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

RESULTADOS EN EL MODELO PROPHET SECRETARY Y MODELOS RELACIONADOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO

SANTIAGO VICENTE REBOLLEDO VIDAL

PROFESOR GUÍA:
JOSÉ SOTO SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
IVÁN RAPAPORT ZIMERMANN
VÍCTOR VERDUGO SILVA

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

SANTIAGO DE CHILE
2024

Resumen

En el modelo Prophet Secretary Matching con agentes, consideramos un hiper-grafo $G = (V, E)$, dónde las hiper-aristas son de orden a lo más k, con k un entero positivo. Un conjunto finito de *agentes* se presentan uno a uno en orden uniformemente aleatorio y un algoritmo asigna, ya sea una única arista de E, o bien, ninguna arista, a cada agente, en el momento en que este se presenta. El conjunto de aristas asignadas a agentes debe formar un matching en G. Al momento de presentarse, cada agente revela para cada arista de E un valor(real positivo) asociado a esta, provenientes de una distribución de probabilidad sobre todas las valuaciones posibles de aristas. Con esto, el algoritmo busca hacer asignaciones tal que la suma de valores que cada agente da a la arista que le fue asignada sea lo más grande posible. Una de las dificultades del problema recae en el hecho de que, al inicio del proceso, el algoritmo solo conoce la distribución de probabilidad de cada agente,y su valuación efectiva de las aristas solo es revelada en el momento en que este se presenta.

En esta memoria nos enfocamos en estudiar cotas superiores sobre la competitividad máxima que puede tener cualquier algoritmo bajo distintas condiciones sobre G. El primer caso que estudiamos es para G general y mostramos una cota superior $O(\frac{\log(k)}{k})$ para todo algoritmo. El segundo es cuándo G posee un solo elemento. En este caso presentamos un nuevo método para calcular cotas superiores, que llamamos "método continuo", y lo utilizamos para obtener 2 cotas superiores ya conocidas.

Por último presentamos un nuevo modelo que llamamos "modelo no-show" en el que cada agente tiene una probabilidad de no presentarse nunca al proceso, y establecemos cotas superiores en este modelo para $k = 1$ y $k = 2$.

A mis padres, Rolando y Fernanda.

Tabla de Contenido

Capítulo 1

Introducción

En esta memoria estudiamos algunos problemas de selección en linea en los que se estudia la relación entre un algoritmo que debe tomar decisiones en tiempo real con información incompleta y la solución óptima "offline" con información completa.

En los modelos **Prophet** y **Prophet Secretary** [1] se nos entrega una familia $(D_i)_{i\in[m]}$, dónde cada D_i es una distribución de probabilidad sobre $\mathbb{R}_+$. A partir de estas se genera una secuencia $(X_i)_{i\in[m]}$ de realizaciones ocultas, es decir, valores reales dónde $X_i \sim D_i$ para todo $i \in [m]$. Por último, se genera una permutación $\Pi : [m] \to [m]$, que representa el orden en que se revelarán los valores. Seleccionamos un único valor de manera "online", esto es, en la etapa k se nos revela el par (X_{π_k}, π_k) y debemos decidir entre seleccionar ese valor y obtener ganancia X_{π_k}, o pasar a la etapa $k+1$. El proceso termina una vez seleccionado un valor o bien una vez terminadas las m etapas. Nuestro objetivo es diseñar un mecanismo de selección, que llamaremos algoritmo, que reciba una instancia de distribuciones $(D_i)_i$ y maximice nuestra ganancia esperada. Dado un algoritmo, llamaremos ALG a su ganancia esperada, cuándo no haya ambigüedad respecto a qué algoritmo nos estamos refiriendo. Llamaremos OPT a la ganancia esperada por un "profeta" que posee información completa desde un comienzo y que por lo tanto siempre selecciona el valor máximo. Luego,

$$OPT = \mathbb{E}(\max_i X_i).$$

Podemos asociar el proceso anterior con la metáfora de un vendedor que desea vender un único objeto. Existen m agentes interesados en el objeto y se presentan uno a uno con una oferta por el objeto (el agente i ofrece X_i), frente a lo cuál el vendedor elige entre vender el objeto a ese agente o pasar al siguiente agente. Las distribuciones $(D_i)_{i\in[m]}$ representan cierto conocimiento previo que tiene el vendedor respecto al interés de cada agente por el objeto.

En el modelo **Prophet** el orden de llegada de los agentes es seleccionado de forma **adversarial**, es decir, la permutación Π es definida al comienzo por un "adversario" que busca minimizar la ganancia esperada de todo algoritmo. Este orden se establece a priori para todos los algoritmos, es decir, el adversario no sabe de antemano qué algoritmo específico será utilizado.

[1] Durante la memoria utilizaremos palabras en negrita cuándo estas sean definiciones o modelos.

Por otro lado, en el modelo **Prophet Secretary** el orden de llegada de los agentes es seleccionado de forma uniformemente aleatoria, es decir, se escoge una permutación Π equiprobablemente entre el conjunto de permutaciones posibles $\{\Pi \mid \Pi : [m] \to [m]\ \}$.

Dada una instancia, decimos que la **razón de competitividad** (o **razón**) de un algoritmo en esa instancia es $\frac{ALG}{OPT}$ (1 en el caso degenerado cuándo $OPT = 0$).

Dado $\alpha \in [0, 1]$, decimos que un algoritmo es α**-competitivo** si para toda instancia tiene una **razón** mayor o igual a α.

Dado $\beta \in [0, 1]$, decimos que un modelo es β**-difícil**, si mostramos que todo algoritmo en ese modelo es a lo más β**-competitivo**. Una forma de mostrar que un modelo es β**-difícil** (y será el método que utilizaremos a lo largo de este informe) es presentar una instancia en la cuál todo algoritmo óptimo tiene **razón** β.

Si en un modelo $M1$ existe un algoritmo α**-competitivo** y un modelo $M2$ es β**-difícil** con $\beta < \alpha$, decimos que $M1$ y $M2$ están **separados**.

Se conocen en el modelo **Prophet** algoritmos $\frac{1}{2}$**-competitivos** [7] y se sabe que el modelo es $\frac{1}{2}$**-difícil**, por lo tanto los mejores algoritmos conocidos son óptimos.

Por otra parte, en el modelo **Prophet-Secretary** el mejor algoritmo conocido hasta el momento es 0,669**-competitivo** [2], mientras que se ha logrado probar que es 0,7235**-difícil** [5] con lo que:

$$0{,}669 < \text{Mejor competitividad posible para } \textbf{Prophet Secretary} < 0{,}7235.$$

Por lo tanto, determinar el algoritmo con mejor competitividad para **Prophet-Secretary** es un problema abierto.

En **Prophet** y **Prophet Secretary** el objetivo es seleccionar solo 1 objeto (o valor) de una lista. En esta memoria estudiaremos estos problemas, pero también versiones más generales, dónde el objetivo consiste en elegir iterativamente un matching, en un grafo o hipergrafo, seleccionando aristas cuyos pesos se revelan en linea.

1.1. Resumen de lo presentado

En está memoria estudiamos algoritmos y ejemplos en los modelos **Prophet-Secretary** y modelos relacionados.

En la sección 2 nos enfocamos en el modelo **Prophet Secretary**. Comenzamos presentando algunos resultados clásicos para luego presentar una nueva cota superior que muestra que para el caso k general (con k representando el tamaño máximo de la aristas de un hiper grafo) el modelo es $O(\log(k)/k)$-**difícil**.

