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

COLUSIÓN TÁCITA DE ALGORITMOS Q-LEARNING EN EL DILEMA DEL PRISIONERO REPETIDO CON ESTADOS MARKOVIANOS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN ECONOMÍA APLICADA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

HUGO BERNARDO LEYTON ALLENDE

PROFESOR GUÍA:
JUAN ESCOBAR CASTRO

MIEMBROS DE LA COMISIÓN:
CARLOS NOTON NORAMBUENA
RONALD FISCHER BARKAN

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL GRADO DE MAGISTER EN ECONOMIA APLICADA
Y AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL
POR: **HUGO BERNARDO LEYTON ALLENDE**
FECHA: 2021
PROF. GUÍA: JUAN ESCOBAR CASTRO

COLUSIÓN TÁCITA DE ALGORITMOS Q-LEARNING EN EL DILEMA DEL PRISIONERO REPETIDO CON ESTADOS MARKOVIANOS

La creciente incorporación de nuevas tecnologías en nuestra vida cotidiana esta moldeando nuestro estilo de vida. Algoritmos nos ofrecen contenido, nos recomiendan productos, planifican nuestro viaje e incluso nos califican. Al igual que las personas, las firmas delegan cada día más sus decisiones a algoritmos. La incorporación de algoritmos de pricing puede traer efectos no deseados para el publico; precios astronómicos, discriminación arbitraria e incluso colusiones.

Esta tesis estudia la capacidad de algoritmos de Q-learning de coludirse tácitamente, mediante el monitoreo directo e indirecto utilizando variables de mercado comunes. Se evalúan esquemas de estrategias compatibles con colusión tácita y equilibrios cooperativos. Además se estudia la convergencia a ellas por parte de estos algoritmos.

Si bien no se aprecia convergencia a las estrategias presentadas, aun así se observan precios colusivos, la convergencia de los algoritmos ignora las nociones de racionalidad propuestas por el concepto de equilibrio, obteniendo precios colusivos expuestos a eventuales desvíos del competidor.

Se concluye que las restricciones de información no siempre son suficientes para evitar precios supracompetitivos. Es más, las restricciones de información pueden facilitar colusiones tácitas en mercados de baja elasticidad o con estados de la demanda dependientes intertemporalmente. En estos escenarios se facilita un monitoreo indirecto, en el cual los algoritmos pueden identificar las acciones del rival sin necesidad de tener su historial de precios. Estos escenarios constituyen una excepción que debe ser considerada para garantizar una regulación exitosa.

Esta tesis muestra mecanismos mediante los cuales los algoritmos son capaces de sostener colusiones tacitas sin monitoreo mutuo. La capacidad de estos algoritmos de monitorear indirectamente a sus rivales deja obsoleto el concepto de colusión que es planteado en la regulación vigente. Debido a esto, esta tesis propone que las nuevas regulaciones debieran apuntar a limitar el monitoreo indirecto entre estos algoritmos; traduciéndose esto en restringir la memoria de los algoritmos a variables de mercado y en transparentar las estrategias que estos algoritmos aplican al menos en términos de dar a conocer sus parámetros. Además, esta tesis señala variables de mercado que pueden ser utilizadas como medio de monitoreo indirecto.

Queda para trabajos posteriores el estudio de entrada de competidores y el incremento en el espacio de acciones de los jugadores.

Agradecimientos

Me gustaría agradecer a mi familia, que siempre brindaron cariño en los momentos más difíciles, por estar siempre presentes y brindar su apoyo durante todos mis años de estudio. En especial a mis padres, Sonia y José Miguel por su esfuerzo para que nunca me hiciera falta algo y motivarme a seguir la carrera. Y a mis hermanos, Francisca, Sonia y Jose Miguel, por ser siempre un excelente ejemplo a seguir.

A mis amigos, Rocío, Danilo, Manuel, Sebastián, Nicolás y Matías, quienes siempre me proporcionaron de ánimo y buenos deseos. En especial a Rocío, Nicolás y Danilo, quienes colaboraron en la redacción de este trabajo y brindaron sugerencias para su desarrollo. Tambien fueron indispensables las incontables horas de conversación con Rafael, Diego, Flora, Mauricio y Guillermo, haciendo el trabajo de tesis y la cuarentena mucho más llevadero.

Bajo ninguna circunstancia agradecería a Pezoa, quien no aportó absolutamente nada.

A los profesores Fischer y Noton, miembros de mi comisión por su muy buena disposición para brindar ayuda en todo momento y guiarme a lo largo del desarrollo de este trabajo.

Finalmente, un gran agradecimiento al Profesor Escobar, por su excelente disposición y paciencia. Mucha paciencia.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Introducción

El uso de algoritmos de fijación de precios y herramientas tecnológicas como el big data, se difunde cada vez más en la vida cotidiana, es común recurrir a servicios automatizados de atención al cliente, usar buscadores online, llevar a cabo planificación de viajes en automóvil y ver anuncios con un nivel de personalización excepcional. Esta difusión es generalizada y en buena parte imperceptible para el publico, quienes toman decisiones siendo influenciados por algoritmos, por ejemplo, la compañía Netflix afirmó en 2018 que el 80 % de los videos vistos en su plataforma habían sido el resultado de recomendaciones algorítmicas. Las firmas han delegado a algoritmos buena parte de sus decisiones comerciales, esto está cambiando el panorama competitivo en el que operan trayendo consecuencias para el consumidor final. Si bien la magnitud de este fenómeno es en gran medida desconocido, un número creciente de empresas está utilizando algoritmos informáticos para mejorar sus modelos de precios, personalizar los servicios y predecir las tendencias del mercado. Ejemplos de estos son los algoritmos utilizados para discriminación de precios por las aerolíneas, el algoritmo de tarifas de Uber, el trading de alta frecuencia y los algoritmos de fijación de precios de los retailers de Amazon. El 50 % del volumen de venta en los mercados de equities era realizado por algoritmos al año 2011 tomando una relevancia tal que está afectando los precios de los inmuebles próximos a los mercados en una competencia por disminuir latencias. Hoy los traders deben conformarse con apuestas a largo plazo pues los arbitrajes a corto plazo pertenecen a algoritmos.

Si bien la incorporación de algoritmos está indudablemente asociada a eficiencias significativas, que benefician tanto a las empresas como a los consumidores en términos de productos y servicios nuevos, mejores y más personalizados, hay distorsiones generadas por estos algoritmos que se dejan ver. En la primavera de 2011, dos minoristas en línea ofrecieron copias del libro de texto de Peter Lawrence The Making of a Fly en Amazon por $ 18.651.718,08 y $ 23.698.655,93 (más $3,99 de envío) respectivamente. Este fue el resultado de que ambos vendedores utilizaron algoritmos de precios automatizados. Todos los días, el algoritmo utilizado por el vendedor 1 establece que el precio del libro sea 0.9983 veces el precio cobrado por el vendedor 2. Más adelante en el día, el algoritmo del vendedor 2 ajusta su precio para que sea 1.27059 veces el del vendedor 1. Los precios aumentaron exponencialmente y permaneció por encima de un millón de dólares durante al menos diez días, hasta que uno de los vendedores se dio cuenta y ajustó su precio a $ 106.23. En diciembre de 2014 la aseguradora Allstate se vio envuelta en numerosas acusaciones de cargar primas ilegales a sus clientes de forma arbitraria; Allstate permitía descuentos de hasta un 90 % y sobreprecios de hasta un 800 %, al punto que cobraba un 30 % más si un cliente nacía en el 12 de enero de 1968 en lugar del 9 de abril del mismo año, todo esto como consecuencia de la implementación de un algoritmo de optimización de tarifas. Existen ejemplos aún mayores, el 6 de mayo de 2010 el

índice DJIA (Dow Jones Industrial Average) se desplomaba un 9 % para recuperar su valor 20 minutos después, esto como consecuencia de una sobrerreacción de algoritmos de trading a las noticias llegadas de la crisis de deuda griega. El 6 de julio de 2016 de la libra esterlina caía un 6.1 % para recuperarse en minutos, estas fuertes variaciones han sido bautizadas flash crashes y pueden llegar a generar millones de dolares en pérdidas a los inversores.

Las causas de estas distorsiones podrían no ser accidentales, este uso generalizado de algoritmos ha generado preocupaciones sobre posibles comportamientos anticompetitivos, ya que pueden facilitar que las empresas logren y mantengan la colusión sin ningún acuerdo formal o interacción humana. Los distorsiones mencionadas podrían ser consecuencia de manipulaciones intencionales, en abril del 2015 se dió el primer caso de la primera acusación de colusión en el ecommerce[1], según la acusación, los acusados fijaron los precios de ciertos productos vendidos en Estados Unidos a través del portal de Amazon. Para implementar sus acuerdos, los acusados adoptaron algoritmos de precios específicos con el objetivo de coordinar los cambios en sus respectivos precios. Frente a este caso el fiscal de la División Antimonopolio del Departamento de Justicia Bill Baer declaró: "No toleraremos conductas anticompetitivas, ya sea que ocurran en una habitación llena de humo o por Internet utilizando complejos algoritmos de precios. Los consumidores estadounidenses tienen derecho a un mercado en línea gratuito y justo, así como a empresas físicas ". Acerca de la capacidad de colusion y la falta de transparencia de estos algoritmos, la división de competencia de la OCDE publicó Algorithms And Collusion (OECD, 2017), documento en el cual destaca la rapidez con la que los algoritmos afectan las condiciones del mercado y proporcionan poderosos mecanismos de monitoreo de precios, por lo que se hace presente la necesidad de un nuevo marco regulatorio al respecto. La autoridad de mercados y competencia del Reino Unido en su publicación Algorithms: How they can reduce competition and harm consumers sigue en la misma linea.

El presente trabajo se inspira en el estudio de Calvano, Calzolari, Denicol'o y Pastorello (2019), quienes muestran que algoritmos q-learning fijan precios supracompetitivos incluso sin tener la capacidad de comunicarse entre ellos. Se encuentra que estos precios son sostenidos por estrategias de castigo finito con gradual retorno a precios suprecompetitivos, además encuentran que entre los factores que favorecen la obtención de precios altos esta el largo de la memoria que los algoritmos poseen. Esta memoria induce estrategias que condicionan en los precios pasados que el algoritmo competidos ha escogido. Considerando el impacto de este resultado, resulta natural esperar que un regulador prohibiera la implementación de algoritmos que pueden conocer los precios históricos.

Frente a estos hechos es claro que los algoritmos fijadores de precio dan una ventaja al vendedor, bien implementados, y por tanto una institución reguladora se vuelve necesaria. Hasta el momento los algoritmos mencionados condicionan sus precios en los precios pasados del rival (con excepción el de Allstate), en particular no es claro que los precios pasados tengan una relación con los precios presentes. Existe gran variedad de información con la que se puede alimentar un algoritmo, cantidad de visitas en la web, tiempo en venta, cantidad de ventas históricas y parámetros del consumidor, ¿Qué valor aporta incorporar una historia de precios?. Es claro que un algoritmo que condiciona en precios históricos estaría monitoreando a sus rivales, permitiendo así la formación de carteles cuya colusión es, al menos, tácita.

[1] https://www.justice.gov/opa/pr/former-e-commerce-executive-charged-price-fixing-antitrust-divisions-first-online-marketplace

Debido a esto es natural pensar que un regulador debería prohibir la incorporación de precios pasados en los sets de información de los algoritmos, pero ¿sería esta una medida exitosa?

El objetivo del presente trabajo es identificar bajo que condiciones limitar la información de los algoritmos trae condiciones positivas para los precios obtenidos, obtener las estrategias resultantes de estas restricciones y entregar las bases para una propuesta regulatoria. Adicionalmente se busca anticipar reacciones adversas a estas medidas, proponiendo una estrategia de equilibrio que pueda permitir resultados colusorios a pesar de esta restricción. Por medio de simulaciones se busca verificar si los algoritmos alcanzan este equilibrio de forma espontánea. El modelo de equilibrio propuesto será dependiente de la estructura de mercado, en particular la correlación intertemporal de variables, por tanto, también es objetivo de esta tesis identificar variables que permitan la coordinación entre los algoritmos. La motivación detrás de este escenario es que, una vez captado este efecto por los agentes, este puede ser utilizado como medio de comunicación encubierta para sostener una colusión. Si bien la presente tesis no busca estudiar estos mecanismos de comunicación, si busca identificar hasta qué punto los algoritmos en base a Q-Learning pueden coordinarse por este medio, llegando así a una colusión tacita.

No se estudiará en la presente tesis las decisiones de implementación de estos algoritmos llevadas a cabo al interior de las firmas. Esto queda como desafío para futuros estudios, extendiendo así el trabajo de Salcedo (2015) que muestra que en los juegos repetidos que la elección de algoritmos llega a resultados colusivos de forma dominante cuando los algoritmos pueden ver los precios de sus pares (esto es, las personas encargadas de implementar los algortimos preferirán los que tiendan a coludirse), la segunda parte de este trabajo se vuelve útil para extender este resultado una vez se tiene una nueva forma de comunicación tácita entre los algoritmos que permite a los agentes encargados de ponerlos en marcha y refinar su elección tal como Salcedo propone.

Capítulo 1

Antecedentes

1.1. Estudios previos

Con respecto al dilema del prisionero, Axelrod (1981) estudia como la cooperación surge a de forma espontánea entre participantes egoístas sin la necesidad de una autoridad central, proponiendo la estrategia tit for tat como estrategia óptima. Tambien presenta el estudio en donde se propone el perfil de pagos utilizados en esta tesis (Axelrod, 1980). Siguiendo en esta linea Carmel (1996) obtiene la estrategia óptima del dilema del prisionero iterado para el juego propuesto por axelrod generalizando tit for tat. Moriyama (2009) estudia cómo se desenvuelven los algortimos Q-learning en dilema del prisionero repetido, estudiando qué características deben tener los valores del juego para permitir resultados colaborativos. Deriva un teorema que entrega la cantidad de veces que se debe repetir la jugada de colaborar para que el valor Q relativo al desvío sea menor al relativo a colaborar. De esta forma caracteriza las instancias en las cuales los algoritmos Q-learning llegan a resultados colusorios en el juego repetido.

Con respecto a los algoritmos Q-learning Watkins y Dayan (1992) obtienen las condiciones por medio de las cuales estos algoritmos convergen a las políticas óptimas. Obtiene como condición suficiente que los algoritmos posean una tasa de exploración finita perteneciente a una clase conocida como Greedy in the Limit with Infinite Exploration (GLIE), la cual establece que la probabilidad de jugar toda acción en cada estado es siempre positiva y la probabilidad de escoger la acción avara[1] tiende a 1. Hingston (2004) estudia como los agentes Q-learning interactúan en pares sin previo conocimiento. Muestra que los algoritmos Q-learning son capaces de explotar sistemáticamente a sus competidores no adaptativos.

Ittoo (2017) estudia las limitaciones de algoritmos Q-learning y los desafíos que estos enfrentan para su implementación en distintos mercados ofreciendo una introducción al problema de colusión tacita. Si bien no niega la posibilidad de colusión tacita potencial, si considera que la implementación de esta trae desafíos que se encuentran lejos del alcance de los oferentes. Tales desafíos son la especificación de preferencias, la formalización del ambiente en términos de datos, las preferencias no estacionarias de los agentes, la escalabilidad y la tasa de exploración necesaria en un mercado real. Ezrachi (2017) hace un estudio de la

[1] Se entiende por acción avara a aquella que reporta la mayor utilidad de continuación en ese periodo

colusión tacita desde un punto de vista regulatorio, resaltando lo obsoleta de las legislaciones actuales frente al desafío de la colusion tacita y los incentivos a implementarlos que esta obsolescencia implica en la elección de políticas de precios por parte de las firmas. Concluye que este vacío legal invita a las firmas a implementar algoritmos en lugar de escoger los precios individualmente.

Con respecto a la presencia de algoritmos en el mercado se encuentran los estudios a grandes firmas, Chen (2015) que lleva a cabo un estudio del algoritmo de elección de precios de Uber "surge pricing". Muestra como este algoritmo puede presentar una respuesta instantánea a la demanda, manipulando los precios en desmedro de los consumidores y levantando interrogantes acerca de la regulación y transparencia. Chen (2016) desarrolla una metodología para detectar los clientes que utilizan algoritmos y lo utiliza para revisar el mercado de venta de productos en Amazon. En su estudio encuentra que los vendedores que utilizan algoritmos poseen más chances de hacer ventas que sus contrapartes que no utilizan algoritmos anticipando de esta forma una adopción general de algoritmos por parte de los vendedores en el largo plazo. Finalmente llega la conclusión de que el uso de algoritmos no necesariamente implica una baja de precios para los consumidores. La fuerte y rápida variación de precios genera la necesidad de plataformas de monitoreo de precio automáticas por parte de los compradores. Boehmer (2018) muestra que los precios de las acciones transadas por algoritmos de trading de alta frecuencia poseen una alta correlación en sus valores, dando así a conocer como estos algoritmos tienen el poder de manipular el mercado.

Salcedo (2015) considera un modelo dinámico en el que las firmas juegan escogiendo algoritmos los cuales escogen precios durante un periodo relativamente largo. Los supuestos claves son que los algoritmos ven los precios pasados del rival y que al instante de la revisión ellos pueden descifrar a la perfección cuál es el algoritmo del rival. De esta concluye que todo equilibrio en subjuego perfecto del juego repetido tendrá como elección algoritmos que intenten coludirse (estrategia tit for tat extendida) para cada actualización de algoritmos, llegando a precios colusivos con probabilidad 1 en el largo plazo.

Calvano (2019) estudia el comportamiento de algoritmos Q-Learning que escogen precios en modelos oligopólicos. Considera modelos en donde la demanda no posee comportamiento markoviano, sino más bien es la llegada de clientes según parámetros fijos. Además, en su estudio los algoritmos poseen acceso a los precios de los rivales. Sus resultados revelan que los algoritmos convergen a precios por sobre los precios competitivos de forma consistente, con ello además las matrices de decisión generadas por los algoritmos son específicas de sus pares formando un comportamiento similar al de las estrategias de castigo finito en caso de desvíos y estrategias tit for tat en varias etapas.

Por ultimo Harrington (2017) propone un mecanismo para la detección de colusión tácita de algoritmos, este consiste en identificar por ingeniería inversa la presencia de esquemas de castigo y recompensa, prohibiendo los sets de algoritmos que respondan con precios supracompetitivos.

1.2. Conceptos acerca de los juegos repetidos

En lo que sigue se considera un juego repetido en el cual interactúan 2 algoritmos Q-learning decidiendo precios periodo a periodo, sin poseer ninguna información adicional acerca del mercado, es decir, deciden por ensayo y error. Cabe mencionar que no incorporan la existencia de un rival, al cual entonces lo consideran parte de su entorno.

En cada periodo los algoritmos poseen un set de información disponible, la modificación de este constituye el principal sujeto de estudio de la presente tesis. El set de información considera el estado de la demanda y los precios históricos jugados por el rival y por si mismo entre otros. Finalmente estos sets inducen estados sobre los cuales los algoritmos determinan una estrategia.

En lo que sigue se enuncian definiciones relevantes.

1.2.1. Equilibrio en subjuego perfecto (EPS)

El perfil σ es equilibrio en subjuego perfecto si para toda historia h^t, $\sigma \mid_h^t = (\sigma_1 \mid_h^t, ..., \sigma_n \mid_h^t)$ es equilibrio de Nash del juego repetido.

1.2.2. Estrategia markoviana

Una estrategia $\sigma_i : \cup_t H^t \to A_i$ es markoviana si para todo set de historias h^t y $\hat{h}^t$, si $s\left(h^t\right) = s\left(\hat{h}^t\right)$, luego:

$$\sigma_i\left(h^t\right) = \sigma_i\left(\hat{h}^t\right)$$

En otras palabras, la decisión tomada por el agente hoy no tiene memoria.

1.2.3. Equilibrio markoviano perfecto (EMP)

Un equilibrio markoviano perfecto (EMP) es un perfil de estrategias markovianas que es equilibrio perfecto en subjuego.

1.3. Q-Learning

El Q-Learning es un tipo de algoritmo de aprendizaje por reforzamiento, propuesto por Watkins (1992) para abordar procesos de decisión markoviano. Dado un proceso estacionario, un set de estados finitos y un conjunto de posibles acciones finitas e independientes de estos estados, el algoritmo Q-learning es capaz de converger a una política óptima si la probabilidad de visitar cada estado es siempre positiva aunque con limite 0 en un horizonte infinito.

En principio podría surgir la duda de se si utilizar este tipo de algoritmos o no. Siguiendo la argumentación de Calvano (2019) se esgrimen 3 razones. La primera es que es muy popular entre académicos de las ciencias de la computación, además la convergencia al óptimo se encuentra demostrada para estos algoritmos. Segundo, bastan pocos parámetros para implementar cada experimento, dejando de lado arbitrariedades y haciendo claras sus implicancias. En tercer lugar comparte la misma arquitectura de otros programas que han obtenido buenos resultados.

1.3.1. Algoritmos Q-Learning en el juego repetido

El objetivo de cada algoritmo es obtener una estrategia que maximice su flujo presente descontado de ingresos, esto es:

$$E\left[\sum_{t=0}^{\infty} \delta^t \pi_t\right]$$

Considerando δ el factor de descuento y π_t la utilidad en el periodo t. En otras palabras se esta buscando una estrategia que permita resolver el problema de programación dinámica cuya función de valor de Bellman es como sigue.

$$V(s) = \max_{a \in A} \left\{ E[\pi \mid s, a] + \delta E\left[V\left(s_{t+1}\right) \mid s, a\right] \right\}$$

El al algoritmo Q-learning resuelve este problema por medio de una matriz cuyas celdas corresponden al valor presente descontado de tomar una acción $a \in A$ en un estado $s \in S$. A ella se de denomina matriz Q y se encuentra definida de forma implícita como:

$$Q(s, a) = E(\pi \mid s, a) + \delta E\left[\max_{a_{t+1} \in A} Q\left(s_{t+1}, a_{t+1}\right) \mid s, a\right]$$

Lo cual es equivalente a escribir la ecuación de Bellman por medio de la identidad: $V(s) \equiv \max_{a \in A} Q(s, a)$. En otras palabras la convergencia de la matriz Q nos da una aproximación de los valores de continuación que resuelven el problema del agente. Una vez obtenida la matriz Q se puede verificar la calidad de esta aproximación por medio de resolver el sistema de ecuaciones obtenido. A la diferencia entre el Q obtenido y V la llamaremos Q-loss siguiendo la convención de Calvano.

Los algoritmos tratan vía ensayo y error con el entorno, cada periodo reciben un pago el cual es incorporado a la matriz Q por la siguiente regla

$$Q_{t+1}(s, a) = (1 - \alpha) Q_t(s, a) + \alpha \left[\pi_t + \delta \max_{a_{t+1} \in A} Q_t\left(s_{t+1}, a_{t+1}\right)\right]$$

En $t = 0$ se inicia con una matriz a elección y cada periodo se actualiza una celda a la vez. Esta regla de asignación es una combinación convexa entre el valor obtenido y el valor antiguo en la matriz Q. Al parámetro α de esta combinación lo llamaremos tasa de aprendizaje.

