 2018.

[5] Moshe Babaioff, Nicole Immorlica y Robert Kleinberg: *Matroids, Secretary Problems, and Online Mechanisms.* En *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*, páginas 434–443. SIAM, 2007.

[6] Martin J. Beckmann: *Dynamic programming and the secretary problem.* Computers & Mathematics with Applications, 19(11):25–28, 1990.

[7] Sourav Chakraborty y Oded Lachish: *Improved Competitive Ratio for the Matroid Secretary Problem.* En *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*, páginas 1702–1712. SIAM, 2012.

[8] José Correa, Andrés Cristi, Laurent Feuilloley, Tim Oosterwijk y Alexandros Tsigonias-Dimitriadis: *The Secretary Problem with Independent Sampling.* En *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA'21)*, páginas 2047–2058. SIAM, 2021.

[9] José Correa, Andrés Cristi, Boris Epstein y José Soto: *The Two-Sided Game of Googol.* Journal of Machine Learning Research, 23(113):1–37, 2022.

[10] José Correa, Raimundo Saona y Bruno Ziliotto: *Prophet secretary through blind strategies.* Mathematical Programming, 190(1-2):483–521, 2021.

[11] Paul Dütting, Silvio Lattanzi, Renato Paes Leme y Sergei Vassilvitskii: *Secretaries with Advice.* En *Proceedings of the 22nd ACM Conference on Economics and Computation (EC'21)*, páginas 409–429. ACM, 2021.

[12] Eugene B. Dynkin: *The optimum choice of the instant for stopping a Markov process.* Soviet Mathematics, 4:627–629, 1963.

[13] Soheil Ehsani, MohammadTaghi Hajiaghayi, Thomas Kesselheim y Sahil Singla: *Prophet Secretary for Combinatorial Auctions and Matroids.* En *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, páginas 700–714. SIAM, 2018.

[14] Thomas Erlebach, Michael Hoffmann y The Algorithmics Column por Gerhard J. Woeginger: *Query-Competitive Algorithms for Computing with Uncertainty.* Bulletin of EATCS, 2, 2015.

[15] Moran Feldman, Ola Svensson y Rico Zenklusen: *A Simple O(log log(rank))-Competitive Algorithm for the Matroid Secretary Problem.* Mathematics of Operations Research, 43(2):638–650, 2018.

[16] Patrick Jaillet, José A. Soto y Rico Zenklusen: *Advances on Matroid Secretary Problems: Free Order Model and Laminar Case.* En *Integer Programming and Combinatorial Optimization 2013*, volumen 7801 de *Lecture Notes in Computer Science*, páginas 254–265. Springer, 2013.

[17] Haim Kaplan, David Naori y Danny Raz: *Competitive Analysis with a Sample and the Secretary Problem.* En *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, páginas 2082–2095. SIAM, 2020.

[18] Haim Kaplan, David Naori y Danny Raz: *Online Weighted Matching with a Sample.* En *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA'22)*, páginas 1247–1272. SIAM, 2022.

[19] Robert Kleinberg y SM Weinberg: *Matroid Prophet Inequalities.* En *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing (STOC'12)*, páginas 123–135. ACM, 2012.

[20] Ulrich Krengel y Louis Sucheston: *On semiamarts, amarts, and processes with finite value.* Probability on Banach spaces, 4:197–266, 1978.

[21] Oded Lachish: *O(log log rank) Competitive Ratio for the Matroid Secretary Problem.* En *2014 IEEE 55th Annual Symposium on Foundations of Computer Science (FOCS 2014)*, páginas 326–335. IEEE, 2014.

[22] Denis V. Lindley: *Dynamic Programming and Decision Theory.* Applied Statistics, 10:39–51, 1961.

[23] Tengyu Ma, Bo Tang y Yajun Wang: *The Simulated Greedy Algorithm for Several Submodular Matroid Secretary Problems.* Theory of Computing Systems, 58(4):681–706, 2016.

[24] José A. Soto: *Matroid secretary problem in the random-assignment model.* SIAM Journal on Computing, 42(1):178–211, 2013.

[25] José A. Soto, Abner Turkieltaub y Victor Verdugo: *Strong Algorithms for the Ordinal Matroid Secretary Problem.* Mathematics of Operations Research, 46:642–673, 2021.