Luego nos enfocamos en el caso con 1 objeto e introducimos un nuevo método para calcular cotas ya conocidas, que llamaremos **método continuo**. Usando este, obtenemos las

cotas superiores ya conocidas $\sqrt{3}-1$, debida a Correa, Saona y Ziliotto [2], y la cota 0,7235, debida a Giambartolomei, Mallmann-Trenn y Saona [5], que se tienen para el problema.

Por último, en la sección 3 presentamos un nuevo modelo que llamaremos **No-Show**, casi idéntico al modelo **Prophet secretary**, pero con la diferencia de que en este los agentes pueden decidir no presentarse al proceso con cierta probabilidad. Vemos que este modelo es más difícil que **Prophet Secretary** y presentamos cotas superiores que superan las que se tienen en este último. En particular mostramos que para 1 objeto ambos modelos están **separados**.

Capítulo 2

Prophet Secretary Matching con agentes

2.1. Descripción del problema

Consideremos la siguiente situación:

Sean m, n y k enteros positivos no nulos y tal que $k < n$. Un vendedor posee un conjunto $[n]$ de objetos que desea vender en subconjuntos para obtener la máxima ganancia posible. Durante el día recibirá uno a uno un conjunto $[m]$ de clientes que le presentarán ofertas por distintas combinaciones de objetos, por ejemplo, un cliente podría presentarle una oferta por los subconjuntos $\{1, 2, 3\}$ y $\{2, 5, 6\}$, pero no estar interesado en llevarse ninguno de sus elementos por si solos. Ningún cliente estará interesado en una combinación de objetos de tamaño mayor a k. El vendedor conoce la valoración de cada cliente solo en el momento en que este se presenta, previo a ello solo tiene una "estimación" probabilística de como valora cada cliente cada combinación. Cada vez que un cliente se presente el vendedor debe venderle una única combinación de objetos (pudiendo esta ser vacía, es decir, no venderle nada) y obtendrá como ganancia la oferta que le había hecho el cliente por esa combinación. Una vez hecha la asignación, el cliente no regresará y la asignación no podrá deshacerse. El vendedor no le da valor a conservar ninguno de sus objetos y busca maximizar su ganancia total esperada después de haber hecho asignaciones a los $[m]$ clientes. Formalmente, modelamos este problema de la siguiente forma.

Consideremos un Hiper-Grafo $G = (V, E)$ con $|V| = n$, y aristas de tamaño a lo más k. Permitimos aristas paralelas, es decir, aristas diferentes que contienen los mismos vértices. Sea $A = [m]$ un conjunto de agentes. Cada agente $i \in [m]$ tiene asociada una función de valuación $f_i : E \to \mathbb{R}_+$ de las aristas de G, proveniente de una distribución de probabilidad D_i sobre todas las funciones de valuación posibles, es decir, sobre el conjunto $\{f | f : E \to R_+\}$. Las distribuciones son independientes entre si. Por último, sea $\pi(A)$ una permutación de A seleccionada ya sea de forma adversarial (para el problema **prophet**) o de forma aleatoria (para el problema **prophet secretary**). En el problema **prophet matching con agentes**, (respectivamente **prophet secretary matching con agentes**), los agentes se presentan

a un algoritmo en el orden π descrito anteriormente. El algoritmo debe asignar de manera online aristas del grafo a los agentes de modo que siempre las aristas asignadas sean un matching M. Más precisamente, el algoritmo comienza con un matching M vacío. En el paso k se presenta el agente π_k y le revela su función de valuación f_{π_k}, ante lo cual el algoritmo puede saltarse el agente (no asignarle ninguna arista), o bien asignarle una arista e que no haya sido asignada antes y tal que $M \cup e$ es un matching. La arista e se agrega entonces al matching M de aristas asignadas y el algoritmo recibe una ganancia igual a $f_{\pi_k}(e)$, que es el valor que el agente π_k le asigna a la arista e. El proceso termina luego de que se presenten todos los agentes.

Un caso particular de los problemas **prophet** y **prophet secretary** con agentes, es cuando los agentes son **single-minded**, es decir, donde cada agente i es "dueño" de una arista e_i en E de manera preestablecida, y las únicas funciones de valuación que pueden ser presentada por el agente i (i.e. el soporte de D_i) son funciones que le asignan peso no negativo a e_i y peso 0 a cualquier arista diferente de e_i.

En esta memoria nos restringimos a hipergrafos k-uniformes, es decir, donde cada arista consiste de exactamente k vértices.

El caso clásico del problema **prophet** (respectivamente **prophet secretary**) corresponde entonces al caso **single-minded**, donde además el hipergrafo tiene 1 solo vértice v y es 1-uniforme (es decir, hay m "aristas paralelas" cada una de ellas conteniendo al único objeto v).

Otro caso particular es cuándo las distribuciones $(D_i)_{i\in[m]}$ de todos los agentes son idénticas. A este modelo lo llamaremos **IID**. Es claro que este modelo es "más fácil" para el algoritmo que el modelo **prophet secretary** (es decir se puede obtener igual o mejor competitividad), pues es un caso particular de este. Para el caso **IID** con un objeto se conoce un algoritmo 0,7451**-competitivo** y se sabe que esta es la mejor competitividad que puede lograrse. [3]

2.2. Survey de resultados relacionados

2.2.1. Algoritmos

Problema del Profeta clásico

Se conoce desde 1977 por Krengel, Sucheston y Garling [6], quienes plantearon originalmente el problema, un algoritmo $\frac{1}{2}$**-competitivo** para el problema clásico del profeta y se sabe que esta es la mejor competitividad que puede conseguirse. En el problema clásico del profeta tenemos un sólo objeto y el orden de llegada de los agentes es adversarial.

La competitividad $\frac{1}{2}$ se consigue estableciendo al inicio del proceso el umbral de aceptación $V^* = \frac{1}{2} \cdot OPT$ [7]. Esto quiere decir que rechazaremos a cualquier agente que muestre un valor menor a V y aceptaremos al primer agente en mostrar un valor mayor a V^*. Recordar

que $OPT = \mathbb{E}(\text{máx}_i\, X_i)$ dónde X_i es el valor del agente i.

Para ver que el problema es $\frac{1}{2}$**-difícil**, consideremos la instancia con 2 agentes, en que, el primero en llegar es determinista y muestra valor 1 y el segundo en llegar es de tipo "longshot", esto es, muestra valor n con probabilidad $\frac{1}{n}$ y 0 en otro caso. Luego $OPT = 2 - \frac{1}{n}$ y $ALG = max(1, n \cdot \frac{1}{n}) = 1$, con lo que ALG/OPT converge a $\frac{1}{2}$ cuándo n tiende a infinito.

Prophet Secretary

A diferencia del problema **Prophet** clásico, el problema **Prophet Secretary** aún no está cerrado. Esto es, aún no coinciden la cota inferior y superior que se tienen para su **competitividad**. El mejor algoritmo conocido hasta el momento es 0,669**-competitivo** [2] y la mejor cota superior conocida muestra que es 0,7235**-difícil** [5]. En la sección 2.4 veremos en detalle esta última cota, mientras que en esta sección veremos un algoritmo $1 - \frac{1}{e}$**-competitivo** de Esfandiari, Hajiaghayi, Liaghat y Monemizadeh [4] que al igual que el ejemplo anterior está basado en umbrales de aceptación. En este, muestran que utilizando un único umbral la mejor competitividad que se puede lograr es $\frac{1}{2}$. En cambio ellos proponen un algoritmo que utilice n umbrales, uno para cada paso. Así, el algoritmo comenzará con un umbral alto que irá reduciendo levemente en cada caso. Podemos interpretarlo como que cuándo aún nos quedan muchos agentes por ver somos más exigentes para aceptar a uno de ellos y vamos siendo menos exigentes a medida que los agentes restantes son menos. Estos umbrales se establecen al inicio del proceso y no dependen de los agentes que vayamos observando. A esto se le llama un algoritmo **Non-Adaptive**, pues no toma en cuenta la información que va recibiendo a mediados del proceso, en decir, llegado el k-ésimo agente su decisión sobre aceptarlo o no, no depende de los $k-1$ agentes observados anteriormente.