Dada la matriz Q_0 inicial, el algoritmo aprenderá de las decisiones que tome. Para garan-

tizar que el aprendizaje sea convergente a el valor óptimo de Q es necesario que recorra todas las posibles acciones una cantidad mínima de veces. Para ello se implementa una probabilidad ϵ de jugar al azar. A la fracción de tiempo en la cual se esta jugando al azar la llamaremos fase de exploración. Por otra parte la fracción de tiempo en la que no se esta explorando se le denomina fase de explotación, esta consiste en elegir la acción que reporta el mayor valor de Q. Dado que Q representa la utilidad de continuación, al escoger siempre la acción asociada al mayor Q, se le llama a esta fase como .ªvara". Estas fases son definidas de forma exógena por la siguiente regla de decaimiento exponencial:

$$\epsilon_t = e^{-\beta t}$$

Donde $\beta = 0$ implica que se explora en todo instante, a mayor beta la probabilidad de explorar en ese periodo decae más rápido.

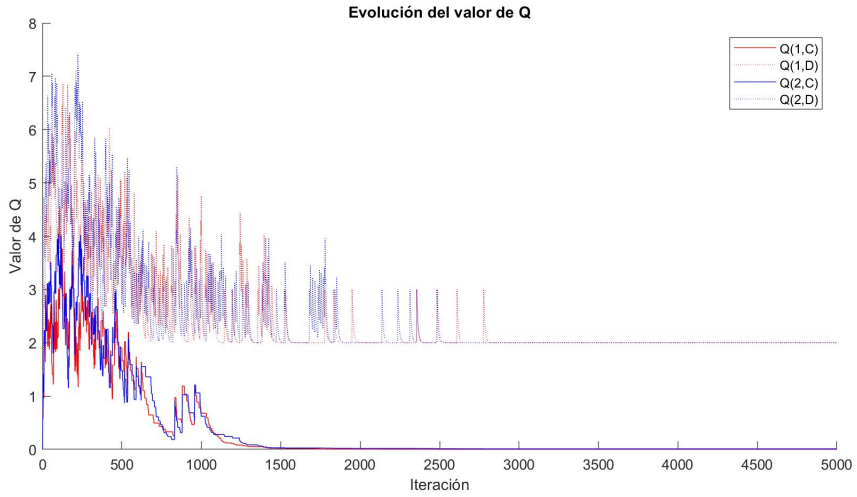


Figura 1.1: Evolución de la matriz Q dada la exploración exponencial.

Se adjunta un anexo dedicado a dar una breve revisión a los distintos regímenes de exploración.

Capítulo 2

Modelo

2.1. Conformación del modelo

Se puede identificar dos etapas principales, ambas con una componente teórica y una computacional que busca validar los resultados obtenidos.

La primera etapa consiste en evaluar el desempeño de algoritmos Q-learning en un ambiente de un juego repetido en un dilema del prisionero de una variable estado única y constante, cuyos parámetros son fijos. Simplificando el trabajo de Calvano y extendiéndolo con las restricciones en los sets de información propuestas, con esto se busca obtener una simplificación coherente con un modelo de mercado de más acciones disponibles para los jugadores.

En efecto, esta etapa consiste en una demanda markoviana de una cadena de un nodo. Este enfoque es útil porque nos permite sustraer todos aquellos estados de demanda en los cuales los jugadores no pueden incidir en sus transiciones de continuación. Esta sustracción es útil pues las estrategias obtenidas en esos estados es independiente de la obtenida en el resto de estados. De esta forma se puede obtener un modelo simple con solo 2 estados, pero que se puede extender a casos generales.

La segunda etapa es una extensión de lo anterior, incorporando un efecto rezagado de las acciones de los jugadores en la demanda y cómo los algoritmos Q-learning pueden captar este efecto. Un ejemplo de este efecto rezagado podría ser la cantidad de búsquedas online para un producto, cada vendedor puede observar la cantidad de visitas que recibe su producto, si un competidor fija un precio muy bajo causará que en el próximo periodo ambos vendedores observen una baja cantidad de visitas. Se confecciona un modelo con 2 estados, pues es el mínimo numero de estados interdependientes posible.

Esta extensión se puede interpretar esto como una multiplicidad de cadenas de markov, en la cual cada una de estas explica las fluctuaciones de la demanda según el set de precios escogidos por los jugadores. Un algoritmo Q-learning imputa un estado posterior a cada jugada una vez hecha, esta imputación implica que debe existir un conocimiento especifico de la evolución de la demanda y de su dependencia con los precios. Si bien esta dependencia

puede ser aprendida por los algoritmos[1], para la presente tesis se asume esta dependencia como conocida. Esto es equivalente a que el algoritmo espere a ver el estado del próximo periodo antes de imputar el nuevo ingreso.

Las partes computacionales de ambas etapas consisten en simular los ambientes mencionados en el software Matlab. El tipo de algoritmo escogido es el de aprendizaje por reforzamiento Q-learning, se hacen las pruebas para cada modelo y para cada combinación de parámetros y variables. En base a los resultados obtenidos se presentan implicancias en términos de institucionalidad regulatoria.

Se estudia la convergencia de algoritmos Q-Learning en el dilema del prisionero normalizado. Los jugadores poseen 2 jugadas, jugar precio alto p_H o jugar precio bajo p_L, los pagos en un periodo vienen dados por la siguiente matriz de pagos:

	p_H	p_L
p_H	(1,1)	(-d,t)
p_L	(t,-d)	(0,0)

Tabla 2.1: Dilema del prisionero.

En donde $t \in \{1, ..., 2\}$ es la la ganancia por el desvío y $d \in \{0, ..., 1\}$ es la perdida por la traición del rival. Se busca un set de pagos que permita representar un modelo de mercado, para ello se requiere un modelo simple y a su vez estudiado por otros autores, en particular la reducción se puede hacer en términos de pagos y luego normalizando. Un ejemplo sería la competencia en cantidades, que normalizada resulta en los pagos del dilema del prisionero de Axelrod (1980).

	$q_M = 1/4$	$q_{EN} = 1/3$
$q_M = 1/4$	(1/8,1/8)	(5/48,5/36)
$q_{EN} = 1/3$	(5/36,5/48)	(1/9,1/9)

Tabla 2.2: Pagos Cournot[a].

[a] Otra reducción podría ser utilizar el desvío óptimo del juego de Cournot $q_D = 3/8$, el juego resultante entrega resultados similares.

Normalizado resulta en:

	$q_M = 1/4$	$q_{EN} = 1/3$
$p_M = 1/4$	(1,1)	(-1/2,2)
$p_{EN} = 1/3$	(2,-1/2)	(0,0)

Tabla 2.3: Pagos Cournot normalizados.

Como primera aproximación se estudian a pares de decisiones, de esta forma se puede interpretar de forma simple la dependencia de los resultados colusorios de la forma del mercado. Podemos ver la tasa de convergencias resultantes o bien la fracción de equilibrios obtenidos, un ejemplo de esto se puede ver con memoria de un periodo $k = 1$ que se muestra a continuación como función de la ganancia por desvío t y la pérdida por traición d.

[1] Por ejemplo por medio de la regla de Bayes

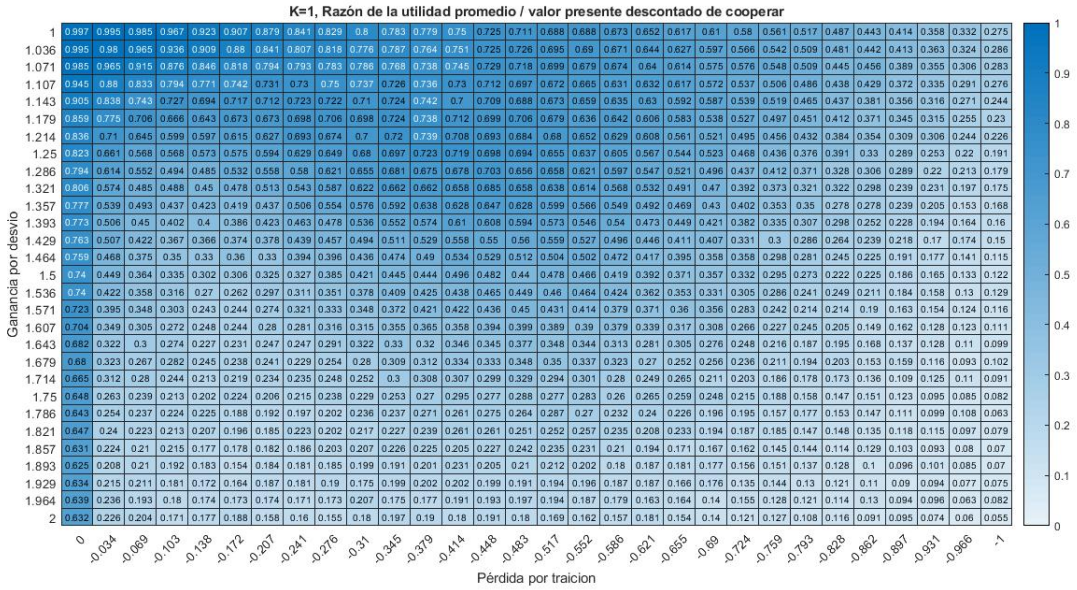


Figura 2.1: Razón de utilidad percibida con respecto a la monopólica $k = 1$

El incremento de la ganancia por desvío reduce fuertemente las ganancias de los agentes y un comportamiento no monótono aunque similar. Se adjunta en el anexo el análisis exploratorio con respecto a los distintos perfiles del dilema del prisionero. Se observa que las variaciones de los parámetros d y t tienen un fuerte impacto en los resultados y por tanto en las potenciales conclusiones. De esto se vuelve fundamental una elección apropiada del perfil de pagos para continuar. Se escogen 2 perfiles:

- Los pagos inicialmente propuestos por Axelrod.

	p_H	p_L
p_H	(3,3)	(0,5)
p_L	(5,0)	(1,1)

Tabla 2.4: Pagos Dilema de Axelrod.

normalizado resulta en:

	p_H	p_L
p_H	(1,1)	(-1/2,2)
p_L	(2,-1/2)	(0,0)

Tabla 2.5: Pagos Axelrod normalizados.

Este juego ha sido estudiado por Moriyama (2009), Carmel (1996) y Hington (2004), mientras tambien ha sido mencionado por otros autores como , Harrington (2017). Además, los pagos de este juego equivalen a la discretización del modelo de Cournot de 2 decisiones, producir la cantidad monopólica y la cantidad competitiva. Se adjunta en el anexo la tasa de éxito de los agentes para distintos valores de calibración e historias.

- Pagos propuestos por el autor

	p_H	p_L
p_H	(1,1)	(-1/4,7/4)
p_L	(7/4,-1/4)	(0,0)

Tabla 2.6: Pagos propuestos.

Que corresponde además a una simplificación del modelo propuesto por Calvano Et Al (2019) Se propone este set de pagos debido a que reúne 2 condiciones favorables, admite equilibrios que los pagos de Axelrod no admite y además presentó una baja tasa de convergencias a precio monopólico durante el análisis exploratorio realizado.

Se propone un modelo del dilema del prisionero iterado por diversos motivos entre los cuales esta la existencia de abundante literatura computacional y teórica acerca de este problema, la factibilidad computacional ya que requiere significativamente menos capacidad computacional que un modelo de mercado, las estrategias encontradas por otros autores son compatibles con sus correlativas del dilema del prisionero y es suficientemente simple y determinante como para no dejar pie a arbitrariedad en el comportamiento de mercado electo.

2.2. Supuestos

Se propone un modelo que busque simular un efecto rezagado de los precios en estados de demanda futuros para un mercado. En principio es necesario mencionar los supuestos que nos permiten utilizar un modelo de demanda markoviana cuyas transiciones son dependientes y condicionadas según las acciones presentes de los oferentes en un mercado real.

El primer supuesto es que las variables de estado se pueden identificar claramente, por ejemplo la demanda puede ser periodica; mensual, anual e incluso semanal, otro ejemplo puede ser el numero de visitas en un periodo, el cual es fácilmente obtenible en un portal digital. Otras variables se pueden considerar por ejemplo la volatilidad en los precios, la demanda por bienes de la competencia, el clima e incluso información propia de cada comprador como la sexo, domicilio y rutina de viajes, tales variables pueden ser utilizadas para responder con precios que, lejos de ser competitivos, son convenientes. Por ultimo los precios ofrecidos por los competidores pueden ser obtenidos rápidamente por medio de Apis que los principales portales de venta como Amazon entregan. En caso de no entregarlos existen técnicas de minería de datos como el web scraping que permiten el monitoreo perfecto de la competencia en tiempo real.

Se busca un enfoque que permita identificar, dada una acción, la diferencia en la razón de convergencia a precios colusivos de los algoritmos. Es por esto que este principio se puede aplicar posteriormente a una multiplicidad de mercados cuya demanda tenga dependencia intertemporal. Sin embargo esto requiere un segundo supuesto, la existencia de un efecto entre decisiones presentes de los oferentes y las demandas futuras u otros efectos con rezago.

En los mercados financieros es claro este efecto, cambios de precios afectan rápidamente la demanda y la volatilidad, sin embargo en mercados de bienes este efecto no es tan claro, los mercados de bienes traen consigo una serie de fricciones que hacen estos efectos se puedan confundir con otros factores como la periodicidad o la competencia con bienes parcialmente sustitutos.

El tercer supuesto es que existe una regla de transición identificable, Q-learning en cada periodo a una celda un valor de continuación asociado al próximo estado, esto sugiere, o bien conocer el próximo estado o conocer la distribución de probabilidad asociado a la transición. No se puede negar que esta es una dificultad para la implementación en un mercado real, sin embargo si la demanda es modelable por medio de una cadena de markov entonces solo bastará un periodo de entrenamiento prolongado para obtener las transiciones por medio de la regla de Bayes.

2.3. Ambiente

Se considera una demanda markoviana de 2 estados, un estado de alta demanda cuyos pagos equivalen al dilema del prisionero normalizado y un estado de baja demanda cuyos pagos son los del dilema del prisionero normalizado multiplicado por un factor $x \in (0, 1)$, este factor representa la magnitud del cambio en la demanda, de esta forma tendremos 2 dilemas del prisionero. Los pagos del estado de baja demanda L corresponden a:

	p_H	p_L
p_H	(x,x)	(-dx,tx)
p_L	(tx,-dx)	(0,0)

Tabla 2.7: Pagos en estado de baja demanda

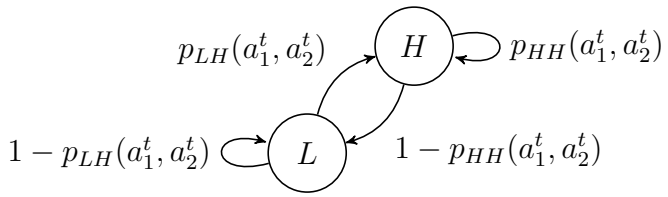
y los de alta demanda H:

	p_H	p_L
p_H	(1,1)	(-d,t)
p_L	(t,-d)	(0,0)

Tabla 2.8: Pagos en estado de alta demanda

El uso del factor x conserva la proporción entre los pagos de cada jugada, el objetivo de mantener esta proporción constante es aislar los efectos causados por perfiles de pago específicos. En otras palabras esto se implementa así para que los resultados no se puedan atribuir a distintas proporciones entre los pagos de cada jugada.

Las probabilidades de transición entre cada estados dependen de la jugada elegida por los jugadores en el periodo, las probabilidades de transición se muestran en la siguiente figura:



De esta forma, $p_{LH}(a_1^t, a_2^t)$ denota la probabilidad de que el próximo estado sea H si el estado anterior es L y se jugó a_1^t y a_2^t, si esta probabilidad es 1 las decisiones presentes afectan la demanda del periodo subsecuente de forma determinista. La disminución de $p_{LH}(a_1^t, a_2^t)$ proporciona incertidumbre en el intento de afectar la demanda, esta incertidumbre se incorpora para dificultar la capacidad de colusión, buscando así emular la incertidumbre de las acciones en un mercado real. Esta incertidumbre es parametrizada por una probabilidad p que define las transiciones.

$$p_{LH}(a_1^t, a_2^t) = p_{HH}(a_1^t, a_2^t) = \begin{cases} 1 & \text{si } \ a_1^t = p_H \ \text{ y } \ a_2^t = p_H \\ p & \text{si no} \end{cases}$$

De esta forma la elección de un precio alto llevará de forma certera al estado de alta demanda en el periodo siguiente, pero aun así existe la posibilidad de llegar a este estado si no se juega precios altos.

Cabe mencionar que no todos los estados de transición son alcanzables en este modelo, por ejemplo, no es posible obtener demanda baja una vez ambos jugadores hayan decidido manipularla.

2.4. Métricas

La primera medida es la tasa media de utilidad percibida por los agentes. La cual se obtiene del promedio por medio del calculo del ingreso descontado percibido y se normaliza.

$$u = \frac{u^p - u^n}{u^m - u^n}$$

Con u^p la utilidad descontada percibida, u^p la monopólica y u^p la correspondiente al equilibrio de Nash. Esta normalización se lleva a cabo debido a que el valor del ingreso monopólico puede variar de escenario a escenario, variación que lo haría no comparable. Por su construcción esta medida vale 0 en el equilibrio de nash y 1 en el máximo valor descontado obtenido. Por esto 1 hacer referencia a precios colusorios y 0 a precios competitivos.

Se considera que un par de algoritmos ha convergido si el set de valores máximos de su matriz de decisión Q sigue siendo el mismo durante 1000 iteraciones. Para evitar el ruido en el resultado se cancela la exploración 10 iteraciones antes de comenzar a verificar su convergencia. Otro enfoque sería extender las iteraciones hasta que todos las parejas de algoritmos converjan, sin embargo no hay necesidad de esto, la convergencia se alcanza con frecuencia antes de las 40000 iteraciones. Se reporta la tasa de pares de algoritmos que convergen a alguna estrategia.

Se reporta además la tasa de equilibrios perfectos en subjuego obtenidos, esta condición es equivalente a la de equilibrio markoviano perfecto y se obtiene resolviendo sistema de ecuaciones en busca de los valores de continuación condicionados a las estrategias obtenidas. Con estos valores de continuación se evalúa si el desvío unilateral es una decisión racional, si no lo es entonces se considera que es un equilibrio. Además, se reporta la diferencia sin normalizar en la estimación de los valores de la matriz Q, llamada Q-loss por Calvano (2019).

2.5. Utilidad de este enfoque

Dado al menos 2 estados y un par de acciones que condicione la transición entre ellos este modelo permite la comparación entre la convergencia entre los juegos de demanda con cadena con y sin dependencia. Por consiguiente entrega un factor no contemplado por la literatura precedente y es que; el efecto en el cambio del largo de la memoria puede tener, bajo los supuestos mencionados anteriormente, efectos contrarios a lo que se buscaba en un comienzo.

La parametrización de la probabilidad de transición da origen a un único par de jugadas que afectan al estado futuro de la demanda, el hecho que este par de jugadas sea única permite usar la tasa de preferencia a ellas como métrica de éxito, junto con despejar efectos adversos de otros modelos más complejos como la dificultad de atribuir causalidad a los resultados. Por estos efectos es útil tener un modelo tan simple como sea posible.

En este ambiente además surgen estrategias que admiten precios colusivos sin necesidad de esquemas de castigo propuestos por Harrington (2017), esto dejaría obsoleta su propuesta para estos ambientes. Es indispensable incorporar las condiciones de dependencia intertemporal para evitar resultados no previstos en la medida de restricción de memoria a los algoritmos.

2.6. Estrategias posibles

2.6.1. Gatillo sin monitoreo

Considerando ambientes de dependencia rezagada entre los estados de la demanda presente y precios pasados se propone una estrategia que, sin disponer de precios históricos, es capaz de mantener llegar a resultados colusivos y al mismo tiempo ser un equilibrio markoviano perfecto. Los precios colusivos se obtienen por medio de una manipulación tácitamente coordinada de la demanda.

Consideramos la estrategia $\sigma = \sigma_{S*}(s)$ descrita por:

$$\sigma(s) := \begin{cases} p_h, & \text{si } s \in S* \\ p_{nash}, & \text{si no} \end{cases}$$

En un oligopolio que enfrenta demanda markoviana tal que su tasa de transición es función de sus precios pasados $q(s^t, a^{t-1})$ la estrategia σ define un equilibrio markoviano perfecto si en cada s:

$$V(\sigma, \sigma, s) \geq u(\theta, \sigma, s) + \delta Q_s(\sigma, \sigma) V(\sigma, \sigma)$$

Considerando θ al mejor desvío del estado contingente, δ el factor de descuento y $V(\sigma, \sigma, s)$ el valor de continuación, el cual se puede obtener de resolver el sistema de ecuaciones condicionando en la estrategia.

$$V(\sigma, \sigma) = [I - \delta Q(\sigma, \sigma)]^{-1} U(\sigma, \sigma)$$

Con $V(\sigma, \sigma)$ el vector de valores de continuación, $U(\sigma, \sigma)$ el vector de utilidades y $Q(\sigma, \sigma)$ la matriz de transición.

Cabe mencionar que la viabilidad de este equilibrio requiere que el estado de la demanda sea dependiente es de los precios pasados, en este sentido esta estrategia no implica un esquema de castigo, como el rival desconoce la historia de jugadas no tiene forma de castigar. Esta estrategia es sostenible por las probabilidades que condicionan los pagos de futuros, esto quiere decir que es el entorno quien hace el castigo en caso de desvío del acuerdo colusivo. Desde luego, siempre que el jugador rival se mantenga en la estrategia.

2.6.2. Subida de precios

Consideramos la estrategia $\sigma_{sp} = \sigma_{spS*}(s)$ descrita por:

$$\sigma_{sp}(s) := \begin{cases} p_{nash}, & \text{si } s \in S* \\ p_h, & \text{si no} \end{cases}$$

Esta estrategia no induce un equilibrio markoviano debido a que el desvío es preferible en todos los estados fuera de $S*$. Converger a estas estrategias implicaría que los algoritmos buscan el mejor equilibrio de nash para jugar precios competitivos.

2.6.3. Monopolio ciego

Consideramos la estrategia $\sigma_{mc} = \sigma_{mcS*}(s)$ descrita por:

$$\sigma_{mc}(s) := \begin{cases} p_h, & \text{si } s \in S* \\ p_h, & \text{si no} \end{cases}$$

Esta estrategia tampoco induce un equilibrio.

2.6.4. Siempre Bertrand

Consideramos la estrategia $\sigma_{sb} = \sigma_{sbS*}(s)$ descrita por:

$$\sigma_{sb}(s) := p_{nash}$$

Esta estrategia es un símil a los precios competitivos encontrados en un mercado. Esta estrategia induce un equilibrio markoviano perfecto al jugar el equilibrio de Nash en cada periodo.