Dada una secuencia $(\tau_1, \ldots, \tau_n)$ de **umbrales**, un algoritmo utilizará esa secuencia de la siguiente forma: En cada paso $k \in [n]$, nuestro algoritmo acepta el valor X_{π_k} ssi $X_{\pi_k} < \tau_k$. Así, podemos pensar el algoritmo del problema del **Profeta clásico** como uno que utiliza el mismo **umbral** en todos sus pasos, es decir, utiliza la secuencia $(\tau_k)_{k\in[n]}$ con $\tau_k = \frac{1}{2} \cdot OPT$ para todo $k \in [n]$.

Teorema 2.1 (Esfandiari, Hajiaghayi, Liaghat y Monemizadeh [4]) *Sea $(\tau_1, \ldots, \tau_n)$ una secuencia no creciente de **umbrales** tales que:*

(i) $\tau_k = \alpha_k \cdot OPT$ *para todo* $k \in [n]$.

(ii) $\alpha_n = \frac{1}{n+1}$.

(iii) $\alpha_k = \frac{n\alpha_{k+1}+1}{n+1}$ *para todo* $k \in [n-1]$.

*Entonces un algoritmo que utilice esta secuencia de **umbrales** tendrá competitividad mayor o igual a α_1 y además α_1 tiende a $1 - \frac{1}{e} \approx 0{,}63$ cuándo n tiende a infinito.*

2.3. Nuestra cota superior para k grande

Consideremos el Hiper-Grafo $G = (V, E)$ descrito a continuación.

Vértices

Sea m un entero positivo distinto de cero. Sea T el "triangulo inferior" de una matriz de tamaño $m \times m$. Es decir, T corresponde al conjunto de coordenadas (i, j) tales que $i > j$. La condición $i > j$ permite identificar las coordenadas de T con 2 números naturales distintos, sin necesitar especificar un orden entre ellos, por lo que nos referiremos a las coordenadas de T como conjuntos. De esta forma, para $i \neq j$, las notaciones $\{i, j\}$ y $\{j, i\}$ son intercambiables y corresponden a la misma coordenada de T.

Consideraremos que existen 4 vértices en cada coordenada de T, 2 de color Rojo y 2 de color Azul. Así, definimos los vértices de la coordenada $\{i, j\}$ como el conjunto

$$V_{ij} := \{R^0_{ij}, R^1_{ij}, B^0_{ij}, B^1_{ij}\}.$$

Diremos que los vértices de la forma R^x_{ij} y B^x_{ij}, son **rojos** y **azules** respectivamente. Por nuestra discusión anterior R^x_{ij} y R^x_{ji} corresponden al mismo vértice. Análogamente para los vértices azules.

Con esto, nuestro conjunto de vértices será

$$V := \bigcup_{\substack{i,j \\ i>j}} V_{ij} \ .$$

Aristas

Fijando $i \in [m]$, construimos una arista de E seleccionando exactamente 1 vértice rojo y 1 vértice azul de V_{ij}, para cada $j \in [m] \setminus \{i\}$. Llamemos E_i al conjunto de todas las aristas que es posible construir de esta forma. Las selecciones de vértices rojos y azules en cada coordenada las podemos codificar por secuencias $r \in \{0, 1\}^m$ y $b \in \{0, 1\}^m$ respectivamente. Por ejemplo, $r \in \{0\}^m$ representa que hemos seleccionado el primer vértice rojo R^0_{ij} en cada coordenada $\{i, j\}$. Con esto podemos representar los conjuntos E_i como

$$E_i = \{ \bigcup_{j \in [m] \setminus \{i\}} \{R^{r_j}_{ij}, B^{b_j}_{ij}\} | r \in \{0, 1\}^m, b \in \{0, 1\}^m \}.$$

Finalmente definimos

$$E = \bigcup_{i \in [m]} E_i \ .$$

como nuestro conjunto de aristas. Para referirnos a una arista en particular de E, usaremos el triplete (i, r, b), con $i \in [m]$ y $r, b \in \{0, 1\}^m$. Con esto indicamos a cuál conjunto E_i pertenece y qué selección r, b le corresponde, es decir:

$$e(i,r,b) := \bigcup_{j\in[m]\setminus\{i\}} \{R_{ij}^{r_j}, B_{ij}^{b_j}\}.$$

Función de valuación

Tenemos m agentes, cada uno identificado por un índice $i \in [m]$. Cada agente i valúa en 1 a cada arista de un subconjunto A_i de E_i y en 0 a toda arista en $E \setminus A_i$. A_i se construye aleatoriamente de la siguiente forma:

1. Nuestro agente fijará $m-1$ vértices, uno en cada coordenada $\{i,j\}$, $j \in [m] \setminus \{i\}$, y A_i corresponderá al conjunto de aristas en E_i que contengan todos los vértices que han sido fijados.
2. En cada coordenada $\{i,j\}$ tal que $j < i$, se fijará uniformemente al azar uno de los 2 vértices rojos.
3. En cada coordenada $\{i,j\}$ tal que $j > i$, se fijará al uniformemente al azar uno de los 2 vértices azules.

Este proceso podemos representarlo de la siguiente forma. Para cada $i \in [m]$ construimos uniformemente al azar una secuencia $s(i) \in \{0,1\}^m$. Con esto

$$A_i(s(i)) = \left\{ \bigcup_{j\in[m]\setminus\{i\}} \{R_{ij}^{r_j}, B_{ij}^{b_j}\} | r \in \{0,1\}^m, b \in \{0,1\}^m, r_j = s(i)_j \text{ si } j < i,\ b_j = s(i)_j \text{ si } j > i \right\},$$

y la función de valuación $f_i : E \to \{0,1\}$ para cada agente $i \in [m]$ será

$$f_i(e) := \begin{cases} 1 & \text{si } e \in A_i(s(i)). \\ 0 & \text{si no.} \end{cases}$$

Con esto, cada agente $i \in [m]$ en cada casilla V_{ij} **fija al azar** un nodo de un color y puede **elegir** uno de los nodos del otro color. Estará interesado solo en aquellas aristas que contengan todos los nodos que ha fijado.

OPT vs ALG

En el análisis que sigue suponemos que ningún agente elige llevarse una arista que valúa en 0.

Observación: Cada arista $e(i,r,b)$ solo posee nodos de la fila o columna i. Por lo tanto cada nodo de V_{ij} solo puede pertenecer a aristas de la forma $e(i,r,b)$ o $e(j,r,b)$, o sea, solo pueden pertenecer a aristas en $E_i \cup E_j$. Luego, como cada agente i solo valúa aristas en el conjunto $A_i \subset E_i$, sólo los agentes i y j estarán interesados en los nodos de la casilla $\{i,j\}$.