2.7. Implementación

En esta sección se presenta el objetivo particular de cada experimento junto con los parametros utilizados.

Cada experimento considera 870 escenarios, en cada escenario 1000 pares de algoritmos Q-learning compiten en un dilema del prisionero iterado 50000 veces, además que posee 2 estados de demanda. Con esto cada estado es en promedio visitado unas 1250 veces cuando no se posee memoria $k = 0$ y unas 350 veces si la memoria es de largo $k = 1$.

2.7.1. Ejemplo 1: Efecto de falsos negativos y diferencia en la demanda

Se consideran 30 valores de la probabilidad de falsos negativos[2] $p \in (0, 1)$ y 29 valores de diferencia de demanda $x \in (0, 1)$. Se utilizan los pagos de Axelrod normalizados y un factor de descuento $\delta = 0.9$. Todos los algoritmos poseen la misma calibración, esto es una tasa de aprendizaje $\alpha = 0.25$ y una tasa de decaimiento exponencial de su exploración $\beta = 2 * 10^{-4}$. Esta experiencia nos permite ver el efecto de los falsos negativos o ruido y el premio en la diferencia de estados.

En ningún punto de esta grilla la estrategia propuesta es equilibrio, por tanto, en conjunto con el ejemplo 2 permiten corroborar si el concepto de equilibrio afecta las tasas de convergencia de estos algoritmos a esos equilibrios.

2.7.2. Ejemplo 2: σ como EMP y factor de descuento

Nuevamente se consideran 30 valores de la probabilidad de falsos negativos $p \in (0, 1)$ pero esta vez con 29 factores de descuento $\delta \in (0, 1)$. Se utilizan los pagos de propuestos por el autor ya que con ellos la estrategia propuesta σ ser equilibrio markoviano perfecto permitiendo así la comparación el ejemplo 1. Se considera una diferencia entre estados de demanda de $x = 0.5$. Nuevamente todos los algoritmos poseen la misma calibración, esto es una tasa de aprendizaje $\alpha = 0.25$ y una tasa de decaimiento exponencial de su exploración $\beta = 2 * 10^{-4}$.

[2] Se les llama falsos negativos debido a que se obtienen valores de continuación colusiorios a pesar de no hacer lo necesario para obtenerlos

Capítulo 3

Resultados

En esta sección se presentan los resultados obtenidos de las experiencias

Ejemplo 1

Los resultados obtenidos de la experiencia 1 son cuando menos desalentadores. La restricción de conocer precios pasados en este ambiente resulta propiciar la colaboración entre los agentes a pesar que esta era menos común en el juego repetido de un periodo.

La utilidad promedio obtenida en el escenario de memoria restringida, es decir $k = 0$, equivale a un 84.3 % de la utilidad monopólica y a un 45.2 % para algoritmos con memoria de un periodo $k = 1$.

Con respecto a las estrategias obtenidas, para $k = 0$ un 84.3 % corresponde a estrategias de precio siempre monopólico y a solo un 0.55 % de precio siempre competitivo. Las restantes no convergieron a estrategias dentro de los 50.000 periodos pese a restringirse la exploración. La mayoría de estas estrategias obtenidas no constituyen equilibrio, solo un 0.55 % lo es (coincidiendo con el mismo porcentaje de estrategias de equilibrio de nash). El restante de las estrategias corresponden a "subamos los precios".

Cabe mencionar que en este ejemplo la estrategia propuesta σ^1 no constituye un equilibrio, debido a que los pagos del desvío son muy altos. Para que esta estrategia logre ser equilibrio basta restar 0.1 o más a la ganancia del desvío. Sin embargo esta resta vuelve todavía más atractiva la estrategia avara de jugar siempre precio monopólico. A partir de esto se puede concluir que la noción de equilibrio no aplica como lo haría entre jugadores humanos, o al menos no lo hace en el contexto propuesto. Esto es discutido a fondo en el capitulo de conclusiones.

Podemos observar que el parametro de la diferencia en las ganancias x es irrelevante, esto significa que basta con tener un estado que permita distinguir un efecto rezagado para sostener precios colusivos incluso si entrega los mismo pagos. En otras palabras, basta que exista una consecuencia detectable para cada precio para poder sostener estrategias σ de gatillo sin monitoreo. Todo esto puede explicar el efecto regresivo de la medida.

[1] Estrategias de gatillo sin monitoreo

Ejemplo 2

La segunda experiencia entrega resultados concordantes con la primera, sigue siendo irrelevante la diferencia de pagos entre los estados, la probabilidad de falso negativo en la colaboración tiene el impacto negativo y el factor de descuento favorece la convergencia a precios colaborativos como es de esperar.

La figura siguiente muestra en azul la región en la cual una restricción de información puede tener resultados positivos, en su mayoría reporta un alza en los precios. Los datos mostrados en la figura corresponden a la resta entre las utilidades obtenidas entre ambos modelos, destacando en azul aquellos que representan una ganancia.

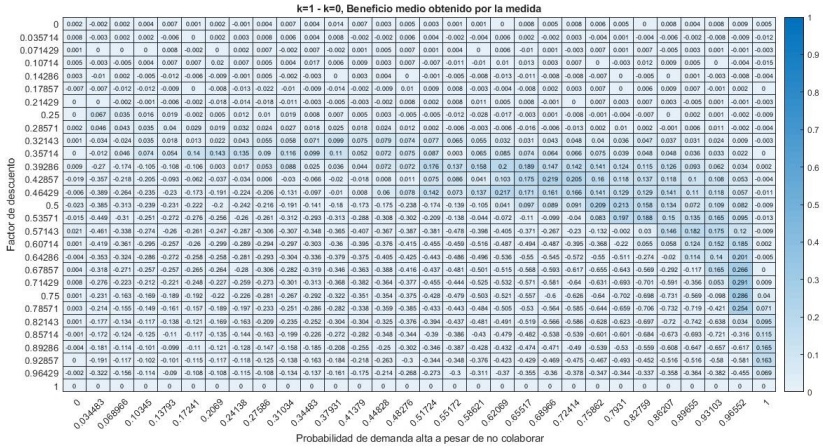


Figura 3.1: k=1 - k=0, Beneficio medio obtenido por la medida

Pese a que para esta configuración la estrategia σ si es un EMP, no se observa convergencia de pares a ella. Nuevamente los algoritmos convergen a estrategia de precio monopólico constante. El motivo por el cual se observan precios altos se puede atribuir también a la baja cantidad de estrategias alternativas disponibles, dada la escasez de estados, por tanto facilita el registro de precios altos y por tanto la convergencia a ellos.

Si se obliga a uno de los algortimos a desviarse unilateralmente se observa que estos desvíos concluyen sin un castigo del competidor rival, y que aun así no son sostenidos por la firma desertora, la cual vuelve en el próximo periodo a la colaboración. Repetir esto consecutivamente termina en que la firma rival comienza a jugar precio bajo en todos los estados, sin ninguna vuelta futura a la colaboración. Este comportamiento solo surge como consecuencia de la intervención reiterada y no forma parte de las estrategias iniciales a las cuales se converge, es decir, la intervención fue lo suficientemente prolongada como para que los algoritmos se adapten cambiando su estrategia.

Capítulo 4

Conclusiones

La reducción en el largo de la memoria altera claramente el tipo de estrategias a las cuales se converge en los oligopolios Q-learning. Lejos de obtener esquemas de recompensa y castigo más bien parecen igualar el precio del rival mientras ofrecen señales para subirlos. Estas señales se obtienen de forma tácita e independiente del conocimiento de precios pasados del rival.

Las estrategias resultantes son principalmente 2, "monopolio ciego" y "subida de precios", con escasas convergencias al equilibrio "siempre Bertrand" y aún menos frecuentes "gatillo sin monitoreo".

Las estrategias "subida de precio" se pueden entender como una búsqueda del mejor equilibrio de Nash, estas estrategias serían análogas a las encontradas por Calvano (2019) en el sentido que los agentes tenderán a subir su precio si es muy bajo y a bajarlo si es muy alto. En otras palabras la convergencia a estas estrategias muestra cómo la secuencia de estados siguientes afectan la decisión presente, haciendo atractivas jugadas por sus consecuencias futuras.

Las estrategias obtenidas por los algoritmos no conforman necesariamente equilibrios, esto no es intuitivo en sí, la existencia de desvíos rentables es muestra que estos algoritmos no poseen regímenes de decisión racionales. La falta de equilibrios se atribuye al mecanismo de decisión del Q-learning, cuyo régimen de exploración de decaimiento exponencial hace que los algoritmos dejen de explorar a partir de un cierto instante, permitiendo así convergencia específica entre cada par de jugadores.

Los resultados encontrados, considerando los ejemplos 1 y 2 en comparación a los dilemas del prisionero en donde las jugadas no tienen impacto en precios futuros, sugieren que disminuir el largo de la memoria incrementa las chances de que los algoritmos converjan a precios colusivos, junto con mantenerlos irracionalmente. Es por este incremento de chances que se observan estrategias "monopolio ciego" más frecuentemente. Además cabe mencionar que la falta de monitoreo directo también impide el castigo, siendo necesaria una gran cantidad de desvíos para que el algoritmo rival responda, esto facilita aun más la colusión.

Todas estas estrategias aún no son atendidas por la regulación de competencia actual y

tampoco por los estudios mencionados una vez el concepto de castigo y recompensa no es aplicable a algoritmos sin memoria.

La principal implicancia de estos resultados es que la capacidad de monitoreo de los algoritmos Q-learning esta subestimada. En tanto exista un set de información publica compartida que refleje en alguna medida las acciones de los jugadores pueden darse precios supracompetitivos. Esto significa que la inclusión de estos algoritmos trae impactos negativos sobre la competencia cuando el poder de mercado de estos es alto.

Una restricción en la información disponible para los algoritmos puede dificultar la obtención de precios monopólicos en algoritmos Q-learning. Sin embargo es necesario un estudio del comportamiento de los precios previo a su implementación. Factores clave para el éxito de esta medida son:

- Alta elasticidad precio de la demanda. Esto permite a los algoritmos obtener ganancias durante sus periodos de desvío y por tanto preferir precios competitivos. Mercados de baja elasticidad pueden llevar a esta medida a ser completamente regresiva, entendiendo regresiva en el sentido que incrementaría los precios de mercado comparado con los precios previos a la intervención.
- La correlación entre precios pasados y todos los componentes de la demanda presente debe ser baja, en otras palabras, los precios presentes no deben ser causa directa de futuros cambios en variables de mercado. Una alta correlación con al menos uno de esos componentes significa que puede ser utilizado como mecanismo de coordinación, incluso sin intención de ello. Esta falta de intencionalidad es la principal dificultad en términos de condenar el uso de estos algoritmos, como argumenta Harrington (2017), muchas veces los directivos de las firmas no tienen conocimiento alguno de qué tratan o cómo funcionan estos algoritmos.

Impedir o bien disminuir el riesgo de obtener colusiones tácitas en los algoritmos es difícil debido a la falta de transparencia de estos algoritmos, no solo en términos de ser una caja negra, sino que los mecanismos por medios de los cuales estos obtienen sus estrategias son particularmente difíciles de predecir y entender. Es por esto que la transparencia es clave para el éxito de un mecanismo regulador. Lamentablemente la transparencia total puede facilitar la colusión, el conocer la estrategia del rival puede permitir que los oferente escojan algoritmos a la medida de su competencia por tanto, se requiere que al menos se den a conocer los parámetros sobre los cuales condicionan sus estrategias.

Se vuelve indispensable para un regulador impedir la comunicación tácita, no solo en el sentido de evitar observar las acciones del rival sino que también evitar ver sus impactos directos. El enfoque más simple sería restringir el aprendizaje de todo algoritmo fijador de precios, de esta forma se facilita su auditoría. Estas medida en conjunto con la restricción de memoria propuesta favorecerían un mercado competitivo.

Queda como idea para próximos estudios ampliar el espacio de acciones, asimetrías de k y estudiar la entrada y acomodamiento de algoritmos Q-learning, toda vez las estrategias obtenidas por ellos son fácilmente obtenidas por ingeniería inversa. Como nota al margen, un régimen de exploración constante y de baja probabilidad parece ser más razonable para

la operación de algoritmos en el día a día y por tanto para nuevos estudios, pues asegura que los ratios obtenidos no sean de un perfil puntual dependiente de la tasa de exploración, sino que son la consecuencia de poblaciones evolutivamente estables, las cuales se asemejan a un mercado real. Además, es difícil que se pueda obtener la información suficiente para entrenar estos algoritmos de forma previa a su operación.

Indudablemente la implementación de estos algoritmos en la industria plantea la necesidad de nuevos marcos regulatorios. Salvaguardar la libre competencia forma parte esencial de las responsabilidades de una sociedad libre.

Bibliografía

[1] Axelrod, R. (1980). Effective choice in the prisoner's dilemma. *Journal of conflict resolution*, *24*(1), 3–25. doi: https://doi.org/10.1177/002200278002400101

[2] Axelrod, R. (1981). The emergence of cooperation among egoists. *American Political Science Association*, *75*(2), 306–318. doi: https://doi.org/10.2307/1961366

[3] Boehmer, E. (2018). The competitive landscape of high-frequency trading firms. *The Review of Financial Studies*, *31*(6), 2227–2276. doi: https://doi.org/10.1093/rfs/hhx144

[4] Calvano, E., Calzolari, G., Denicolo, V., y Pastorello, S. (2019). Artificial intelligence, algorithmic pricing and collusion.
doi: http://dx.doi.org/10.2139/ssrn.3304991

[5] Carmel, D., y Markovitch, S. (1996). Learning models of intelligent agents. *Proc. of AAAI-96*. doi: https://www.researchgate.net/publication/2774090_Learning_Models_of_Intelligent_Agents

[6] Chen, L., Mislove, A., y Wilson, C. (2015). Peeking beneath the hood of uber. *IMC '15: Proceedings of the 2015 Internet Measurement Conference*, 495-508. doi: https://doi.org/10.1145/2815675.2815681

[7] Chen, L., Mislove, A., y Wilson, C. (2016). An empirical analysis of algorithmic pricing on amazon marketplace. , 1339-1349. doi: https://doi.org/10.1145/2872427.2883089

[8] Ezrachi, A., y Stucke, M. E. (2017). Artificial intelligence collusion: When computers inhibit competition. *University of Illinois Law Review*. doi: https://ssrn.com/abstract=2591874

[9] Harrington, J. (2017). Developing competition law for collusion by autonomous price-setting agents.
doi: http://dx.doi.org/10.2139/ssrn.3037818

[10] Hingston, P., y Kendall, G. (2004). Learning versus evolution in iterated prisoner's dilemma. *Institute of Electrical and Electronics Engineers*, *1*, 364 - 372. doi: https://doi.org/10.1109/CEC.2004.1330880

[11] Ittoo, A., y Petit, N. (2017). Algorithmic pricing agents and tacit collusion: A technological perspective. *Chapter in L'intelligence artificielle et le droit, Hervé JACQUEMIN and Alexandre DE STREEL*, 241-256. doi: http://dx.doi.org/10.2139/ssrn.3046405

[12] Moriyama, K. (2009). Utility based q-learning to facilitate cooperation in prisoner's dilemma games. *Web Intell. Agent Syst.*, *7*, 233-242. doi: https://doi.org/10.3233/WIA-2009-0165

[13] OECD. (2017). Algorithms and collusion: Competition policy in the digital age.

doi: www.oecd.org/competition/algorithms-collusion-competition-policy-in-the-digital-age.htm

[14] Salcedo, B. (2015). Pricing algorithms and tacit collusion.

[15] Watkins, C., y Dayan, P. (1992). Technical note: Q-learning. *Machine Learning*, *8*, 279-292. doi: https://doi.org/10.1007/BF00992698

Anexo A

Verificación de equilibrio

Por simplicidad se considera que solo existe un único estado de demanda alta y un único estado de demanda baja, denotamos $s*$ al estado de la demanda en el cual existe un perfil de precios más rentable que el equilibrio y s al que no. Además denotamos $\sigma = \sigma_{S*}(s)$ a la estrategia propuesta y $\theta = \theta(s)$ a la estrategia de desvío. Para determinar si (σ, σ) corresponde a un equilibrio primero se obtienen los valores de continuación dado que ambos jugadores siguen la estrategia propuesta resolviendo:

$$V(\sigma, \sigma, s*) = u(\sigma, \sigma, s*) + \delta(\Pr(s* \mid \sigma, \sigma, s*)V(\sigma, \sigma, s*) + \Pr(s \mid \sigma, \sigma, s*)V(\sigma, \sigma, s))$$

$$V(\sigma, \sigma, s) = u(\sigma, \sigma, s) + \delta(\Pr(s* \mid \sigma, \sigma, s)V(\sigma, \sigma, s*) + \Pr(s \mid \sigma, \sigma, s)V(\sigma, \sigma, s))$$

obteniendo

$$V(\sigma, \sigma, s*) = \left(\frac{u(\sigma, \sigma, s*) - \delta u(\sigma, \sigma, s*)\Pr(s \mid \sigma, \sigma, s) + \delta u(\sigma, \sigma, s)\Pr(s \mid \sigma, \sigma, s*)}{(1 - \delta\Pr(s* \mid \sigma, \sigma, s*))(1 - \delta\Pr(s \mid \sigma, \sigma, s)) - \delta^2\Pr(s \mid \sigma, \sigma, s*)\Pr(s* \mid \sigma, \sigma, s)}\right)$$

$$V(\sigma, \sigma, s*) = \left(\frac{u(\sigma, \sigma, s) - \delta u(\sigma, \sigma, s)\Pr(s* \mid \sigma, \sigma, s*) + \delta u(\sigma, \sigma, s*)\Pr(s* \mid \sigma, \sigma, s)}{(1 - \delta\Pr(s \mid \sigma, \sigma, s))(1 - \delta\Pr(s* \mid \sigma, \sigma, s*)) - \delta^2\Pr(s \mid \sigma, \sigma, s*)\Pr(s* \mid \sigma, \sigma, s)}\right)$$

Finalmente se verifica la condición de desvío único:

$$V(\sigma, \sigma, s*) > u(\theta, \sigma, s*) + \delta(\Pr(s* \mid \theta, \sigma, s*)V(\sigma, \sigma, s*) + \Pr(s \mid \theta, \sigma, s*)V(\sigma, \sigma, s))$$

Esta condición no se sostiene para el dilema del prisionero propuesto (dilema de Axelrod). Ninguna combinación de δ, probabilidad de ocurrencia p y diferencia de pagos x puede sostenerla.

Esto de debe a la naturaleza de la cadena, el pago del desvió propuesto por axelrod es muy alto para que sea atractivo esperar las utilidades futuras.

En la siguiente figura se muestra las regiones acumulativas como función del pago del desvío que permiten que la condición de equilibrio se cumpla.

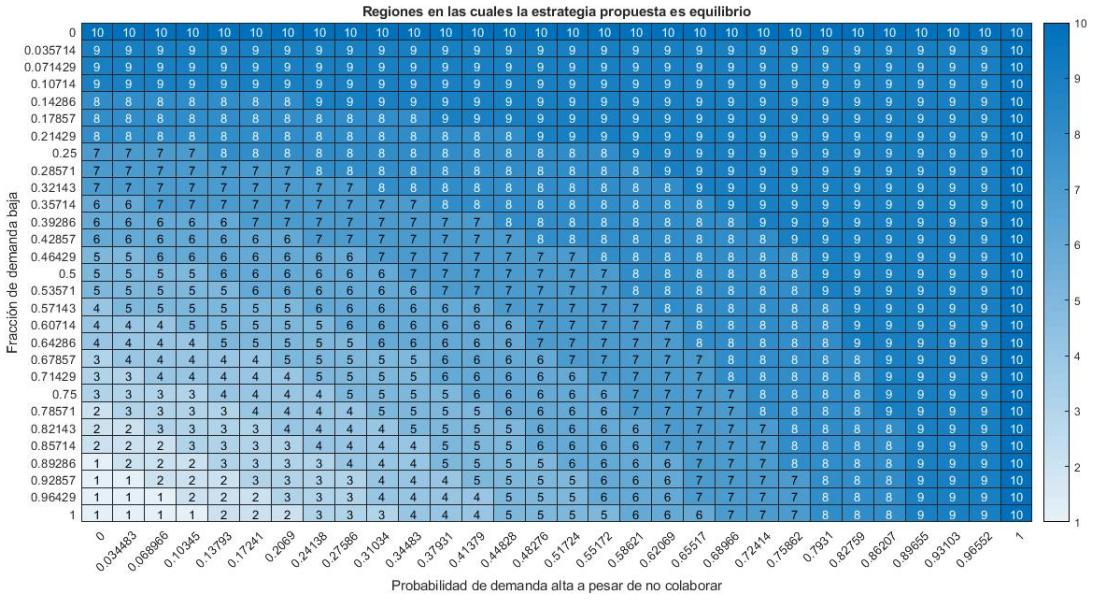


Figura A.1: Regiones acumulativas en las cuales la estrategia propuesta es equilibrio.

La figura muestra región acumulativas para la condición de equilibrio de la estrategia propuesta. Esto es 10 es la región en donde todos los juegos testeados son equilibrio. Por otra parte 1 corresponde a la región en donde el desvío no paga adicional, por tanto es equilibrio en todo el plano.

Ganancia por desvío	Áreas de equilibrio
1.0	Todas
1.1	2 y mayores
1.2	3 y mayores
1.3	4 y mayores
1.4	5 y mayores
1.5	6 y mayores
1.6	7 y mayores
1.7	8, 9 y 10
1.8	9 y 10
1.9	10
Axelrod	Ninguna

Anexo B

Acerca de los regímenes de exploración

Considere un dilema del prisionero iterado con la siguiente matriz de pagos:

	p_H	p_L
p_H	(c,c)	(t,d)
p_L	(t,d)	(p,p)

Tabla B.1: Matriz de pago genérica

Considerando los valores $c = 3$, $t = 0$, $d = 5$ y $p = 1$. Se utiliza un factor de descuento de 0.5, una tasa de exploración constante 0.05 y una tasa de aprendizaje 0.25. Los algoritmos Q-learning juegan simultáneamente por 5000 periodos. Consideremos primero un régimen de exploración constante.

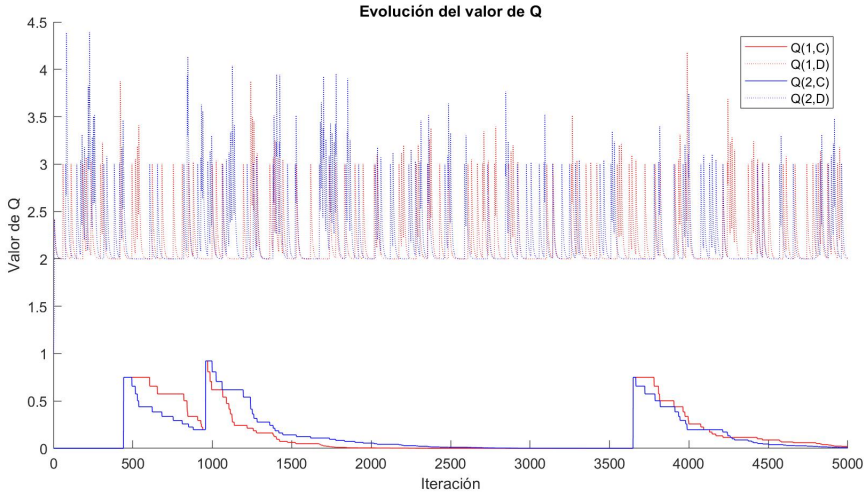


Figura B.1: Evolución del VPD en el tiempo con exploración constante.