Recordar que cada arista contiene exactamente 1 nodo azul y 1 nodo rojo de cada casilla que le corresponde. Por nuestra construcción de A_i y A_j, s.p.g. $i < j$, de llevarse una arista, en la casilla $\{i, j\}$ el agente i estará obligado a llevarse siempre el mismo vértice azul (fijado al azar al principio) y podrá elegir llevarse cualquiera de los 2 vértices rojos. Por su parte, el agente j, tendrá fijo uno de los 2 vértices rojos y podrá llevarse cualquiera de los 2 vértices azules.

Podemos resumir la observación anterior en el siguiente lema:

Lema 2.2 *Para todo $i, j \in [m]$, con $i \neq j$, decimos que los nodos R^1_{ij} y R^2_{ij} son **complementarios**. Análogamente B^1_{ij} y B^2_{ij} son **complementarios**. En la construcción anterior tenemos que:*

(i) Los agentes i y j están interesados en llevarse nodos de V_{ij} y son los únicos agentes en estarlo. Cada uno, de llevarse una arista, se llevará exactamente un objeto azul y un objeto rojo de cada casilla que le corresponde.

*(ii) En cada casilla V_{ij}, de los agentes i y j, uno de ellos fijará uno de los objetos azules (es decir, no estará interesado en el otro objeto azul) y le será indiferente llevarse cualquiera de los dos rojos y el otro fijará uno de los objetos rojos y le será indiferente llevarse cualquiera de los dos azules. Con ello, una condición necesaria para que ambos se lleven una arista que valoran, es que ambos elijan llevarse el objeto (azul o rojo) **complementario** al que el otro agente ha fijado.*

*(iii) Durante el proceso, sea $(a_1, \ldots, a_k)$ la lista de agentes que se han llevado una arista. Una condición suficiente y necesaria para que el próximo agente i en presentarse pueda llevarse una arista, es que para todo $j \in [k]$ el nodo que a_j ha elegido llevarse en V_{ia_j} sea **complementario** al nodo que i ha fijado.*

Demostración. Ver Observación anterior. □

Teorema 2.3 *En la construcción descrita para el hipergrafo G, en el modelo **Prophet Secretary Matching con agentes**, ningún algoritmo obtiene una ganancia esperada mayor a $\log_2(m+1)$ y $OPT = m$. Por lo tanto es un modelo $\mathcal{O}(\frac{log(k)}{k})$**-difícil** dónde k es el tamaño máximo de las hiper aristas.*

Demostración. Veremos primero la ganancia del óptimo, luego la de un algoritmo cualquiera.

OPT

Consideremos la familia de secuencias $(s(i))_i \in [m]$. Estas codifican los nodos fijados por cada agente i. Puesto que el óptimo conoce estas secuencias desde un principio, en cada casilla V_{ij} les entregará a los agentes i y j el objeto **complementario** al que cada uno de ellos haya fijado. Es decir:

Si $i < j$:

- El agente i tomará de la casilla V_{ij} los nodos $B_{ij}^{s(i)_j}$ (pues es el nodo que ha fijado al azar) y $R_{ij}^{1-s(j)_i}$ (pues es el nodo complementario al que ha fijado j).
- El agente j tomará de la casilla V_{ij} los nodos $R_{ij}^{s(j)_i}$ (pues es el nodo que ha fijado al azar) y $B_{ij}^{1-s(i)_j}$ (pues es el nodo complementario al que ha fijado i).

Si $i > j$ tenemos una situación simétrica.

Luego por *(iii)* del Lema anterior el óptimo podrá asignarle una arista a todos los agentes y obtendrá ganancia m.

ALG

Supongamos que durante el proceso un algoritmo ha asociado aristas a la lista $(a_1, \ldots, a_k)$ de agentes. Al observar a un próximo agente i este le revelará los nodos que ha fijado en cada una de las casillas V_{ia_j} para cada $j \in [k]$. Por *(iii)* del lema anterior, para poder asignarle una arista a i el algoritmo tiene que haber elegido en cada casilla V_{ia_j}, $j \in [k]$, un nodo **complementario** al que i ha fijado. Puesto que todo algoritmo tuvo que elegir estos nodos antes de conocer los nodos fijados por i este tendrá una probabilidad 2^{-k} de haber elegido nodos complementarios en todas las casillas correspondientes. Todo algoritmo tendrá que esperar por lo tanto al menos en esperanza 2^k pasos para poder asignarle una arista a un nuevo agente. Con ello para obtener ganancia s, todo algoritmo debe esperar al menos en esperanza $1+2+4+...+2^{s-1} = 2^s - 1$ pasos, con lo que para m pasos todo algoritmo obtiene en esperanza a lo más ganancia $log_2(m+1)$. □

Correa, Cristi, Fielbaum, Pollner y Weinberg muestran en [1], utilizando un algoritmo de **pricing**, que el modelo **Prophet** con k general[1] es $\mathcal{O}(1/k)$**-competitivo**. Luego, dicha cota inferior también aplica a **Prophet Secretary**. Con esto, tenemos el siguiente resultado:

Teorema 2.4

$$\mathcal{O}(1/k) \leq \textit{Competitividad para } \textbf{\textit{Prophet Secretary}} \textit{ con } k \textit{ general} \leq \mathcal{O}(\log(k)/k).$$

2.4. Cotas superiores basadas en método continuo

En esta sección introducimos un método para calcular cotas superiores en **Prophet Secretary** que llamamos **método continuo**. La idea principal es que, en vez de considerar una permutación generada al azar en el orden de los agentes, pensamos que cada agente selecciona uniformemente al azar un tiempo de llegada en el intervalo [0,1]. Así, el vendedor recorre el intervalo y "se encuentra" con cada agente en orden según el tiempo que cada uno haya elegido. Con esto, observamos que en el límite, ciertas familias de m agentes i.i.d. tienen un comportamiento "continuo" para el vendedor al hacer tender m a infinito.

[1] k es el tamaño máximo de las hiperaristas.

En los 2 próximos ejemplos nuestros agentes serán binarios, esto es, mostrarán un peso determinista con cierta probabilidad o mostrarán peso 0 en otro caso. Decimos que un agente está **activo** cuándo muestra un peso positivo, y decimos que está **inactivo** cuándo muestra peso 0. Por simplicidad solo indicaremos la probabilidad de cada agente de estar **activo**.

Las cotas que veremos a continuación son ya conocidas, la cota $\sqrt{3}-1$ descubierta por en 2019 por Correa, Saona y Ziliotto [2] y la cota 0,7235 en 2023 por Giambartolomei, Mallmann-Trenn y Saona [5]. Sin embargo lo novedoso es el **método continuo** que utilizamos para obtenerlas.

2.4.1. Cota superior de $\sqrt{3}-1$ para Prophet Secretary clásico

Agentes

- m agentes "improbables" de peso m con probabilidad $\frac{1}{m^2}$.
- 1 agente determinista de peso a con probabilidad 1.

Llamaremos A a nuestro agente determinista. Observemos que todo algoritmo puede observar a lo más peso m, por lo tanto un agente "improbable" activo siempre es aceptado. Luego la única decisión que toma un algoritmo es, al recibir A (y aún tener posibilidad de aceptarlo), si aceptarlo o rechazarlo. Para determinar el orden de los agentes utilizamos el siguiente método: a cada agente k le es asignado un tiempo t_k uniformemente aleatorio en el intervalo $[0,1]$. Luego, el orden en que los agentes se presentan será la permutación $\pi(1), \pi(2) \ldots, \pi(m+1)$ tal que $t_{\pi(1)} < t_{\pi(2)} < \ldots < t_{\pi(m+1)}$.