En la figura se observan los valores de Q de cada jugador, en rojo el jugador 1 y en azul el jugador 2, la linea continua representa la celda asociada a jugar precio alto p_H. Podemos observar que el valor esperado de cooperar es estrictamente menor a el valor de jugar precio bajo. Las leves alzar del valor de colaborar corresponden a instancia en las que

ambos jugadores juegan p_H a la vez, cada baja de este valor corresponde al registro de utilidad 0 que se corresponde con un alza en la utilidad del rival en el periodo (los dientes de sierra del valor de jugar precio bajo p_L). Si tomamos el mismo modelo y modificamos el régimen de exploración por uno con decaimiento exponencial de tasa $\beta = 2 * 10^{-3}$ se obtiene:

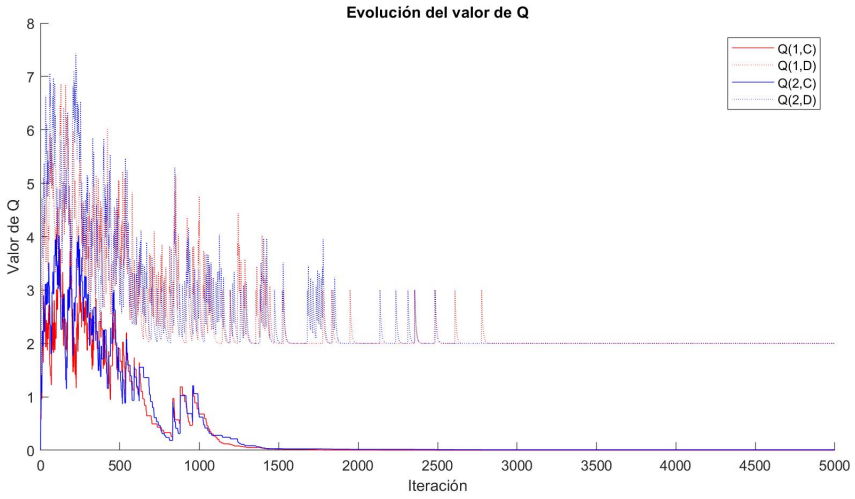


Figura B.2: Evolución del VPD en el tiempo con exploración con decaimiento exponencial.

Cabe mencionar que en la medida que incrementa beta, la exploración cae más abruptamente, de forma que a valores menores de beta se requiere incrementar fuertemente la cantidad de iteraciones.

Cada jugador, si no se encuentra en fase de exploración, escogerá la política avara, es decir escogerá la que en ese momento posea el mayor valor de Q. En la siguiente figura se aprecia el comportamiento de Q cuando ambos jugadores juegan precio alto durante 40 periodos consecutivos a partir de la iteración 1300.

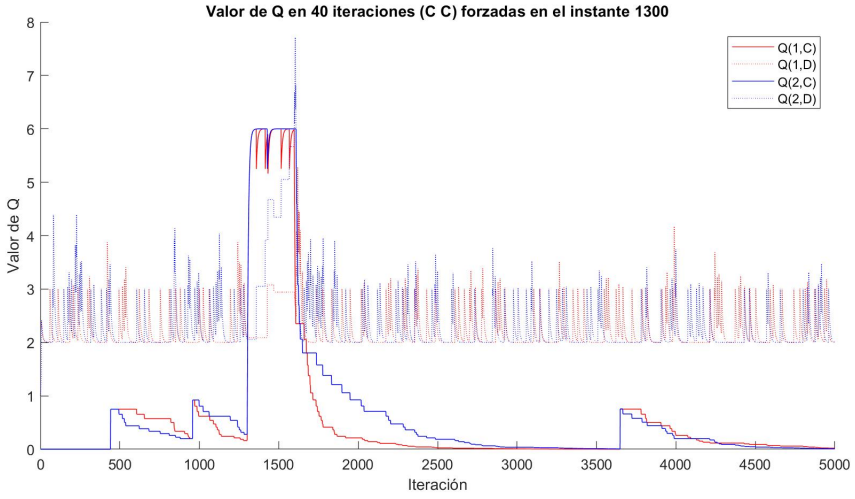


Figura B.3: Evolución del VPD en el tiempo con exploración constante con 40 iteraciones de colaboración consecutivas.

Podemos observar que los algoritmos continúan escogiendo precios altos hasta que paula-

tinamente la el valor de desviarse supera al valor de cooperar. Lo mismo se puede hacer en un régimen de exploración con decaimiento exponencial.

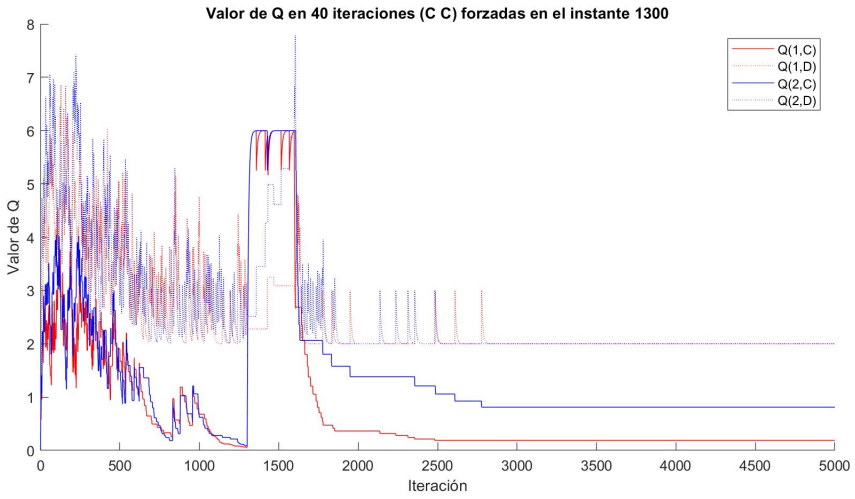


Figura B.4: Evolución del VPD en el tiempo con exploración con decaimiento exponencial con 40 iteraciones de colaboración consecutivas.

Es claro notar que al decaer la probabilidad de exploración el efecto de la intervención se vuelve cada vez más difícil de revertir.

En las simulaciones del dilema del prisionero se utiliza $\beta = 0.2 * 10^{-4}$ con ello se explora en promedio 50000 periodos.

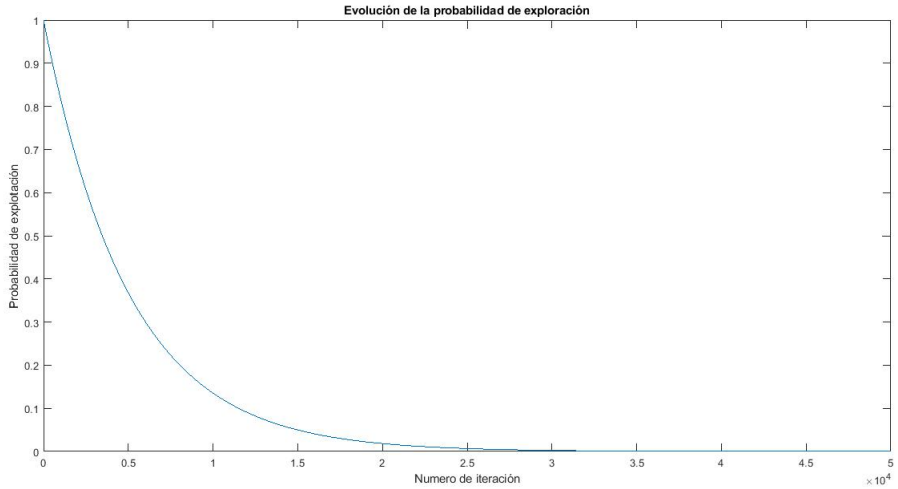


Figura B.5: Evolución de la probabilidad de exploración para $\beta = 0.2 * 10^{-4}$.

Con esto la cantidad de veces que se visita un periodo puede ser obtenida fácilmente dividiendo, con esto para $K = 0$ cada estado se visita por exploración en promedio 2500 veces, para $K = 1$ 625 veces, $K = 2$ 156 veces, $K = 3$ 39 veces, $K = 4$ 10 veces y finalmente $K = 5$ en torno a las 2 veces.

Anexo C

Algunos modelos de rezago

En este anexo se busca estudiar la naturaleza de los juegos repetidos en los que los estados de pagos son dependientes de los precios. Con el fin de evaluar si en teoría se puede sostener estrategias colusivas.

Estados completamente exógenos

Consideremos un mercado de bienes perfectamente sustitutos, cuya demanda es definida por una cadena de markov ergódica con una única clase recurrente no periódica.
Sean $V_i^j(E_i, E_{-i})$ el valor de continuación del jugador i de jugar una estrategia E_i en el estado j y $u_i^j(s_i, s_{-i})$ el pago obtenido por el jugador i de jugar s_i en el estado j dado que rival ha jugado s_{-i}.
Supongamos 2 agentes juegan un dilema del prisionero en el cual no pueden condicionar sobre la historia de precios ni sobre su historia de ventas. Al ser un dilema del prisionero se tiene que $u_i^j(L, s_{-i}) > u_i^j(H, s_{-i})$. Luego para toda estrategia E_i se tiene:

$$u_i^j(L, s_{-i}) + V_i^j(E_i, E_{-i}) > u_i^j(H, s_{-i}) + V_i^j(E_i, E_{-i})$$

En otras palabras, para toda estrategia y todo estado el desvío a L es un EMP (Equilibrio markoviano perfecto). Se puede ver que esto es independiente de la estrategia pues, al no poder condicionar en las historias pasadas, los valores de continuación son independientes de los pagos presentes, permitiendo así el desvió en un periodo para toda estrategia que escoja un precio distinto a p_L.

Por lo tanto es necesaria cierta dependencia a las acciones de los jugadores.

Caso de estados dependientes en precios pasados

En este modelo la variable del estado de la demanda s_t será dependiente de los precios históricos de ambos algoritmos, esto es:

$$s_{t+1}(p_t^1, p_t^2) = \begin{cases} x & si \;\; p_t^1 = p_t^2 = p_H \;\; o \;\; bien \;\; t = 0, 1 \\ z & si \;\; no \end{cases}$$

De esta forma el estado actual es markoviano solamente si al menos un jugador no juega

p_H. Se considera una estrategia gatillo de forma que en periodo t:

$$\sigma(s_{t-1}) := \begin{cases} p_H & si \quad s_{t-1} = x \\ p_L & si \quad sino \end{cases}$$

En otras palabras la demanda actúa como medio de comunicación entre las firmas. Las estrategias gatillo son viables si existe δ tal que cumpla:

Si $s_t = x$

$$\frac{x}{2} \cdot \frac{1}{1-\delta} \geq x + \frac{\delta}{1-\delta} \cdot 0$$
$$\delta \geq \frac{1}{2}$$

Si $s_{t=0} = z$

$$\frac{z}{2} + \frac{x}{2} \cdot \frac{\delta}{1-\delta} \geq z + \frac{\delta}{1-\delta} \cdot 0$$
$$\delta \geq \frac{z}{x+z}$$

Con ello:

$$\delta \geq \max\{\frac{1}{2}, \frac{z}{x+z}\}$$

Con ello la estrategia es un MPE y por tanto un EPS.

Con error tipo 1

Ahora consideraremos una variante, en caso que ambas firmas escojan precio p_H existirá una probabilidad $1-\epsilon$ de que el precio sea bajo.
Las firmas pueden observar el estado actual de la demanda s_t. Vemos según los posibles valores que puede tomar la variable:

$$\frac{x}{2} + \frac{\delta}{1-\epsilon\delta}[\frac{x}{2}\epsilon + \frac{z}{2}(1-\epsilon)] \geq x$$

$$\delta \geq \frac{x}{2x\epsilon - z(1-\epsilon)}$$

Para el próximo estado de la cadena:

$$\frac{z}{2} + \frac{\delta}{1-\epsilon\delta}[\frac{x}{2}\epsilon + \frac{z}{2}(1-\epsilon)] \geq z$$

$$\delta[\frac{x}{2}\epsilon + \frac{z}{2}(1-\epsilon) \geq \frac{z}{2}(1-\epsilon)$$

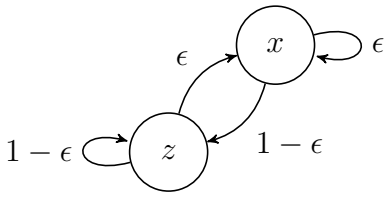
$$\delta \geq \frac{z}{z+x\epsilon}$$

Considerando ambas restricciones se tiene:

$$\delta \geq \max\{\frac{x}{2x\epsilon - z(1-\epsilon)}, \frac{z}{z + x\epsilon}\}$$

Aplicación

Se considera un modelo de juego repetido con 2 jugadores, la demanda de cada periodo es exógena y es descrita por la siguiente cadena de Markov de dos estados:



Y la siguiente matriz de transición:

$$q = \begin{bmatrix} 1-\epsilon & \epsilon \\ 1-\epsilon & \epsilon \end{bmatrix}$$

Cada periodo las firmas escogen algoritmos que fijan precios durante todos los periodos en un un dilema del prisionero. La matriz de pagos de cada periodo esta dada por:

	p_H	p_L
p_H	$(\frac{s_t}{2}, \frac{s_t}{2})$	(-ξ,s_t)
p_L	(s_t,-ξ)	(0,0)

Tabla C.1: Aplicación de 2 estados

Los algoritmos separan cada juego en cada nodo y los tratan por separado. Es decir la incidencia de los precios en la demanda no es captada por los algoritmos de forma directa.

Con error tipo 2

Se considera la probabilidad ϵ de que dado que ambos fijan precio p_L el estado sea alto al periodo siguiente:

$$\frac{x}{2}\frac{1}{1-\delta} \geq x + \delta^2\epsilon x\frac{1}{1-\delta}\frac{1}{1-\delta(1-\epsilon)}$$

$$1 - 2 + 2\delta \geq \delta^2\epsilon\frac{1}{1-\delta(1-\epsilon)}$$

$$-1 + (3-\epsilon)\delta - (2-\epsilon)\delta^2$$

Finalmente:

$$\delta \geq \frac{1}{2-\epsilon}$$

En caso que el estado inicial sea z se tiene:

$$\frac{z}{2}\frac{1}{1-\delta} \geq z + \delta^2 \epsilon x \frac{1}{1-\delta}\frac{1}{1-\delta(1-\epsilon)}$$

Del cual la condición para formar un cartel es:

$$\delta \geq \frac{z}{x + z - \epsilon z}$$

Anexo D

Análisis del dilema del prisionero

D.1. Simplificaciones

Esta sección se apoya en el hecho de que los juegos de varias acciones se pueden reducir a un set de acciones relevantes, estas consisten en; jugar precio/cantidad monopólico, jugar el mejor equilibrio de Nash y jugar el mejor desvio del precio/cantidad monopolica.

Para la competencia a la Bertrand es directo.

	p_M	p_B
p_M	(1,1)	(0,2)
p_B	(2,0)	(0,0)

Tabla D.1: Pagos dilema del prisionero Bertrand

La competencia en cantidades Cournot normalizada resulta en los pagos de Axelrod.

	$q_M = 1/4$	$q_{EN} = 1/3$
$q_M = 1/4$	(1/8,1/8)	(5/48,5/36)
$q_{EN} = 1/3$	(5/36,5/48)	(1/9,1/9)

Tabla D.2: Pagos competencia Cournot

	$q_M = 1/4$	$q_{EN} = 1/3$
$p_M = 1/4$	(1,1)	(-1/2,2)
$p_{EN} = 1/3$	(2,-1/2)	(0,0)

Tabla D.3: Pagos competencia Cournot normalizado

Otra forma de simplificar Cournot consiste en considerar como segunda jugada la cantidad de desvío con respecto al equilibrio de Nash.

	$q_M = 1/4$	$q_D = 3/8$
$q_M = 1/4$	(1/8,1/8)	(3/32,9/64)
$q_D = 3/8$	(9/64,3/32)	(3/32,3/32)

Tabla D.4: Pagos competencia Cournot con desvío

Normalizado se obtiene:

	$q_M = 1/4$	$q_D = 3/8$
$p_M = 1/4$	(1,1)	(0,1.5)
$p_D = 3/8$	(1.5,0)	(0,0)

Tabla D.5: Pagos competencia Cournot con desvío normalizado

Además, para el juego de Cournot se pueden considerar los pagos contingentes de la estrategia gatillo convencional, si bien esto implica que en cada cuadro se juega una cantidad diferente (los rótulos pierden sentido) también nos entrega un dilema del prisionero:

	Mon	D/C
Mon	(1/8,1/8)	(3/32,9/64)
D/C	(9/64,3/32)	(1/9,1/9)

Tabla D.6: Pagos competencia Cournot con pagos efectivos

Normalizado se obtiene:

	Mon	D/C
Mon	(1,1)	(-1.25,2.12)
D/C	(2.12,-1.25)	(0,0)

Tabla D.7: Pagos competencia cournot con pagos efectivos normalizado

Se puede proceder de forma similar con los pagos obtenidos de modelo propuesto por Calvano, la tabla completa con los pagos se puede ver a continuación:

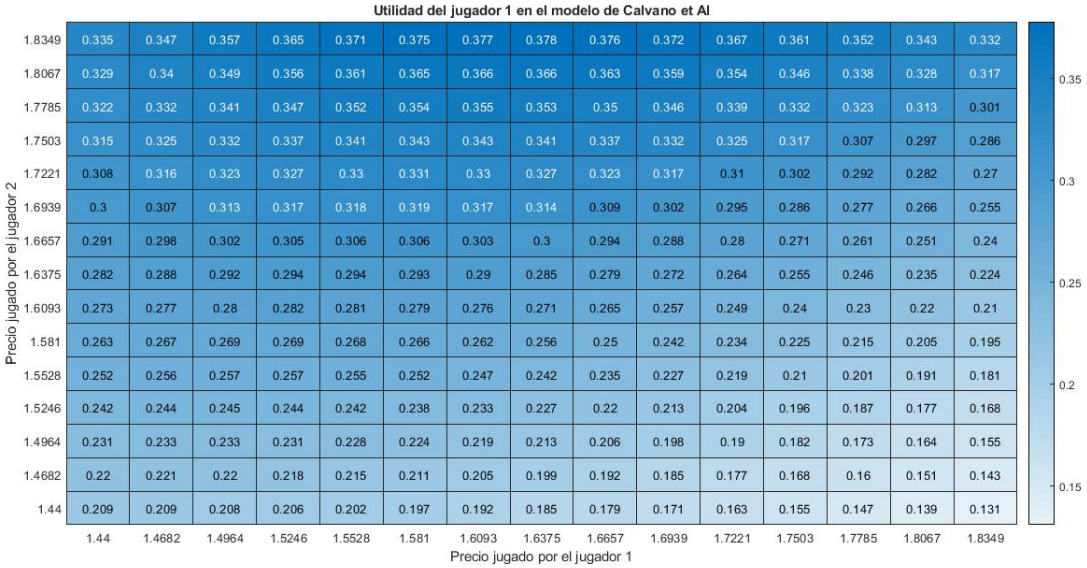


Figura D.1: Utilidades modelo Calvano

	p_M	p_{EN}
p_M	(0.332,0.332)	(0.131,0.335)
p_{EN}	(0.335,0.209)	(0.209,0.209)

Tabla D.8: Pagos Calvano

	p_M	p_L
p_M	(1,1)	(-0.634,1.024)
p_L	(1.024,-0.634)	(0,0)

Tabla D.9: Pagos Calvano normalizado

Considerando como segunda jugada la mejor respuesta del precio monopólico.

	p_M	p_{MR}
p_M	(0.332,0.332)	(0.221,0.378)
p_{MR}	(0.378,0.221)	(0.271,0.271)

Tabla D.10: Pagos Calvano restringido

	p_M	p_{MR}
p_M	(1,1)	(-0.819,1.754)
p_{MR}	(1.754,-0.819)	(0,0)

Tabla D.11: Pagos Calvano restringido normalizado

Repetir la tercera simplificación hecha en Cournot no nos entrega un dilema del prisionero en el modelo de Calvano, esto debido a que los pagos del equilibrio de nash resultan ser menores al obtenido por ser traicionado.

D.2. Simulaciones

Se considera una grilla de 30 x 29 juegos, 30 valores de d entre 0 y 0.5 y 29 valores de t entre 0 y 0.5. Para todos los elementos de la grilla se realizan 1000 experimentos de 50000 periodos cada uno, además, se utiliza una tasa de exploración de $\alpha = 0.25$ y $\beta = 10^{-4}$. Con ello se evalúa la utilidad obtenida por los algoritmos en las ultimas 1000 repeticiones del juego.