Llamemos $g(r)$ al valor esperado que puede ganar un algoritmo que observa $r \cdot m$ agentes improbables y ningún agente determinista. Es decir,

$$\begin{aligned} g(r) &= m \cdot \mathbb{P}[\text{Existe un agente "improbable" activo}] \\ &= m \cdot (1 - \mathbb{P}[\text{Todos los agentes "improbables" están inactivos}]) \\ &= m \cdot (1 - (1 - \frac{1}{m^2})^{r \cdot m}) \\ &= r + O(\frac{1}{m}). \end{aligned}$$

Notemos además que a medida que m tiende a infinito, en todo intervalo de tiempo $[a,b] \in [0,1]$ observamos una cantidad concentrada en $(b-a) \cdot m$ agentes "improbables". Por lo tanto, en el límite, los agentes "improbables" se comportan entregando ganancia al vendedor según la distancia que este recorre dentro de $[0,1]$ (siempre y cuándo este no haya asignado aún su objeto a ningún agente). Así, por recorrer una distancia $\lambda \leq 1$ con su objeto disponible, el vendedor obtiene en esperanza una ganancia λ.

Con ello, en el límite, cuándo un algoritmo óptimo observe al agente A en tiempo t_A, ganará a por aceptarlo y $1 - t_A$ en esperanza por rechazarlo. Luego

$$\text{Algoritmo óptimo acepta A} \iff a > 1 - t_A \ .$$

Con esto, calculamos la ganancia esperada de un algoritmo óptimo condicionando sobre los posibles tiempos t_A:

$$\begin{aligned}\mathbb{E}[ALG] &= \int_0^1 \mathbb{E}[ALG|t_A = x]\,dx \\ &= \int_0^{1-a} 1\,dx + \int_{1-a}^1 (x+a)\,dx \\ &= 1 + \frac{a^2}{2}.\end{aligned}$$

Por otra parte

$$\begin{aligned}OPT &= m \cdot (1 - \mathbb{P}[\text{Todos los agentes improbables están inactivos}]) \\ &\quad + a \cdot \mathbb{P}[\text{Todos los agentes improbables están inactivos}] \\ &= m \cdot (1 - (1 - \frac{1}{m^2})^m) + a \cdot (1 - \frac{1}{m^2})^m,\end{aligned}$$

lo que converge a $1 + a$ cuándo m tiende a infinito.

Con ello,

$$\frac{ALG}{OPT} = \frac{1 + \frac{a^2}{2}}{1 + a}$$

cuyo mínimo se alcanza para $a = \sqrt{3} - 1$, con lo que $\frac{ALG}{OPT} = \sqrt{3} - 1$.

Con ello tenemos el siguiente resultado:

Teorema 2.5 *El modelo* ***Prophet Secretary*** *es* $\sqrt{3} - 1$***-difícil.***

2.4.2. Cota superior de 0.7235 para Prophet Secretary clásico

Mejoramos la cota $\sqrt{3} - 1$ agregando una nueva familia de agentes i.i.d.

Teorema 2.6 *El modelo* ***Prophet Secretary*** *es* 0,7235***-difícil.***

Agentes

- m agentes "improbables" de peso m con probabilidad $1/m^2$.
- m agentes "frecuentes" de peso b con probabilidad p/m.
- 1 agente A determinista de peso a con probabilidad 1.

Dónde a, b y p son parámetros a optimizar. Notemos que tenemos los mismos agentes que en la cota $\sqrt{3} - 1$ pero hemos agregado además m agentes frecuentes. Notar además que,

si se espera mejorar la cota con esta inclusión, la condición $b > a$ es necesaria, pues de no ser así OPT nunca tomaría un valor b (pues siempre podría tomar a en su lugar) y con ello OPT no tendría ningún beneficio con la inclusión de agentes frecuentes y esta solo podría beneficiar a ALG.

Nuevamente determinamos el orden de los agentes de la siguiente forma: a cada agente k le es asignado un tiempo t_k uniformemente aleatorio en el intervalo $[0, 1]$. Luego, el orden en que los agentes se presentan será la permutación $\pi(1), \pi(2) \ldots, \pi(2m + 1)$ tal que $t_{\pi(1)} < t_{\pi(2)} < \ldots < t_{\pi(2m+1)}$.

Se puede mostrar que cuándo m tiende a infinito este problema es equivalente al siguiente problema continuo.

Los agentes "improbables" tendrán el mismo comportamiento que en el ejemplo anterior, es decir, le darán ganancia al vendedor igual a su distancia recorrida con el objeto aún disponible.

Por otra parte, los agentes "frecuentes" **activos** se comportan como un proceso de Poisson de parámetro p en el intervalo [0,1].

OPT

El óptimo tomará un agente improbable si existe alguno activo, si no, tomará un agente frecuente de existir uno activo y en última instancia tomará al agente de peso A. Como en el ejemplo anterior los agentes improbables le dan al óptimo en esperanza ganancia 1. El que un agente frecuente este activo equivale a que una variable exponencial de parámetro p ocurra en el intervalo [0,1] (Recordar que los agentes frecuentes se comportan como un proceso de Poisson de parámetro p en el intervalo [0,1].

Con ello,

$$\begin{aligned} OPT &= 1 + b \cdot (\int_0^1 p \cdot e^{-tp})\ dt + a \cdot (1 - (\int_0^1 p \cdot e^{-tp})\ dt) \\ &= 1 + b \cdot (1 - e^{-p}) + a \cdot e^{-p}. \end{aligned}$$

ALG

Sea $g(x) : [0, 1] \to \mathbb{R}_+$ lo ganado en esperanza por un algoritmo óptimo en un intervalo de tiempo de largo x, bajo la condición de que el agente A no esté en ese intervalo, suponiendo que el proceso termina tras recorrer ese intervalo, es decir, lo ganado en esperanza en el intervalo $[1 - x, 1]$ dado que ya hemos rechazado A y aún no hemos aceptado ningún agente.

Es claro que $g(0) = 0$. Además si llamamos $h(x)$ a lo ganado en esperanza por OPT bajo las mismas condiciones tenemos que

$$\begin{aligned} g(x) &\leq h(x) \\ &= x + b \cdot (\int_0^x p \cdot e^{-tp})\ dt \\ &= x + b \cdot (1 - e^{-px}). \end{aligned}$$

Esta última expresión es continua y $h(0) = 0$, luego existe x^* suficientemente pequeño tal que $g(x^*) < b$. Cuándo el algoritmo óptimo encuentre un valor b en tiempo $t_b = t + (1-x) \in [1-x, 1]$, este lo aceptará si y solo si $b \geq g(1-t)$, es decir, comparará el valor b a lo ganado en esperanza por rechazarlo. Luego, para todo $x < x^*$, el algoritmo óptimo aceptará el primer valor b en observar. Con ello, para todo $x < x^*$, podemos imaginar que lanzamos una variable exponencial X de parámetro p y, si $X = t < x$, quiere decir que nuestro algoritmo observó un valor b en tiempo t y obtiene ganancia total $t+b$, por otro lado, si $X = t > x$, no observamos ningún valor b y obtenemos ganancia x. Con ello, para todo $x < x^*$:

$$\begin{aligned} g(x) &= \int_0^x p \cdot e^{-pt} \cdot (t+b)\ dt + x \cdot \int_x^\infty p \cdot e^{-pt}\ dt \\ &= \frac{1}{p} \cdot (1 - e^{-xp}) \cdot (1+bp). \end{aligned}$$

Observamos que g es continua y estrictamente creciente para $x < x^*$, luego existe D tal que $g(D) = b$. D corresponde a la distancia máxima desde la que empezamos a aceptar b.

$$\begin{aligned} & g(D) = b \\ \iff & \frac{1}{p} \cdot (1 - e^{-Dp}) \cdot (1+bp) = b \\ \iff & D = \frac{\ln(1+bp)}{p}. \end{aligned}$$

Para $x > D$, nuestro algoritmo rechazará todo valor b observado en $[1-x, 1-D]$ y aceptará el primer valor b observado en $[1-D, 1]$. Con ello, para todo $x > D$,

$$g(x) = (1-D) - (1-x) + g(D) = x + b - D = x + b - \frac{\ln(1+bp)}{p}.$$

Luego, tenemos que

$$g(x) = \begin{cases} \frac{1}{p}(1 - e^{-xp})(1+bp) & x \leq \frac{\ln(1+bp)}{p}. \\ x + b - \frac{\ln(1+bp)}{p} & x > \frac{\ln(1+bp)}{p}. \end{cases}$$

Nuestro algoritmo siempre aceptará a un agente improbable, por lo tanto un algoritmo óptimo se debe enfrentar a 3 tipos de decisión:

- Ver un valor b ya habiendo rechazado al agente A.
- Ver un valor b, sin haber visto aún al agente A.
- Ver al agente A (que siempre muestra valor a).

Si un algoritmo óptimo escoge aceptar un valor en un tiempo t en una de estas 3 situaciones es claro que también lo hará en todos los tiempos s, con $s > t$ (siempre y cuando se este enfrentando a la misma situación). Esto es, puesto que ha determinado que lo ganado en

esperanza en el intervalo $[t, 1]$ que le queda por recorrer, es menor que el valor que le están ofreciendo en ese momento y lo ganado en esperanza en el intervalo $[s, 1]$ será menor que lo ganado en $[t, 1]$ porque es un intervalo más corto. Con esto, definimos un algoritmo que utiliza los siguientes tiempos para tomar sus decisiones. Sean T, T_1 y T_2 tales que:

Algoritmo acepta valor b en tiempo t, ya habiendo visto al agente $A \iff T < t$.
Algoritmo acepta valor b en tiempo t, no habiendo visto aún al agente $A \iff T_1 < t$.
Algoritmo acepta valor a en tiempo $t \iff T_2 < t$.

Tenemos que

$$T = 1 - g^{-1}(b) = 1 - \frac{\ln(1+bp)}{p}.$$

T_1 Será optimizado por el algoritmo según el valor de los parámetros a, b y p.

$$T_2 = 1 - g^{-1}(a) = 1 - \frac{\ln\left(\frac{1+bp}{1+bp-ap}\right)}{p}.$$

Calculamos la ganancia esperada del algoritmo condicionando sobre la posición del valor a. Supongamos que $T_2 < T_1$. En este caso:

$$\begin{aligned}
\mathbb{E}(ALG) =& \mathbb{E}(ALG|t_A \in [0,T]) \cdot \mathbb{P}(t_A \in [0,T]) \\
&+ \mathbb{E}(ALG|t_A \in [T,T_2]) \cdot \mathbb{P}(t_A \in [T,T_2]) \\
&+ \mathbb{E}(ALG|t_A \in [T_2,T_1]) \cdot \mathbb{P}(t_A \in [T_2,T_1]) \\
&+ \mathbb{E}(ALG|t_A \in [T_1,1]) \cdot \mathbb{P}(t_A \in [T_1,1]) \\
\\
=& (T + g(1-T)) \cdot T \\
&+ \int_T^{T_2} y_a + g(1-y_a)\ dy_a \\
&+ \int_{T_2}^{T_1} y_a + a\ dy_a \\
&+ (1-T_1) \cdot T_1 + \int_0^{1-T_1} [\int_0^{y_a} pe^{-pt}(b+t)\ dt + e^{-py_a}(y_a+a)]\ dy_a.
\end{aligned}$$

Con esto,

$$\begin{aligned}
\mathbb{E}(ALG) = \frac{1}{2p^2}\Bigg[& -2 + e^{p(T_1-1)}(2-2ap+2bp) + p\Big(2 - 4b + p + 2a(2+p(T_1-1)) \\
& - 2T_1 - p(T_1-2)(2b+T_1)\Big) - 2(p-1)\ln(1+bp) + \ln(1+bp)^2 \\
& + 2(ap-bp-1)\ln(\frac{1+bp}{1-ap+bp})\Bigg].
\end{aligned}$$

Por otro lado si suponemos $T_1 < T_2$ tenemos que:

$$\begin{aligned}\mathbb{E}(ALG) =& \mathbb{E}(ALG|t_A \in [0,T]) \cdot \mathbb{P}(t_A \in [0,T]) \\ &+ \mathbb{E}(ALG|t_A \in [T,T_1]) \cdot \mathbb{P}(t_A \in [T,T_1]) \\ &+ \mathbb{E}(ALG|t_A \in [T_1,T_2]) \cdot \mathbb{P}(t_A \in [T_1,T_2]) \\ &+ \mathbb{E}(ALG|t_A \in [T_2,1]) \cdot \mathbb{P}(t_A \in [T_2,1])\end{aligned}$$

$$\begin{aligned}=& (T + g(1-T)) \cdot T \\ &+ \int_T^{T_1} y_a + g(1-y_a)\ dy_a \\ &+ (T_2 - T_1) \cdot (T_1 + g(1-T_1)) \\ &+ (1-T_2) \cdot T_1 + \int_{T_2-T_1}^{1-T_1} [\int_0^{y_a} pe^{-pt}(b+t)\ dt + e^{-py_a}(y_a+a)]\ dy_a.\end{aligned}$$

Y con ello

$$\begin{aligned}\mathbb{E}(ALG) = &\frac{1}{2} + b + \frac{1}{p^2} + T_1 - \frac{T_1^2}{2} - \frac{(p-1)\ln(1+bp)}{p^2} + \frac{\ln(1+bp)^2}{2p^2} \\ &+ \frac{e^{T_1-1}(-1-p(1+a+b+bp)+p(1+bp)T_1}{p^2} + \frac{(1+bp)\ln(\frac{1+bp}{1-ap+bp})}{p^2}.\end{aligned}$$

Giambartolomei, Mallmann-Trenn y Saona [5] encontraron los parámetros a, b y p que minimizan $\frac{ALG}{OPT}$, siendo estos $a = 0{,}789$, $b = 1{,}24$ y $p = 0{,}421$

Dados estos, graficamos la ganancia del algoritmo según el tiempo $T1$ que este seleccione:

La ganancia óptima se obtiene para $T1 \approx 0{,}211$ en cuyo caso $ALG \approx 1{,}40643$. Por su parte, para estos parámetros, $OPT \approx 1{,}94397$ y tenemos que

$$\frac{ALG}{OPT} \approx \frac{1{,}40643}{1{,}94397} = 0{,}72348.$$

□

Capítulo 3

Modelo No-Show

3.1. Definición del problema

En el modelo **No-Show** cada agente a, además de su distribución D_a, tiene una probabilidad $s_a \in [0, 1]$ de presentarse en el experimento. El vendedor no sabe qué agentes se presentarán, pero sí conoce los valores s_a y las distribuciones D_a. Al igual que en el modelo **Prophet Secretary** los agentes que deciden presentarse lo hacen en orden uniformemente aleatorio y el proceso termina cuándo todos los objetos han sido asignados, o cuándo todos los agentes se han presentado. Además, en los ejemplos que siguen veremos ejemplos en el caso **single-minded**, es decir, cada agente se identifica con una arista o vértice (podemos ver a los vértices como aristas de tamaño 1) y valúa en 0 al resto de aristas y vértices.