D.2.1. Dilema del prisionero $K = 0$

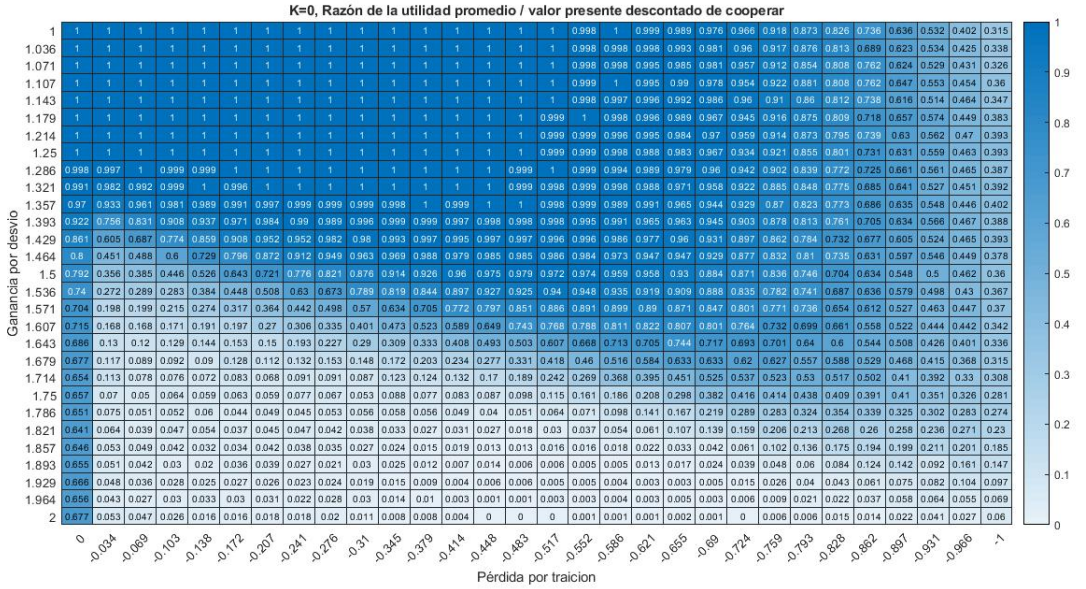


Figura D.2: Razón de utilidad percibida con respecto a la monopólica $k = 0$

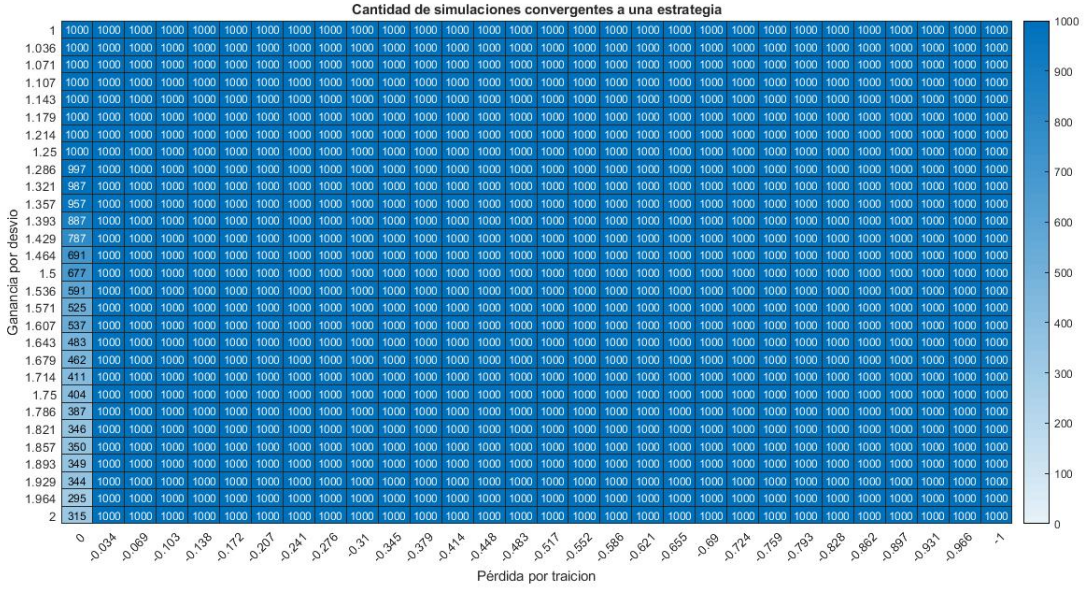


Figura D.3: Tasa se convergencias a estrategias $k = 0$

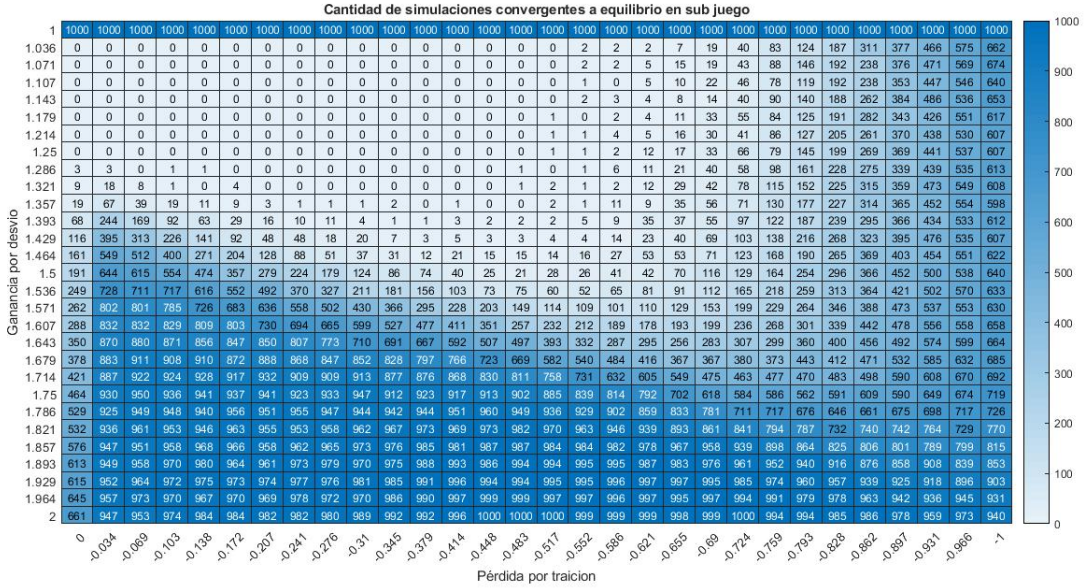


Figura D.4: Tasa de convergencias a EPS $k = 0$

D.2.2. Dilema del prisionero $K = 1$

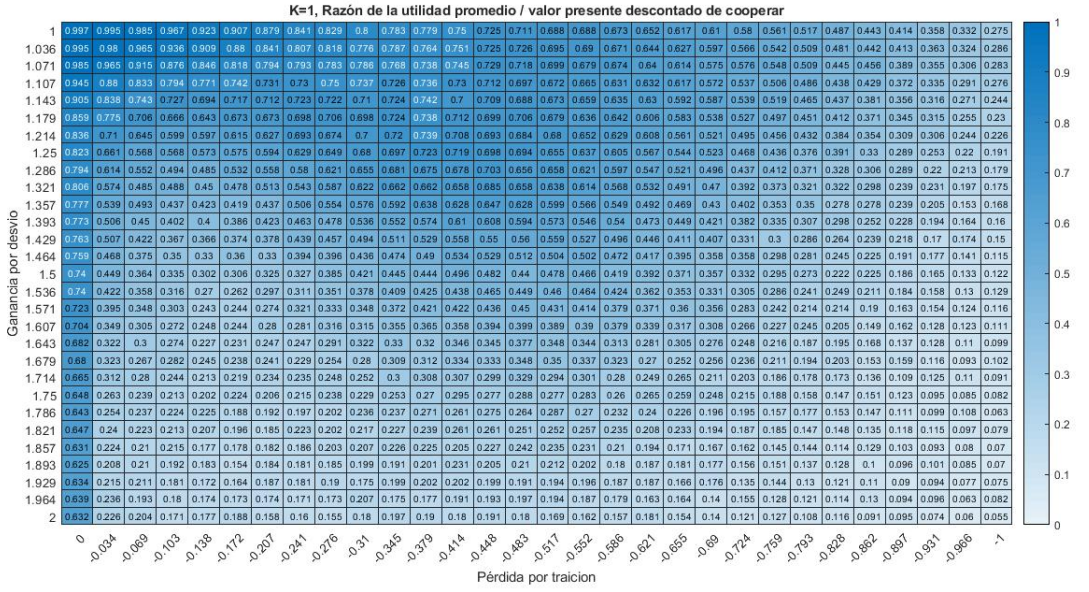


Figura D.5: Razón de utilidad percibida con respecto a la monopólica $k = 1$

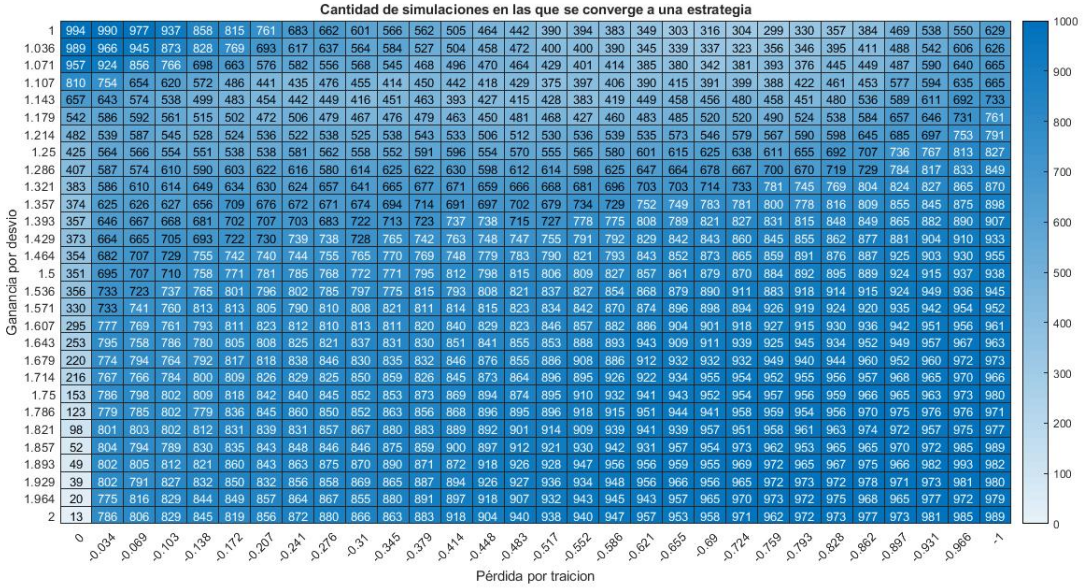


Figura D.6: Tasa de convergencias a estrategias $k = 1$

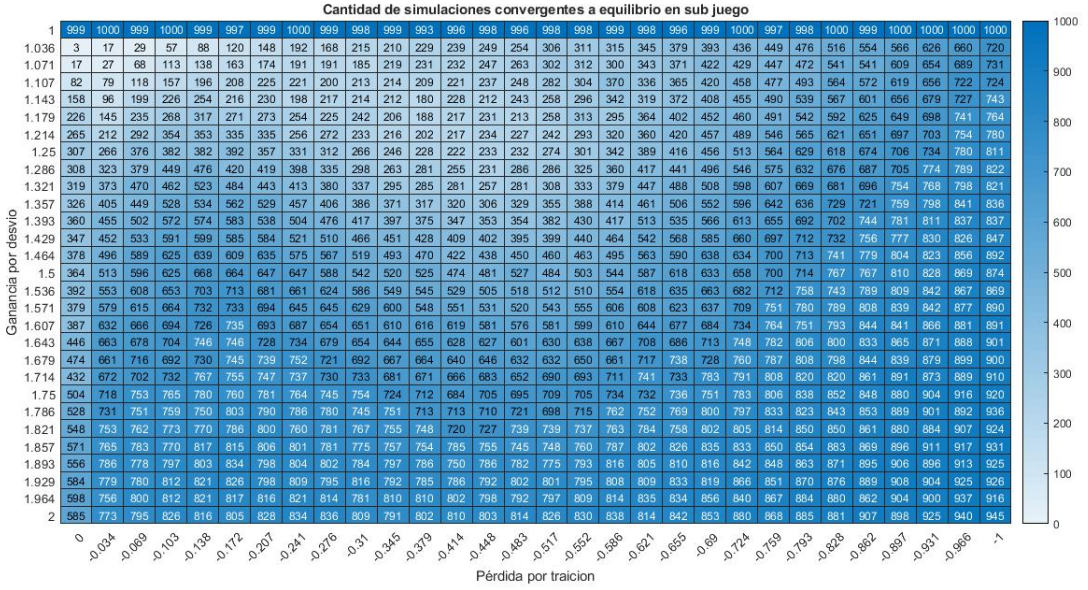


Figura D.7: Tasa de convergencias a EPS $k = 1$

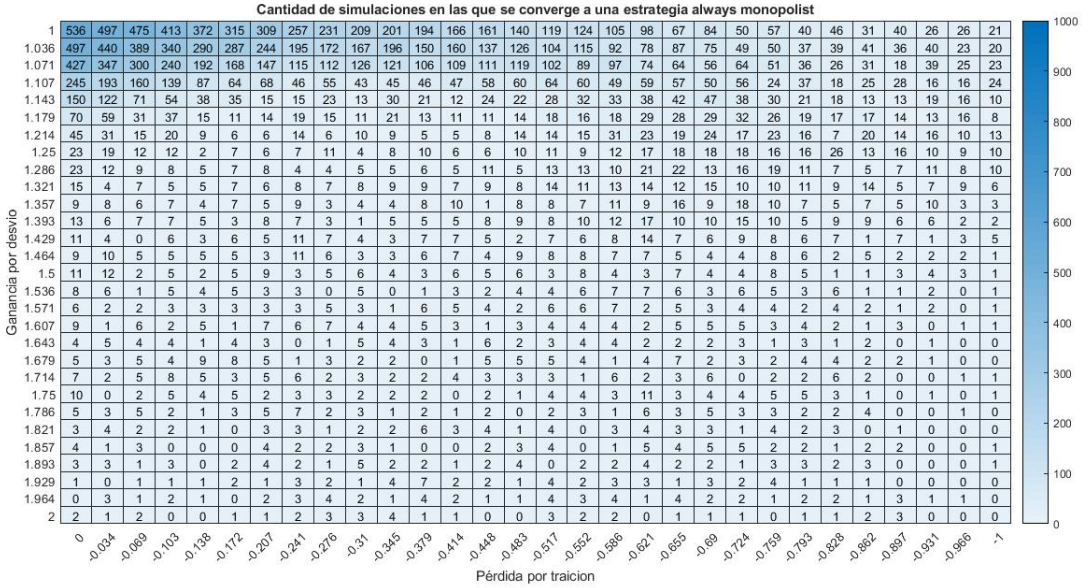


Figura D.8: Tasa de convergencias a estrategias de precio monopólico $k = 1$

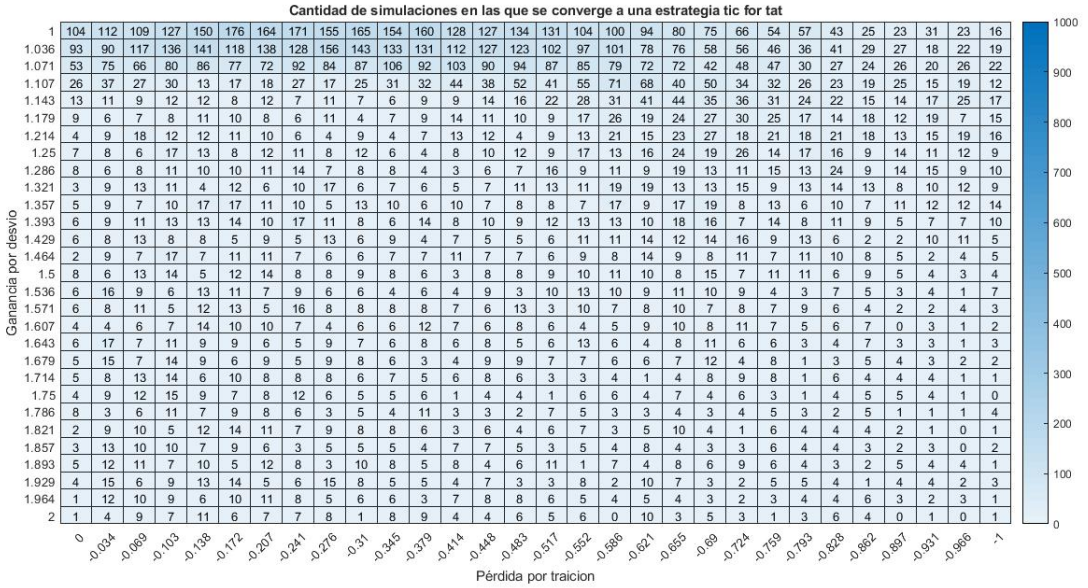


Figura D.9: Tasa de convergencias a estrategia Tit for Tat $k = 1$

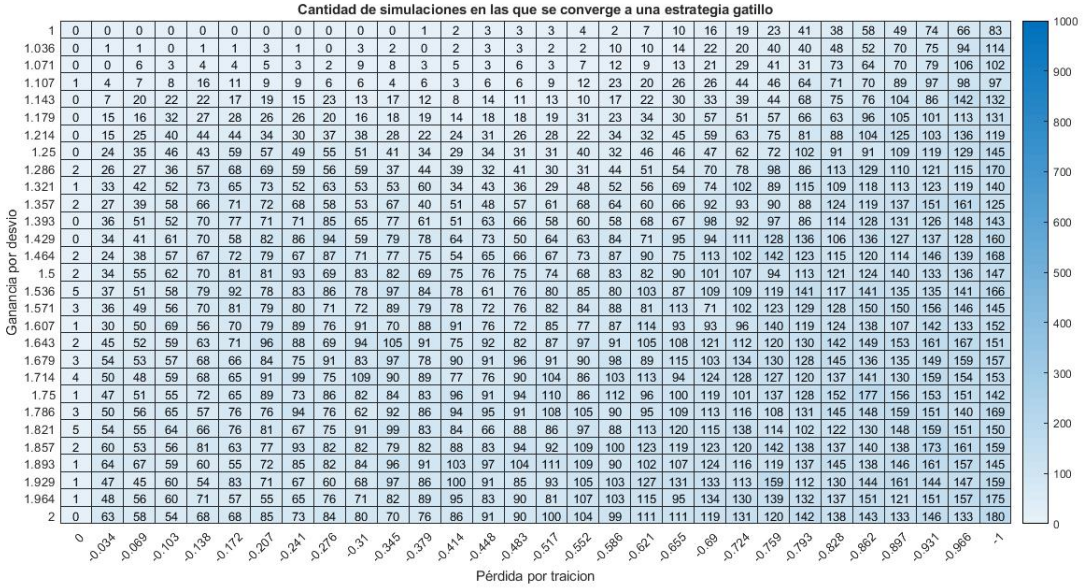


Figura D.10: Tasa de convergencias a estrategia gatillo $k = 1$

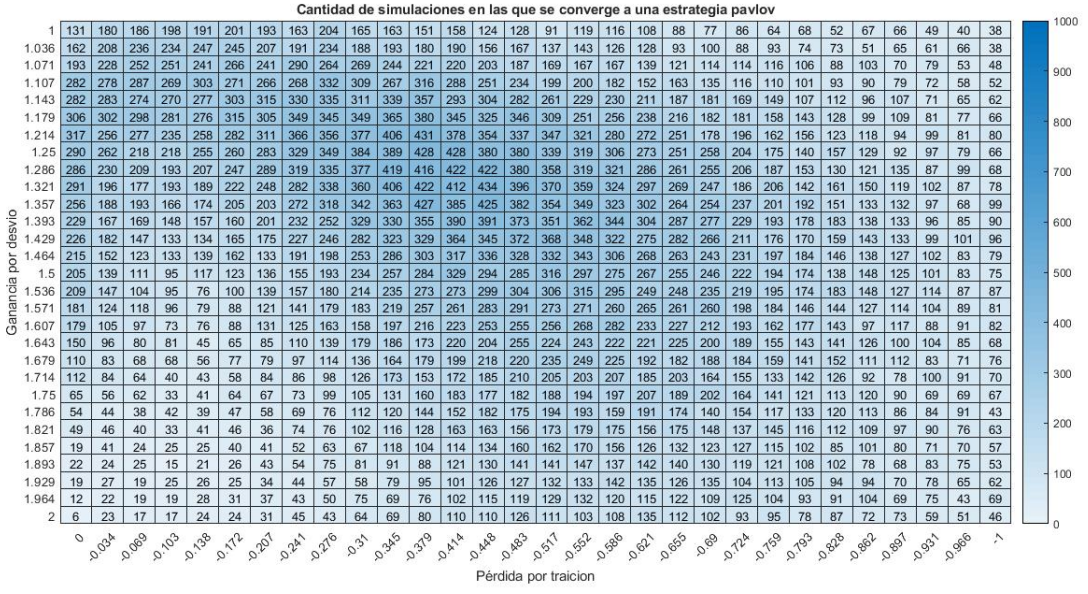


Figura D.11: Tasa de convergencias a estrategia pavlov $k = 1$

D.2.3. Dilema del prisionero $K = 2$

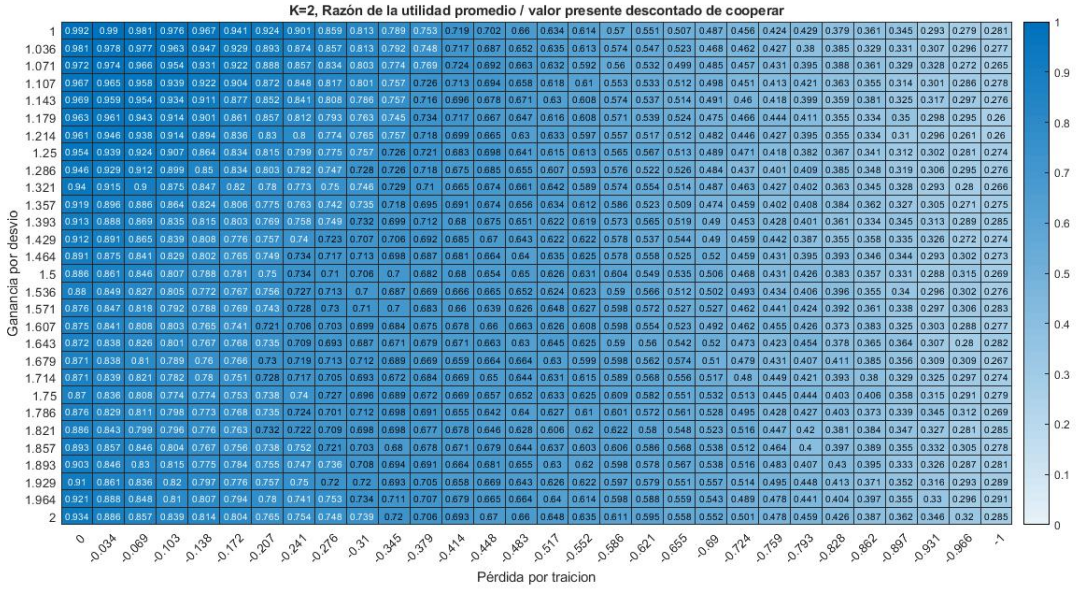


Figura D.12: Razón de utilidad percibida con respecto a la monopólica $K = 2$

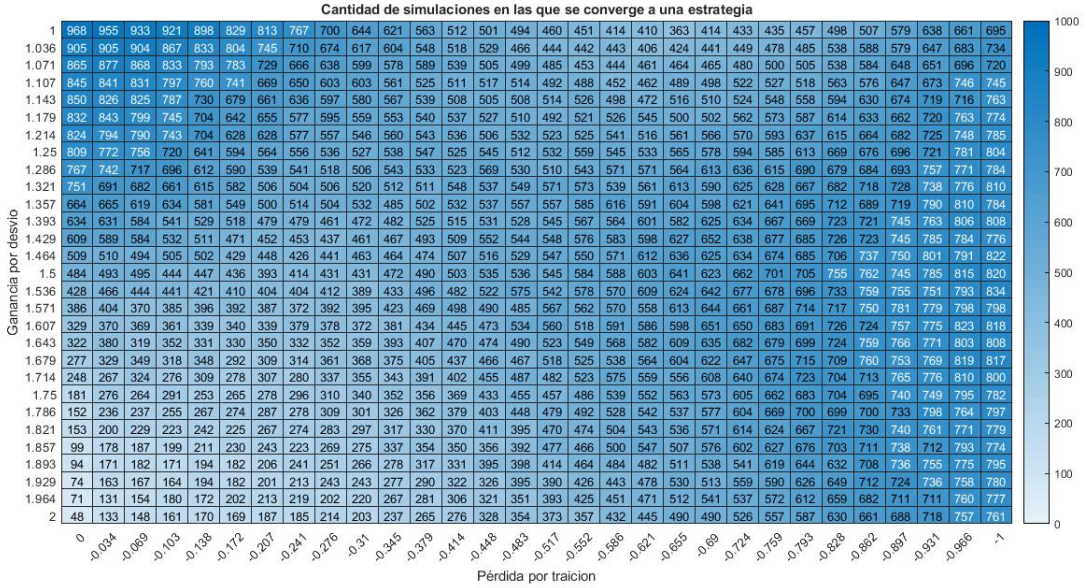


Figura D.13: Tasa de convergencias $K = 2$

D.2.4. Dilema del prisionero $K = 3$

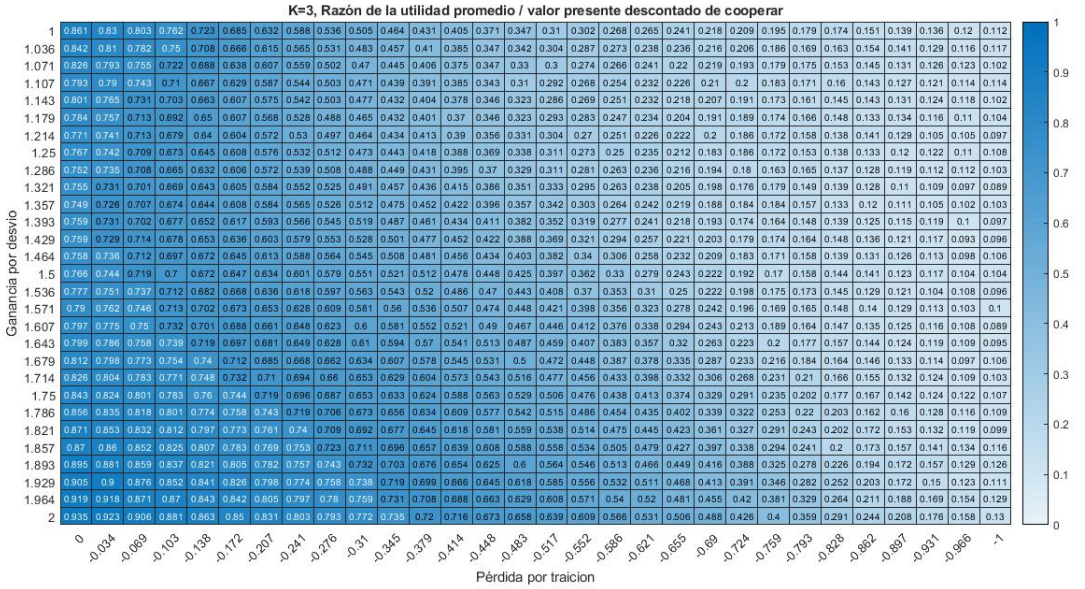


Figura D.14: Razón de utilidad percibida con respecto a la monopólica $K = 3$

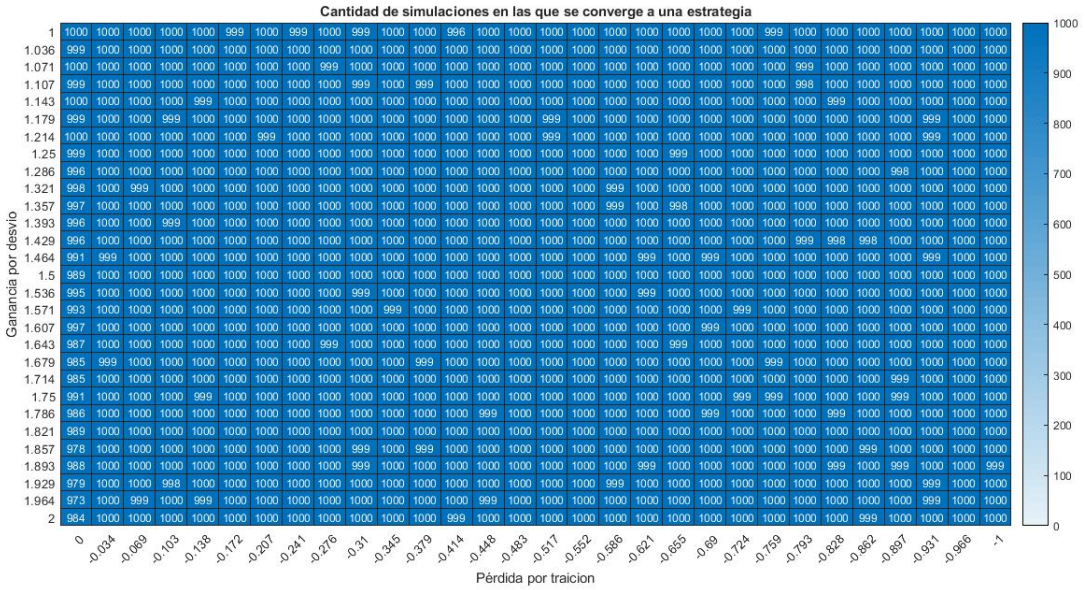


Figura D.15: Tasa de convergencias $K = 3$

Anexo E