El caso particular en que $s_a = 1$ para todo agente a, es precisamente el modelo **Prophet Secretary**, por lo tanto, el modelo **No-Show** es más difícil que **Prophet Secretary**.

3.2. Cotas superiores

Veamos un par de ejemplos en el modelo **No-Show** que mejoran las mejores cotas superiores conocidas para **Prophet-Secretary**. Utilizamos la variable k para indicar el tamaño máximo de las aristas. En todos ellos la probabilidad de un agente a activarse será siempre 1 (es decir, serán agentes deterministas) y por lo tanto omitiremos mencionarla y en cambio lo relevante será su probabilidad de s_a de presentarse. Podemos pensar que cada uno de estos ejemplos tiene una versión equivalente en el modelo **Prophet Secretary** en la que cada agente a tiene probabilidad $1 - s_a$ de mostrar peso 0. (Es decir, en vez de no presentarse, se presenta y asigna peso 0 a todas las aristas).

3.2.1. Instancia para $k = 1$

Consideremos el caso con $k = 1$, es decir, el vendedor posee un sólo objeto, con 2 agentes:

- Agente A de peso 1 y con $s_A = 1$.
- Agente B de peso $2n$ y con $s_B = \frac{1}{n}$.

La idea será hacer tender n a infinito. Decimos que los agentes como B, que muestran un peso que tiende a infinito con una probabilidad que tiende a 0 son agentes "longshot".

Calculamos la ganancia del óptimo y del algoritmo

OPT

Condicionando sobre si el agente "longshot" se presenta o no, es fácil ver que

$$OPT = \frac{1}{n} \cdot 2n + \frac{n-1}{n} \cdot 1 = 3 - \frac{1}{n},$$

que converge a 3 cuándo n tiende a infinito.

ALG

Es claro que si el primer valor que observa un algoritmo óptimo es $2n$, este lo acepta. Si en cambio observa primero el valor 1, debe elegir el máximo entre 1 y lo ganado en esperanza por rechazarlo. Llamemos "$F1$" al evento de que nuestra primera observación sea 1 y "$A2$" el que el agente "longshot" se haya presentado. Luego,

$$\begin{aligned}\mathbb{P}(A2) &= \frac{1}{n},\\ \mathbb{P}(F1) &= \mathbb{P}(A2) \cdot \frac{1}{2} + (1 - P(A2)) = 1 - \frac{1}{2n}.\end{aligned}$$

Con ello,

$$\begin{aligned}ALG &= (1 - \mathbb{P}(F1)) \cdot 2n + \mathbb{P}(F1) \cdot \text{máx}(1, P(A2|F1) \cdot 2n)\\ &= \frac{1}{2} \cdot \mathbb{P}(A2) \cdot 2n + \mathbb{P}(F1) \cdot \text{máx}(1, \mathbb{P}(A2|F1) \cdot 2n).\end{aligned}$$

Observando que,

$$\mathbb{P}(A2|F1) = \frac{\mathbb{P}(F1|A2) \cdot \mathbb{P}(A2)}{\mathbb{P}(F1)} = \frac{\frac{1}{2} \cdot \frac{1}{n}}{1 - \frac{1}{2n}} = \frac{1}{2n} - \frac{1}{4n^2}$$

tenemos que,

$$ALG = 1 + (1 - \frac{1}{2n}) \cdot \text{máx}(1, 1 - \frac{1}{2n}).$$

Que converge a 2 cuándo n tiende a infinito.

Por lo tanto todo algoritmo tiene competitividad a lo más $\frac{2}{3}$ en este modelo cuándo consideramos un único objeto. Notar que esto **separa** los modelos No Show y Prophet Secretary para un objeto pues en este último existe un algoritmo con competitividad 0.669 (Ver [2]).

Teorema 3.1 *El modelo* ***No Show*** *con* $k = 1$ *posee una instancia* $\frac{2}{3}$***-difícil****, lo que lo* ***separa*** *del modelo* ***Prophet Secretary****.*

3.2.2. Instancia para $k = 2$

Consideremos el grafo $G = (\{a, b\}, ab)$, es decir, una única arista. Identificamos un agente con cada vértice y un agente con la arista ab. [1]

- Agente a de peso 1 y con $s_a = 1$.
- Agente b de peso 1 y con $s_b = 1$.
- Agente ab de peso Ln y con $s_{ab} = \frac{1}{n}$.

Con $L > 2$ una constante por determinar. La arista ab corresponde a un agente de tipo "longshot" y los 2 vértices son agentes deterministas.

Es claro que todo algoritmo óptimo siempre se llevará al agente ab si este se presenta. El algoritmo óptimo debe por lo tanto decidir entre 3 estrategias posibles:

1. Tomar ambos vértices cuándo estos se presentan.
2. Tomar solo el segundo vértice que se presente.
3. No tomar ninguno de los vértices que se presente.

En la primera estrategia, nuestro algoritmo tendrá ganancia 2 excepto en el caso en que ab se presente y además sea el primero en observarse. Con ello llamando $E1$ a la ganancia esperada con la primera estrategia tenemos que:

$$\begin{aligned} E1 &= s_{ab} \cdot \frac{1}{3} \cdot Ln + (1 - s_{ab}) \cdot 2 \\ &= \frac{L}{3} + 2 - \frac{2}{n}, \end{aligned}$$

que converge a $2 + \frac{L}{3}$ cuándo n tiende a infinito.

El análisis para las otras 2 estrategias es análogo. Si las llamamos $E2$ y $E3$ respectivamente tenemos que cuándo n tiende a infinito $E2 = 1 + \frac{2L}{3}$ y $E3 = L$. Con ello

$$ALG = \text{máx}(E1, E2, E3).$$

y es fácil ver que

$$OPT = 2 \cdot (1 - s_{ab}) + s_{ab} \cdot Ln = 2 \cdot (1 - \frac{1}{n}) + L$$

que converge a $2 + L$ cuándo n tiende a infinito. Tomando $L = 3$ tenemos que $ALG = E1 = E2 = E3 = 3$ y $OPT = 5$ y por lo tanto el modelo **No-Show** para $k = 2$ es $\frac{3}{5}$**-difícil**.

[1] G es en realidad un hiper grafo con aristas $E = \{a, b, ab\}$ pero lo anotamos asi por simplicidad.

Teorema 3.2 *El modelo* ***No Show*** *con* $k = 2$ *posee una instancia* $\frac{3}{5}$***-difícil.***

3.2.3. Instancias para $k = 2$ y n tendiendo a infinito

Una extensión natural del ejemplo anterior es considerar K_n en vez de K_2. Analicemos dicho caso. Sea $G(V, E) = K_n$ el clique de tamaño n dónde n es fijo. Por agentes tenemos

- Por cada vértice $v \in V$, agentes A_v de peso $\frac{1}{n}$ y con p_{s_v}=1.
- Por cada arista $e \in E$, agentes A_e de peso $\frac{L}{m\varepsilon}$ y con $s_{A_e} = \varepsilon$,

con L una constante por determinar, $m = \binom{n}{2}$ el número de aristas y ε una variable que haremos tender a 0.