Calibración en dilema del prisionero

Nuevamente verificamos la convergencia para el dilema del prisionero repetido, esta vez solo consideramos hasta una memoria de $k = 1$ pues es objetivo de la tesis se centra en la comparación con $k = 0$. Las variables de control son las variables de calibración de los algoritmos Q-learning, esto es la tasa de aprendizaje α y la tasa de decaimiento de la exploración β.

Se presenta solo el perfil de pagos del dilema del prisionero de Axelrod, sin embargo los resultados son similares para otros perfiles de pago testeados.

E.1. Dilema de Axelrod

E.1.1. Dilema de Axelrod $K = 0$

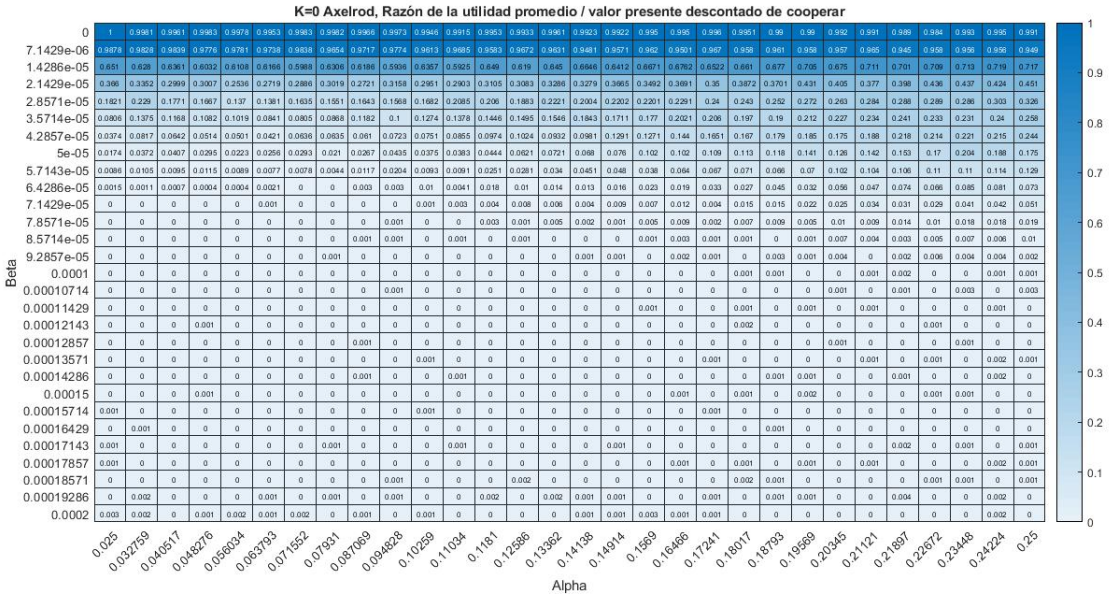


Figura E.1: Razón de utilidad obtenida en el dilema de Axelrod $K = 0$

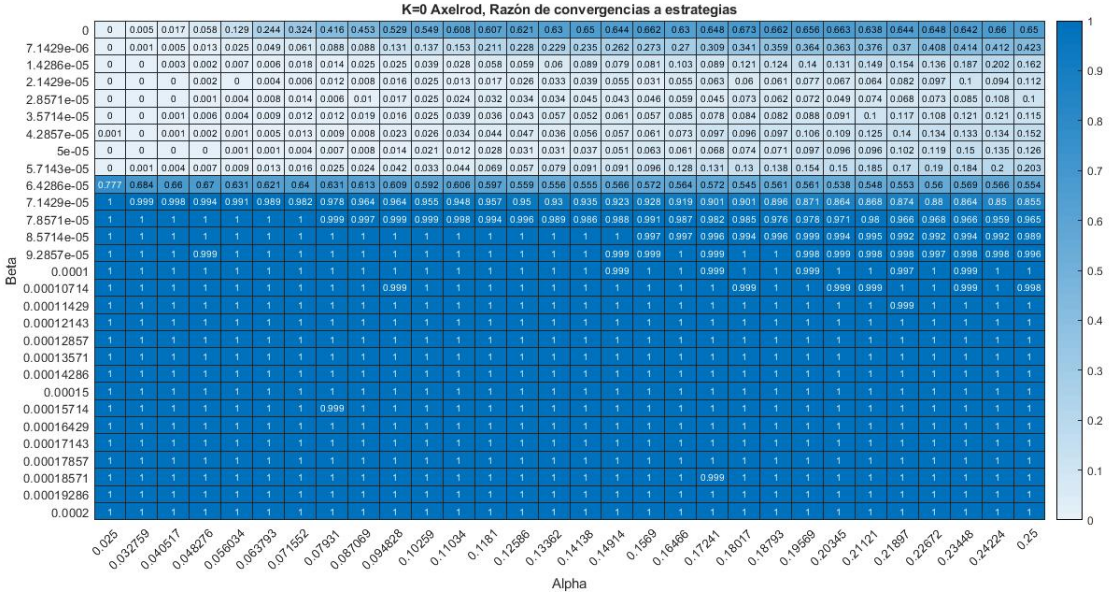


Figura E.2: Tasa de convergencias a estrategia en el dilema de Axelrod $K = 0$

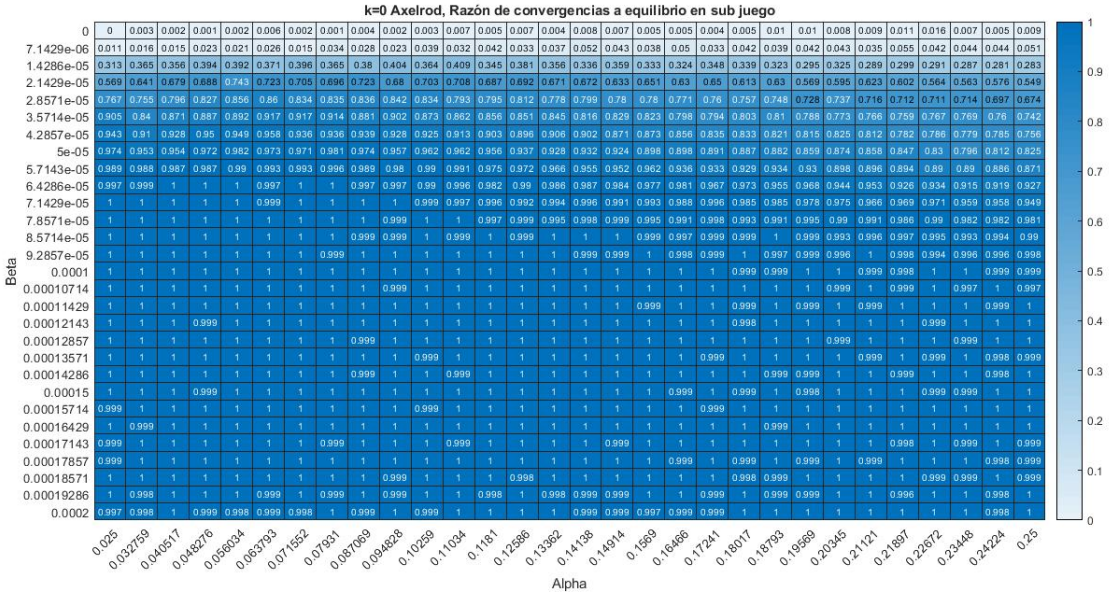


Figura E.3: Tasa de convergencias a equilibrio en el dilema de Axelrod $K = 0$

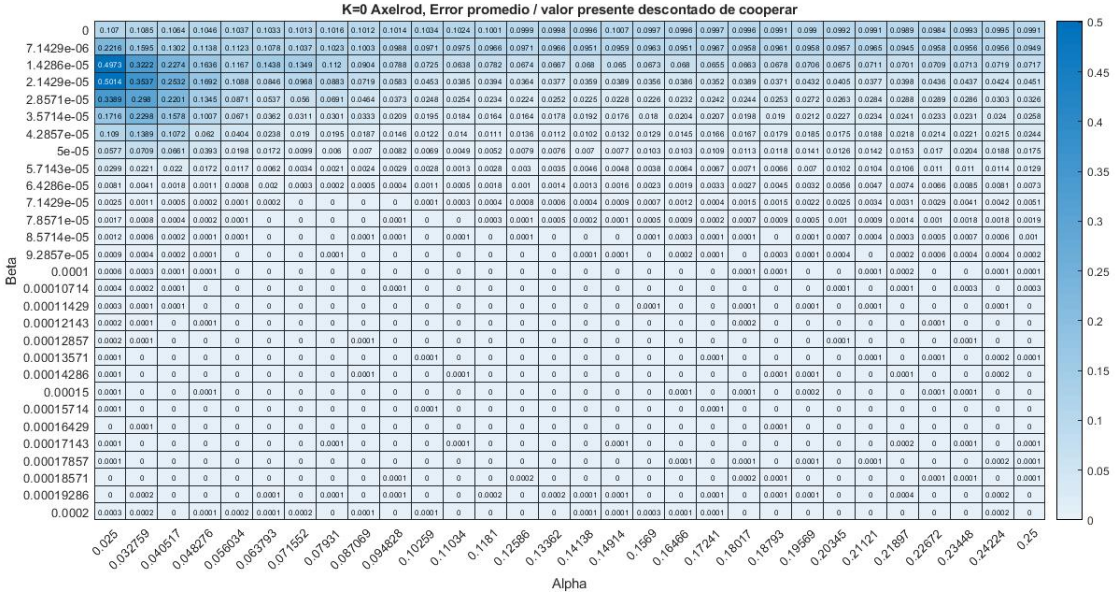


Figura E.4: Tasa de error en la obtención de los valores de continuación en el dilema de Axelrod $K = 0$

E.1.2. Dilema de Axelrod $K = 1$

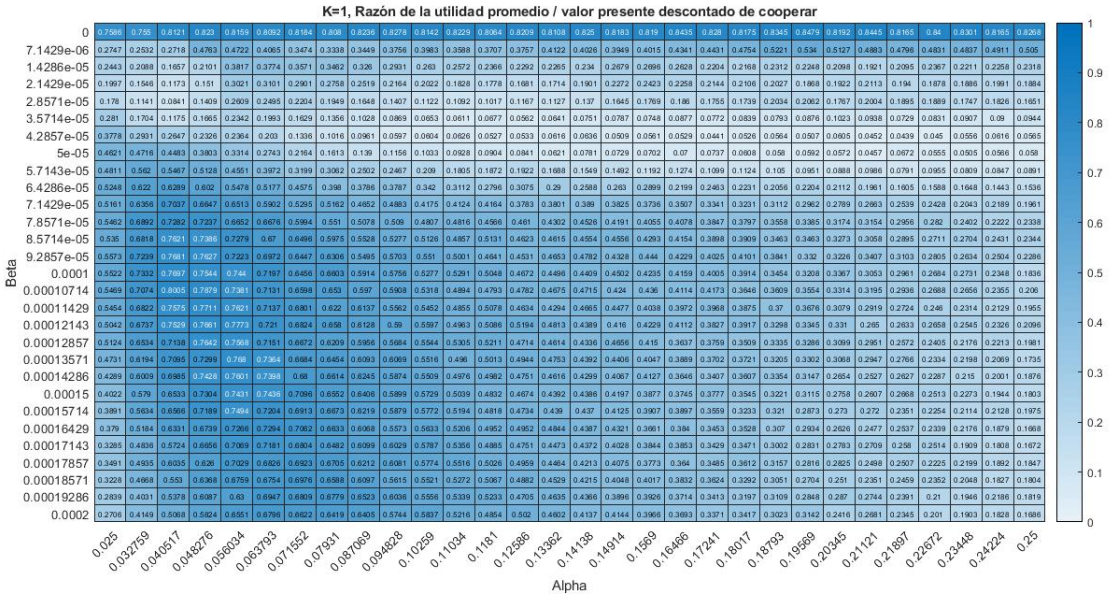


Figura E.5: Razón de utilidad obtenida en el dilema de Axelrod $K = 1$

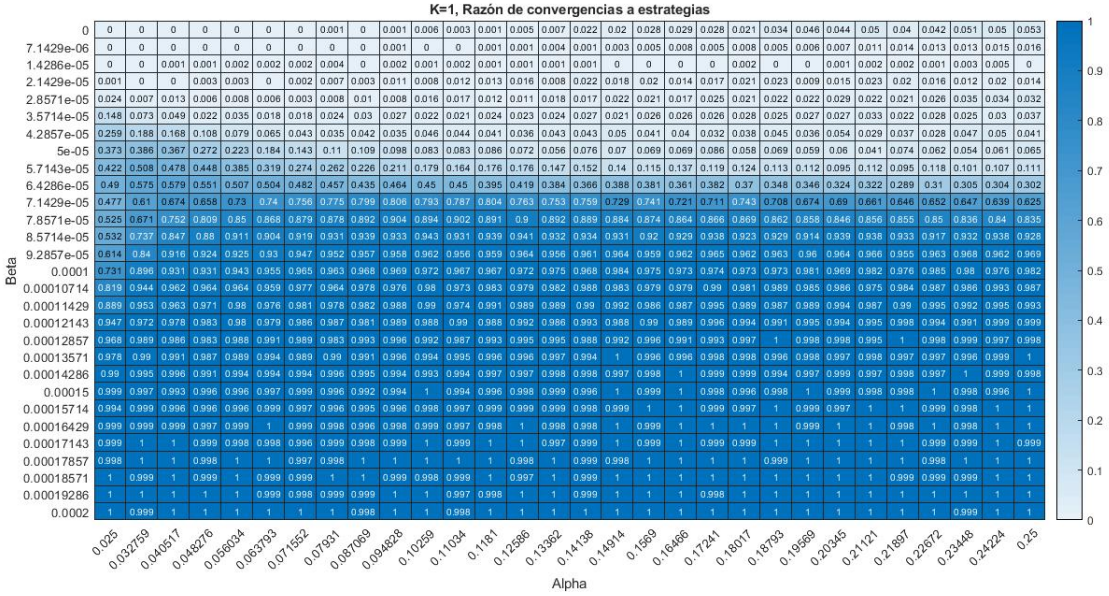


Figura E.6: Tasa de convergencias a estrategia en el dilema de Axelrod $K = 1$

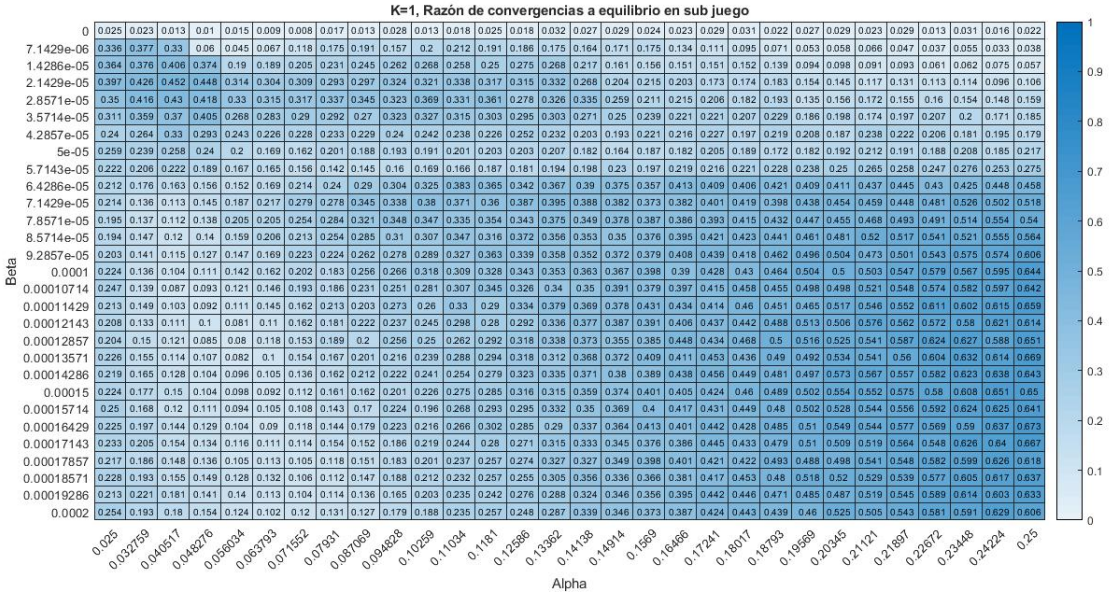


Figura E.7: Tasa de convergencias a equilibrio en el dilema de Axelrod $K = 1$

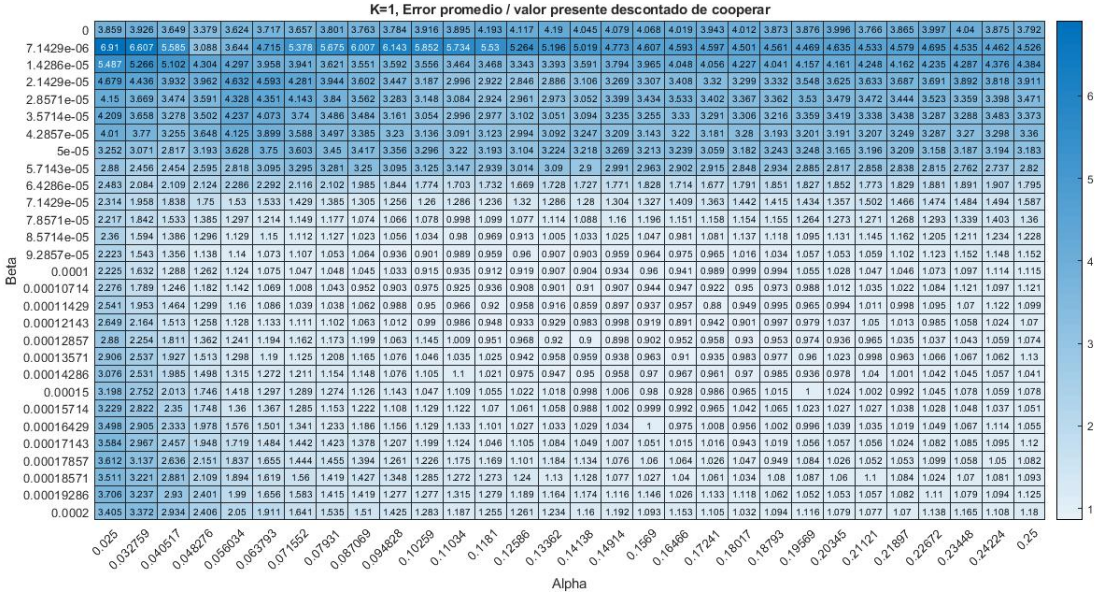


Figura E.8: Tasa de error en la obtención de los valores de continuación en el dilema de Axelrod $K = 1$