OPT

Para ε suficientemente pequeño $\frac{L}{m\varepsilon} \gg 1$, por lo que si hay una arista activa el óptimo siempre la toma y gana aproximadamente $\frac{L}{m\varepsilon}$ (como ε tenderá a 0 lo ganado por todos los vértices será despreciable en comparación).

Notar que podemos despreciar el caso en que existen 2 o más aristas activas:

$$\begin{aligned}
&[\text{Ganancia por llevarse 2 o más aristas}] \cdot \mathbb{P}(\text{Existen 2 o más aristas activas}) \\
&\leq \frac{mL}{m\varepsilon} \cdot (1 - \mathbb{P}(\text{Ninguna arista activa}) - \mathbb{P}(\text{Exactamente 1 arista activa})) \\
&= \frac{L}{\varepsilon}(1 - (1-\varepsilon)^m - m\varepsilon \cdot (1-\varepsilon)^{m-1}),
\end{aligned}$$

que converge a 0 cuándo ε tiende a 0. Intuitivamente esto ocurre porque la ganancia por más de una arista crece linealmente mientras que la probabilidad de que esto ocurra decae geométricamente.

Si no hay ninguna arista activa el óptimo se quedará con todos los vértices y su ganancia será 1.

Con esto

$$\begin{aligned}
\mathbb{E}(OPT) &= \mathbb{P}(\text{Ningún } A_e \text{ activo}) \cdot 1 + \mathbb{P}(\text{Algún } A_e \text{ activo}) \cdot \frac{L}{m\varepsilon} \\
&= (1-\varepsilon)^m \cdot 1 + (1 - (1-\varepsilon)^m) \cdot \frac{L}{m\varepsilon}.
\end{aligned}$$

lo que converge a $1 + L$ cuándo ε tiende a 0.

Analizaremos ALG solo para el caso $n \to \infty$

Sea un algoritmo que siempre acepta una arista si está activa. Respecto a los vértices rechaza una primera porción de ellos y acepta todos los que lleguen pasado cierto punto.

Digamos que acepta la última $\alpha \in [0,1]$ porción de vértices observados, es decir, acepta los últimos S vértices con $\frac{S}{n} = \alpha$.

Al igual que para el óptimo basta con considerar solo los casos en que no hay ninguna arista activa y el caso en que hay exactamente 1 activa. Con ello, cuándo n tiende a infinito tenemos que $\mathbb{E}(ALG)$ es igual a

$$
\begin{aligned}
&[\text{Ganancia por vértices aceptados}] \\
&+ [\text{De estar activa, recibimos arista en tiempo } [0, 1-\alpha]] \cdot \mathbb{P}[\text{Recibimos arista en tiempo } [0, 1-\alpha]] \\
&+ [\text{De estar activa, recibimos arista en tiempo } [1-\alpha, 1]] \cdot \mathbb{P}[\text{Recibimos arista en tiempo } [1-\alpha, 1]] \\
&= \alpha + (1-\alpha) \cdot L + L \cdot \int_0^{\alpha} (1-t)^2 \, dt \\
&= +L\alpha - L\alpha^2 + L\frac{\alpha^3}{3}.
\end{aligned}
$$

El término $L \cdot \int_0^{\alpha}(1-t)^2 \, dt$ se obtiene condicionando sobre el tiempo t_e en que recibimos una arista e activa (de existir esta) dado que la recibimos en $t_e = (t + 1 - \alpha) \in [1-\alpha, 1]$. Para este t_e, habremos ya aceptado una porción t de vértices y por lo tanto la probabilidad de poder aceptar la arista corresponderá a la probabilidad de que ambos vértices incidentes a esta pertenezcan a la porción $(1-t)$ de vértices aún libres, es decir, $(1-t) \cdot (1-t) = (1-t)^2$.

El valor para α que maximiza esta expresión (y por lo tanto la ganancia del algoritmo) es

$$\alpha = \frac{L - \sqrt{L^2 - L}}{L}.$$

Luego minimizando la expresión $\frac{ALG}{OPT}$ sobre L obtenemos que para $L \approx 1{,}13$, $\frac{ALG}{OPT} \approx 0{,}661$.

Observamos por lo tanto que este caso no mejora la cota $\frac{3}{5}$ obtenida anteriormente, sin embargo tiene interés en cuánto puede ser más naturalmente extendido al caso **Prophet Secretary** con $k = 2$. Como se describió anteriormente, remplazaríamos las probabilidades de no presentarse con el caso en que el agente se presenta y asigna peso 0 a todas las aristas. Aún está abierto hacer un análisis preciso de esto último y determinar si la cota 0,661 podría conservarse.

Conclusión

En esta memoria mostramos que para k general el modelo **Prophet Secretary Matching con agentes** es $\mathcal{O}(\frac{log(k)}{k})$**-difícil**. Introducimos un nuevo método que llamamos **método continuo** y lo utilizamos para volver a probar ciertas cotas en **Prophet Secretary**. Por último introducimos un nuevo modelo, que llamamos **No Show** y mostramos que en el caso de 1 objeto este y **Prophet secretary** están **separados**.

Como preguntas abiertas, sería muy interesante si las cotas para k general ,que son $\mathcal{O}(\frac{1}{k})$ como cota inferior y $\mathcal{O}(\frac{log(k)}{k})$ como cota superior, pueden cerrarse. Una observación interesante es que para la cota $\mathcal{O}(\frac{log(k)}{k})$, cada objeto es disputado por solo 2 agentes. Esto es potencialmente una limitación innecesaria, y quizás la cota puede mejorarse haciendo que la disputa de objetos entre agentes forme una estructura más compleja.

En el caso del modelo **No Show** sería interesante determinar si el modelo está separado de **Prophet secretary** para k general.

Por último, para el problema **Prophet secretary** está aún abierto si la cota 0,7235 puede ser mejorada agregando una segunda familia de agentes i.i.d. de tipo "frecuente".

Como trabajo futuro planteamos extender el ejemplo que lleva a la cota 0,7235 al caso matching. Además comprobar si el ejemplo presentado en la sección 3.2.2. del modelo **No Show** puede funcionar también en el caso **Prophet Secretary**, potencialmente haciendo ciertas modificaciones.

Bibliografía

[1] José Correa, Andrés Cristi, Andrés Fielbaum, Tristan Pollner, and S. Matthew Weinberg. Optimal item pricing in online combinatorial auctions. In Karen Aardal and Laura Sanità, editors, *Integer Programming and Combinatorial Optimization*, pages 126–139, Cham, 2022. Springer International Publishing.

[2] Jose Correa, Raimundo Saona, and Bruno Ziliotto. Prophet secretary through blind strategies, 2019.

[3] José R. Correa, Patricio Foncea, Ruben Hoeksma, Tim Oosterwijk, and Tjark Vredeveld. Posted price mechanisms for a random stream of customers. *Proceedings of the 2017 ACM Conference on Economics and Computation*, 2017.

[4] Hossein Esfandiari, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Morteza Monemizadeh. Prophet secretary, 2015.

[5] Giordano Giambartolomei, Frederik Mallmann-Trenn, and Raimundo Saona. Prophet inequalities: Separating random order from order selection, 2023.

[6] Ulrich Krengel and Louis Sucheston. Semiamarts and finite values. *Bulletin of the American Mathematical Society*, 83(4):745 – 747, 1977.

[7] Brendan Lucier. An economic view of prophet inequalities. *SIGecom Exch.*, 16(1):24–47, sep 2017.