Anexo F

Resultados de ejemplos

F.1. Resultados ejemplo 1

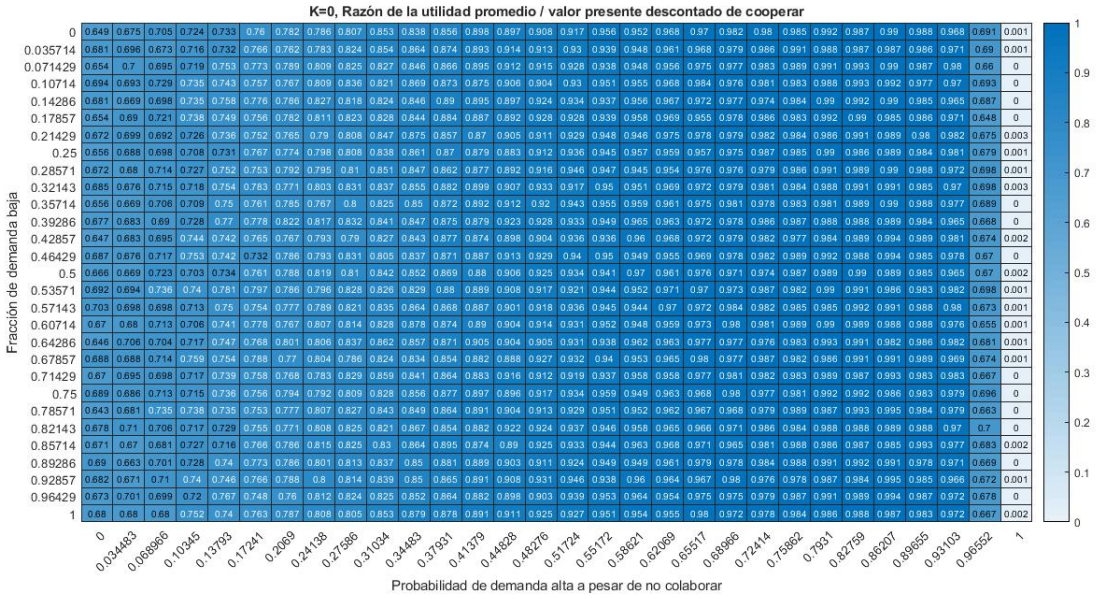


Figura F.1: Razón de utilidad en ejemplo 1, $K = 0$

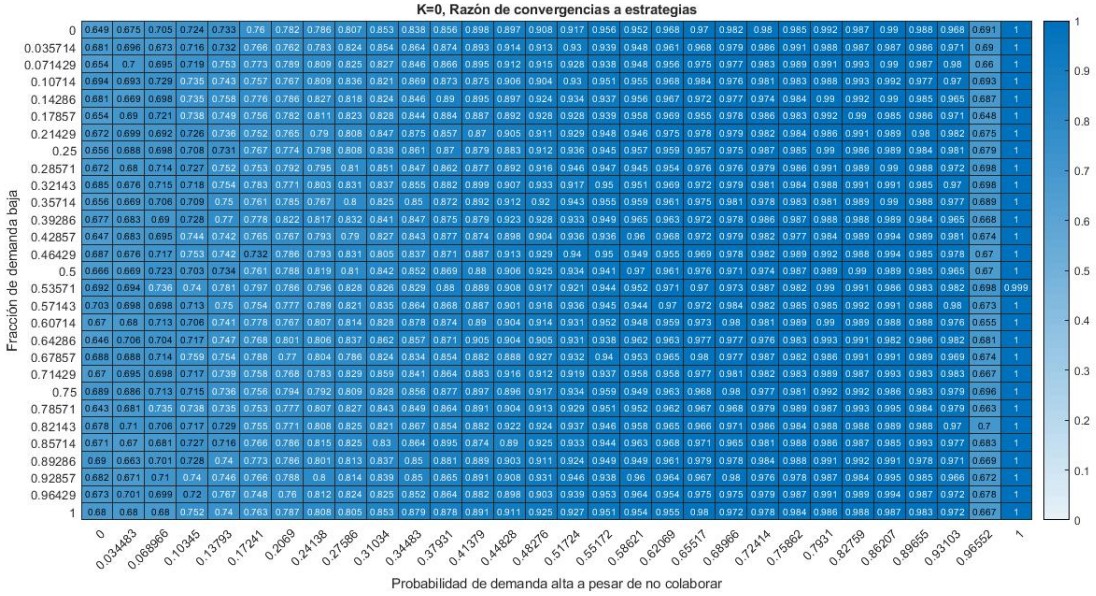


Figura F.2: Tasa de convergencias a estrategia en ejemplo 1, $K = 0$

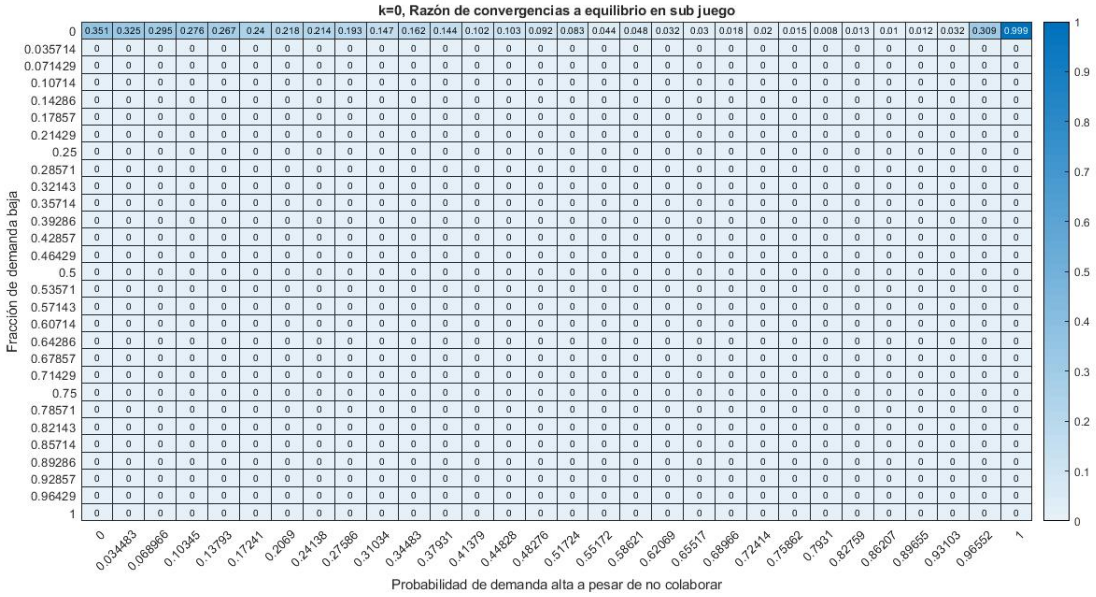


Figura F.3: Tasa de convergencias a equilibrio en ejemplo 1, $K = 0$

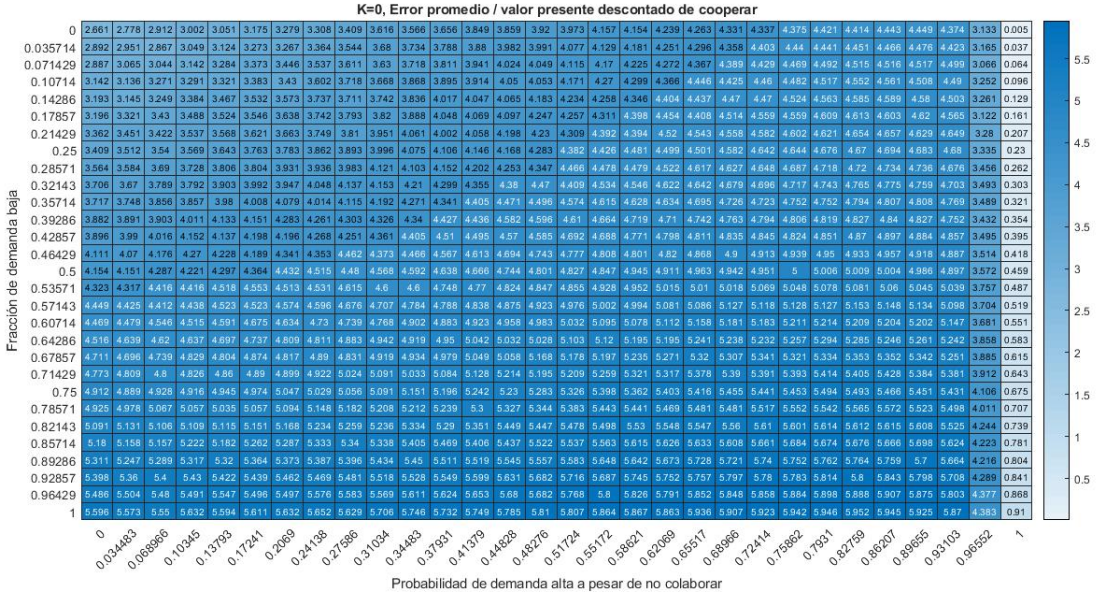


Figura F.4: Tasa de error en la obtención de los valores de continuación en ejemplo 1, $K = 0$

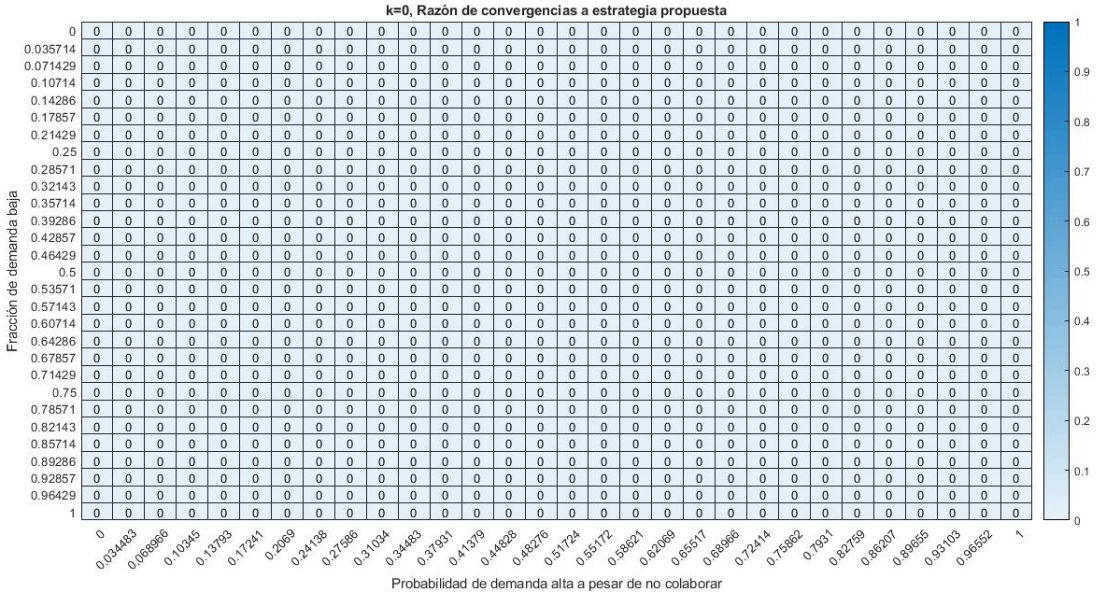


Figura F.5: Tasa de convergencia a la estrategia propuesta en ejemplo 1, $K = 0$

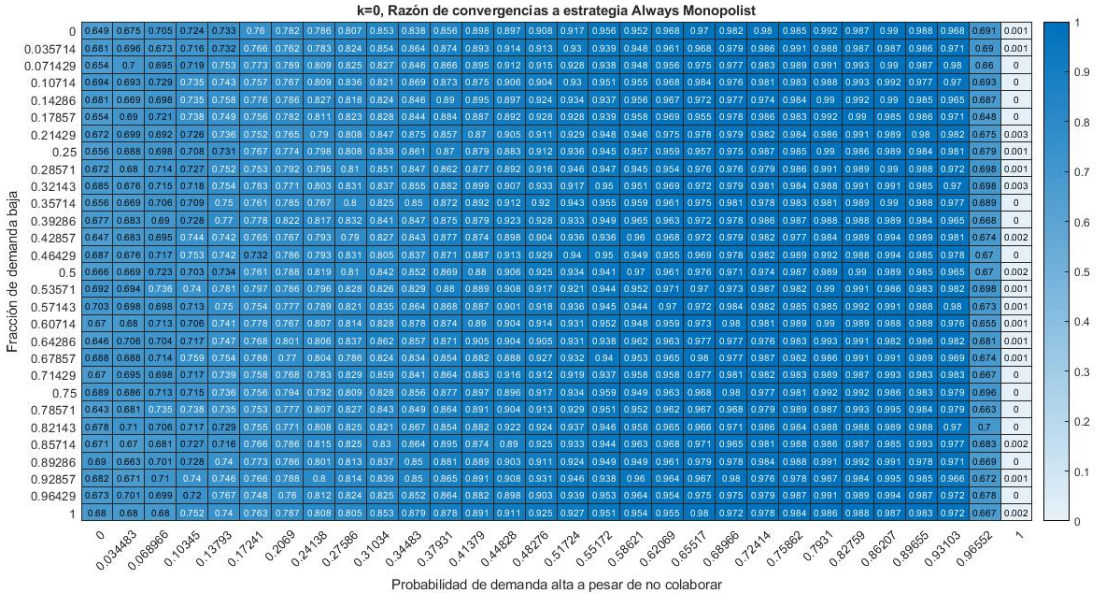


Figura F.6: Tasa de convergencia a estrategia always monopolist en ejemplo 1, $K = 0$

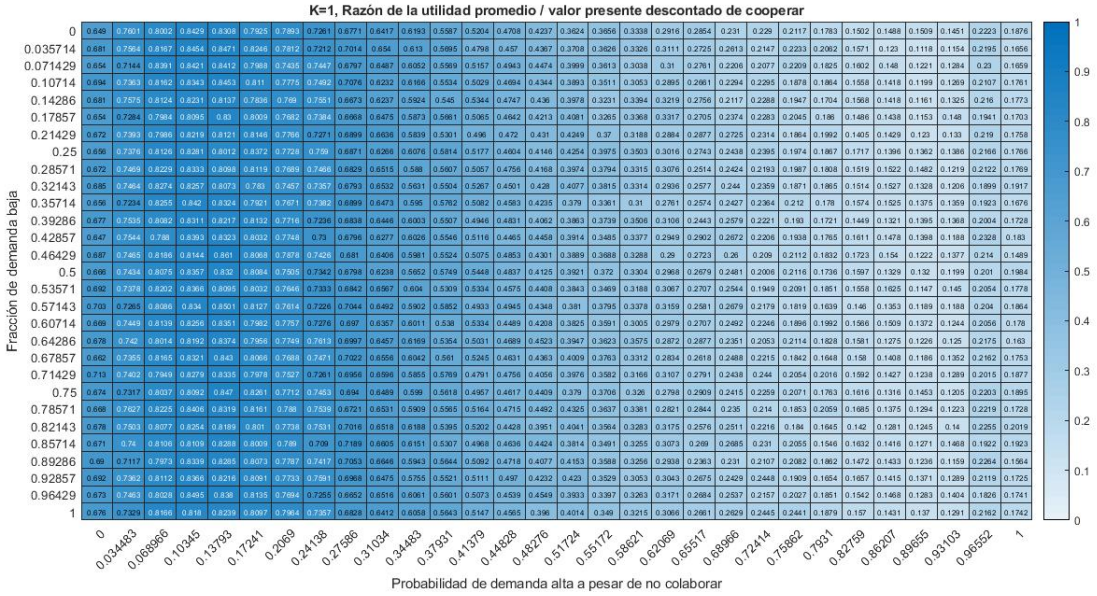


Figura F.7: Razón de utilidad en ejemplo 1, $K = 1$

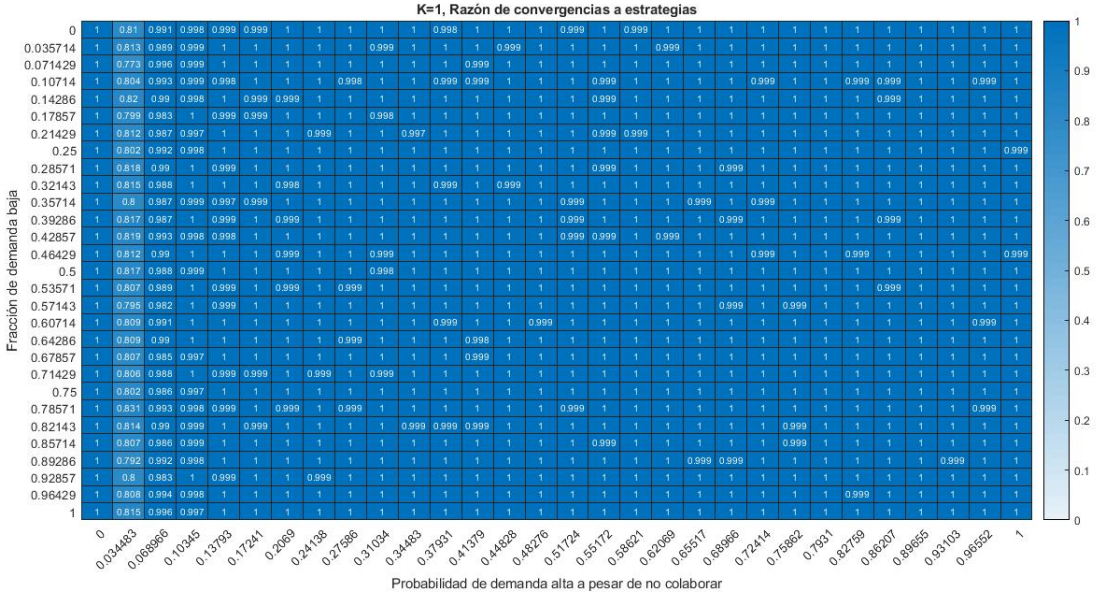


Figura F.8: Tasa de convergencias a estrategia en ejemplo 1, $K = 1$

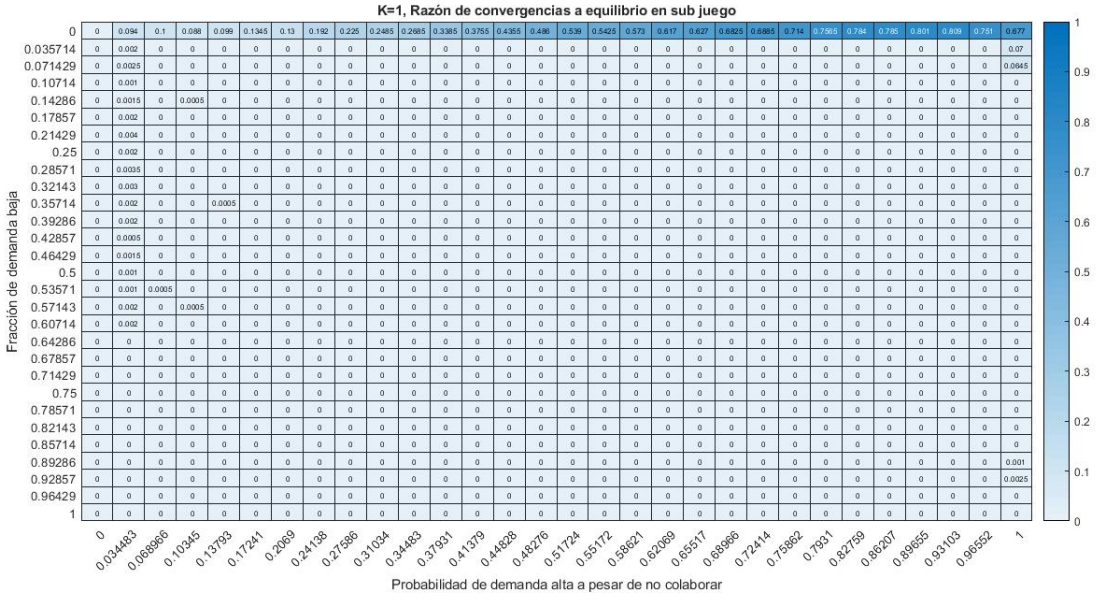


Figura F.9: Tasa de convergencias a equilibrio en ejemplo 1, $K = 1$

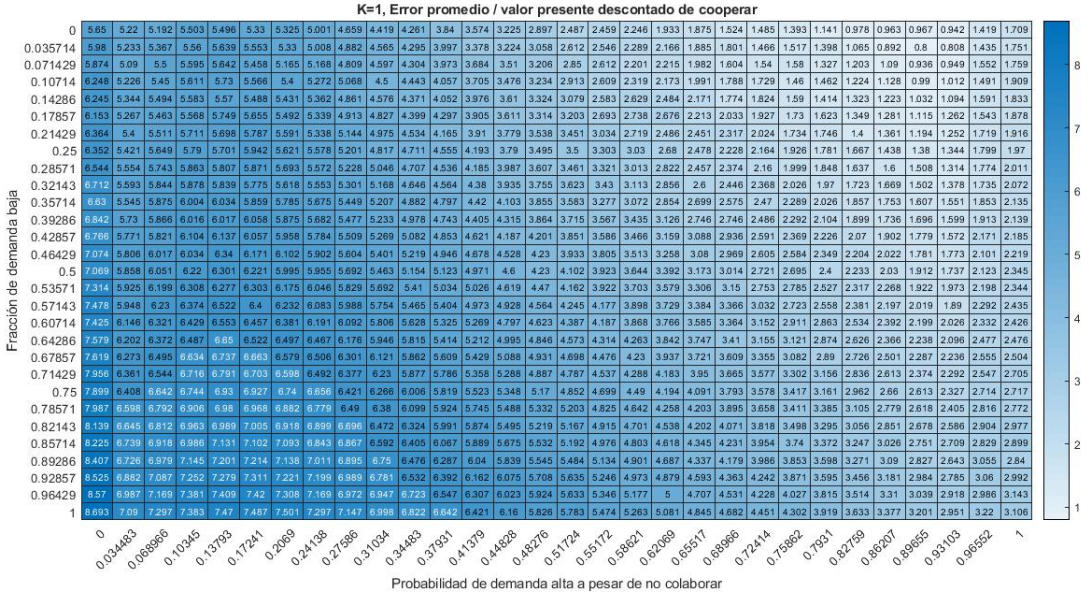


Figura F.10: Tasa de error en la obtención de los valores de continuación en ejemplo 1, $K = 1$

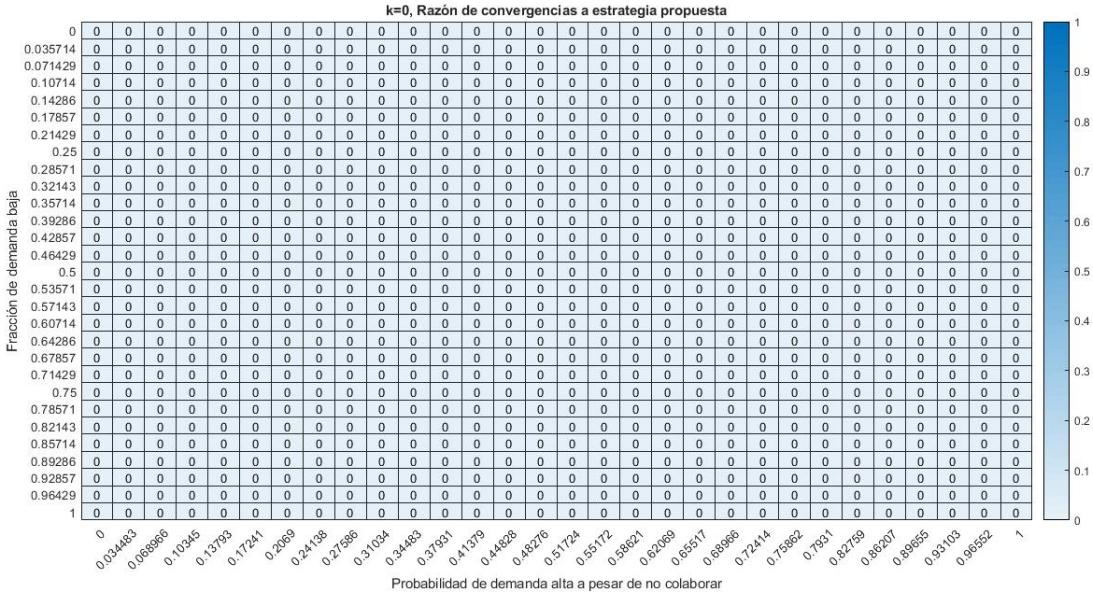


Figura F.11: Tasa de convergencia a la estrategia propuesta en ejemplo 1, $K = 1$

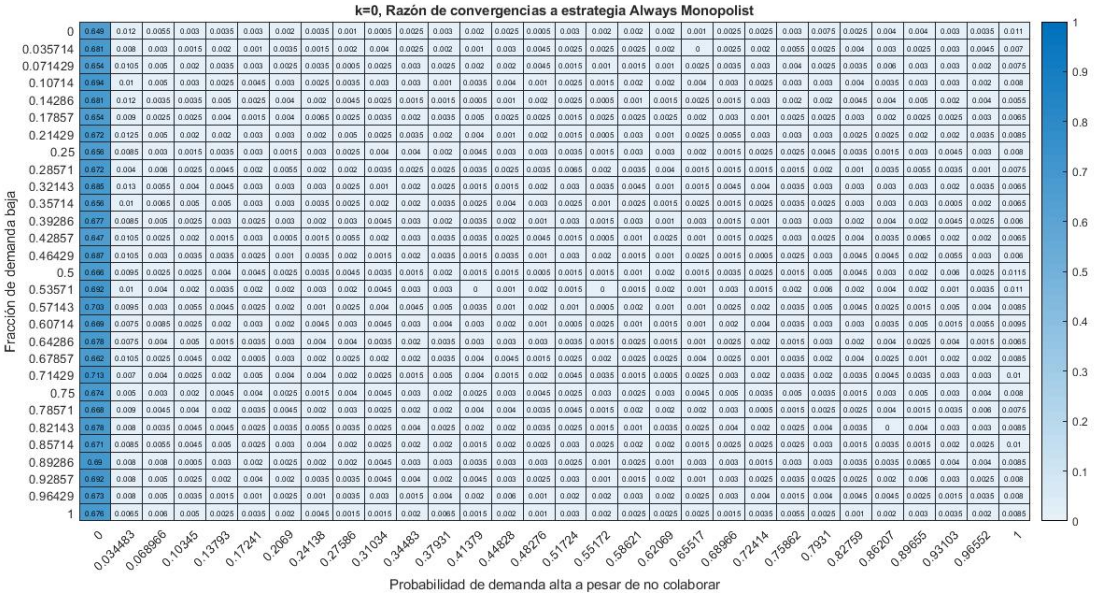


Figura F.12: Tasa de convergencia a estrategia always monopolist en ejemplo 1, $K = 1$

F.2. Resultados ejemplo 2

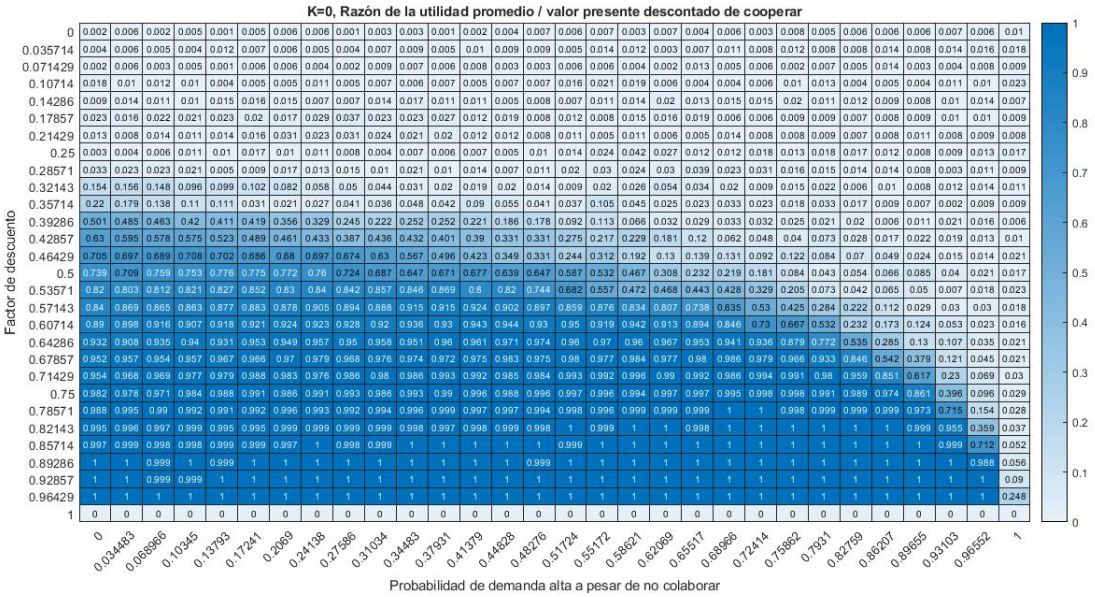


Figura F.13: Razón de utilidad en ejemplo 2, $K = 0$

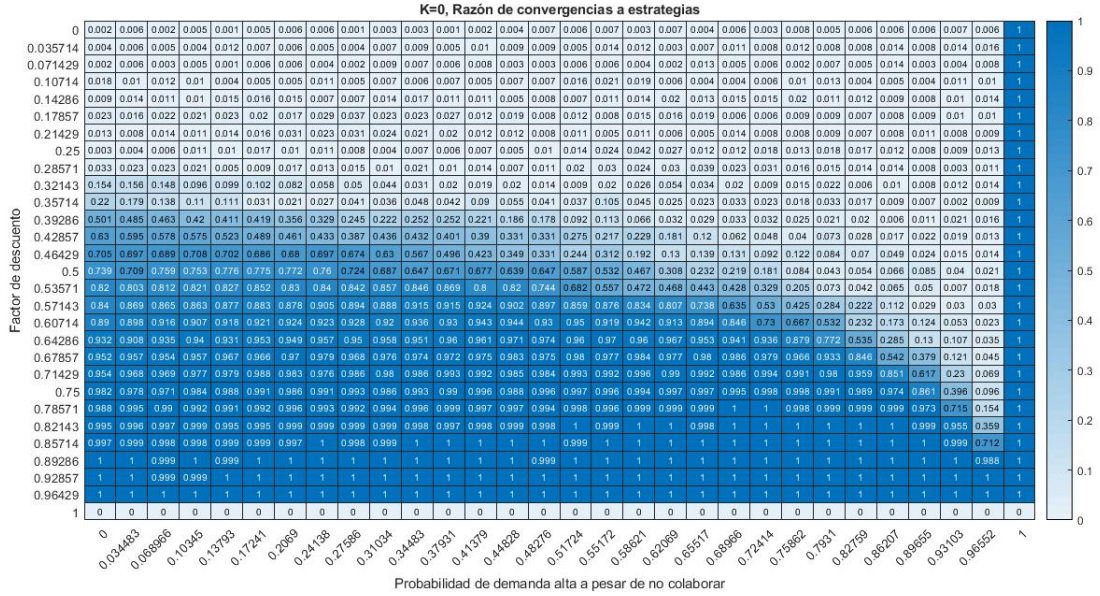


Figura F.14: Tasa de convergencias a estrategia en ejemplo 2, $K = 0$

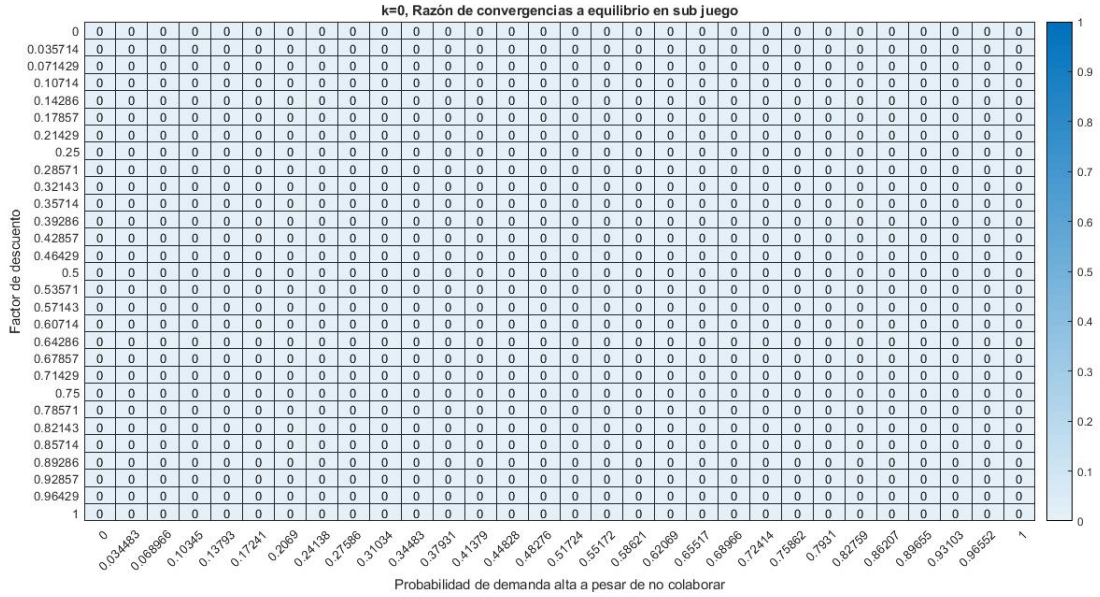


Figura F.15: Tasa de convergencias a equilibrio en ejemplo 2, $K = 0$

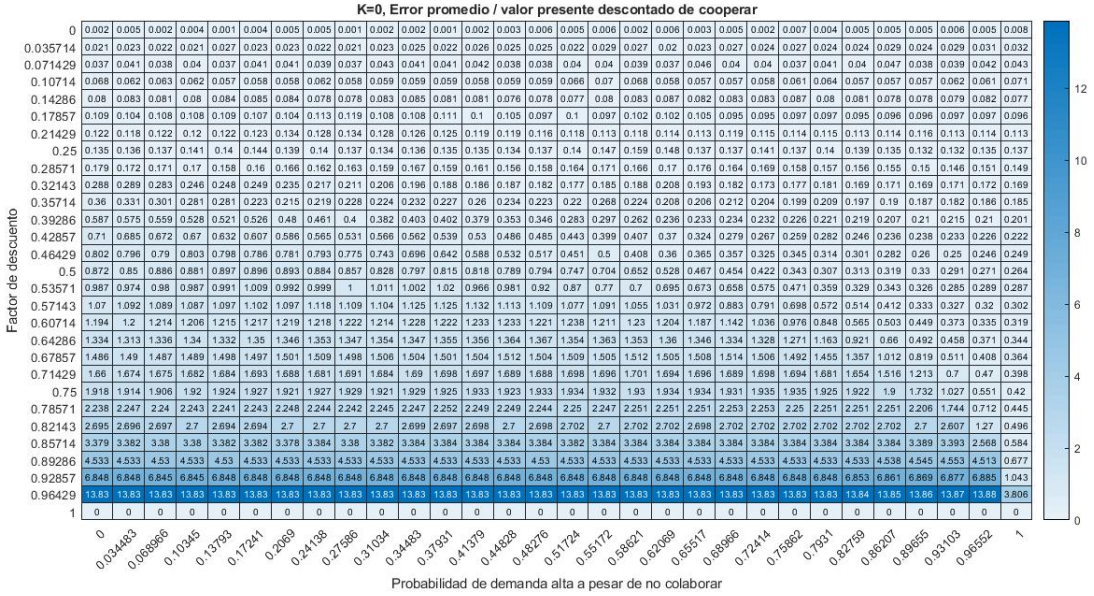


Figura F.16: Tasa de error en la obtención de los valores de continuación en ejemplo 2, $K = 0$

Figura F.17: Tasa de convergencia a la estrategia propuesta en ejemplo 2, $K = 0$

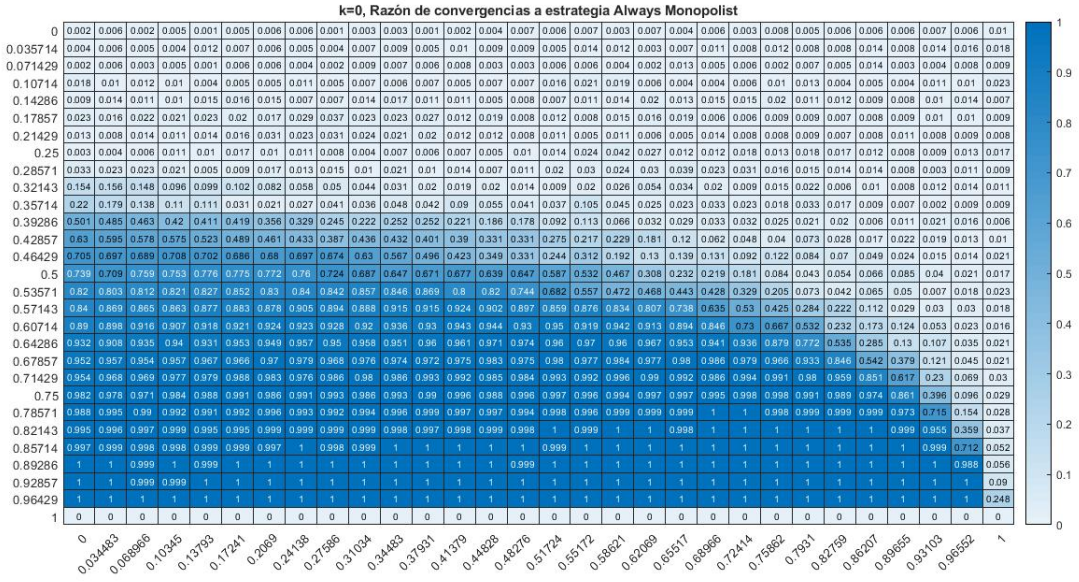


Figura F.18: Tasa de convergencia a estrategia always monopolist en ejemplo 2, $K = 0$

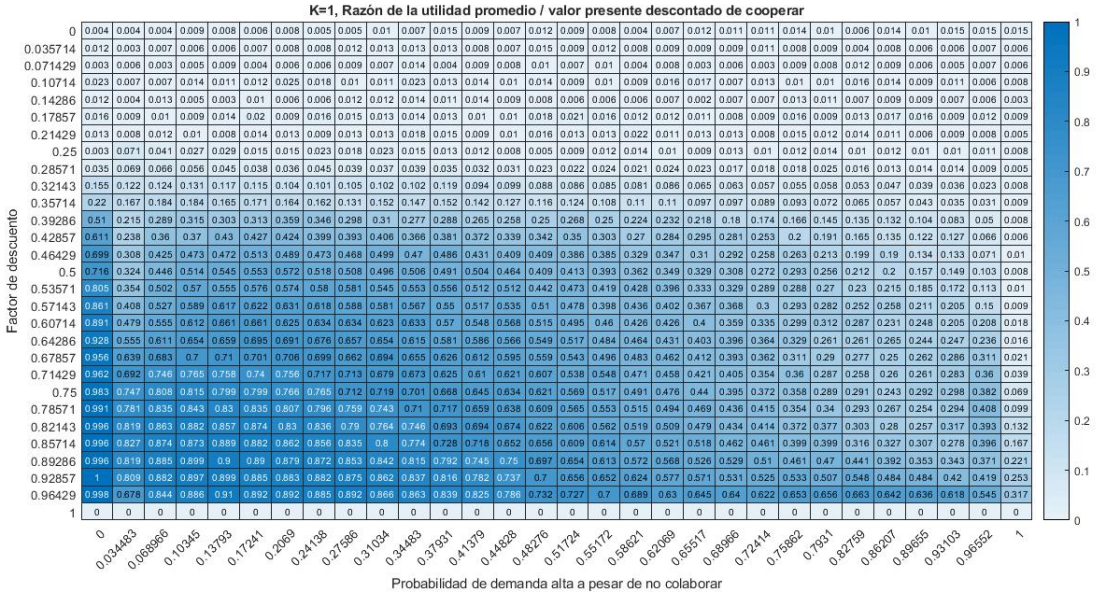


Figura F.19: Razón de utilidad en ejemplo 1, $K = 0$

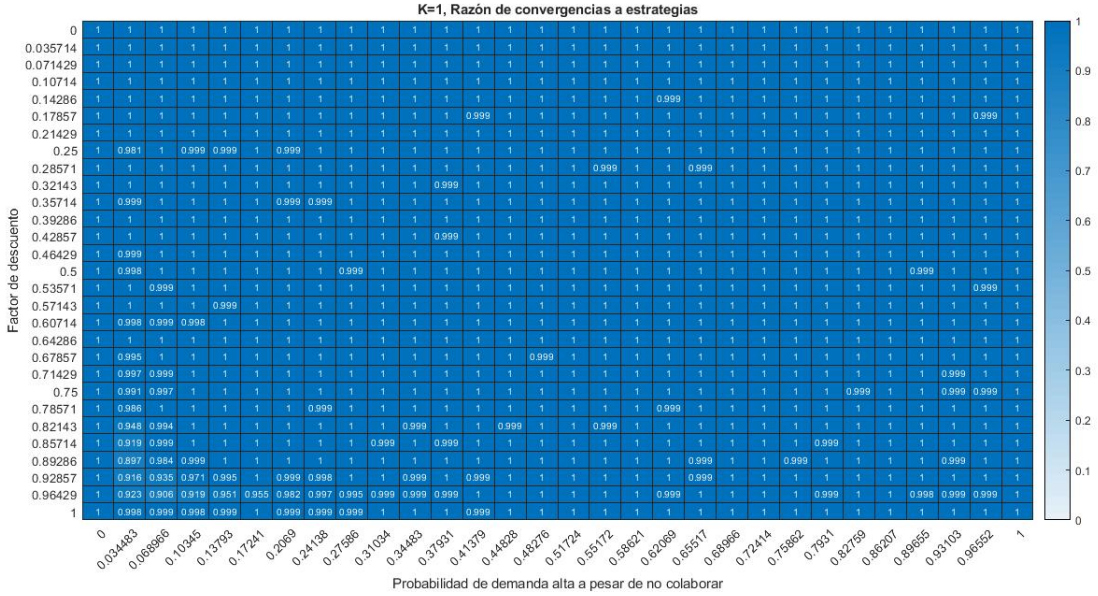


Figura F.20: Tasa de convergencias a estrategia en ejemplo 1, $K = 0$

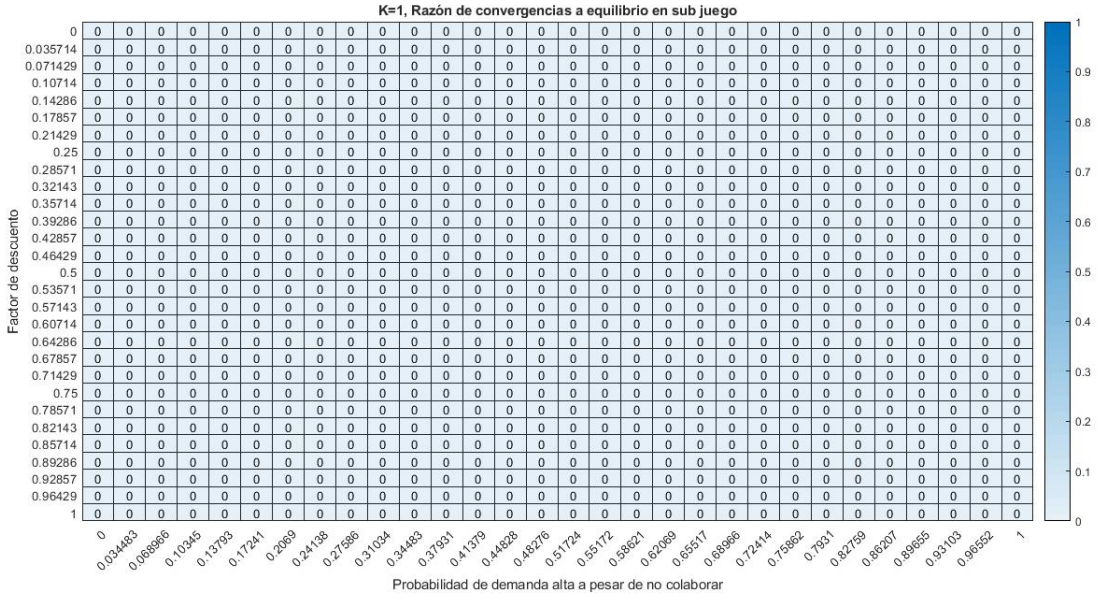


Figura F.21: Tasa de convergencias a equilibrio en ejemplo 1, $K = 0$

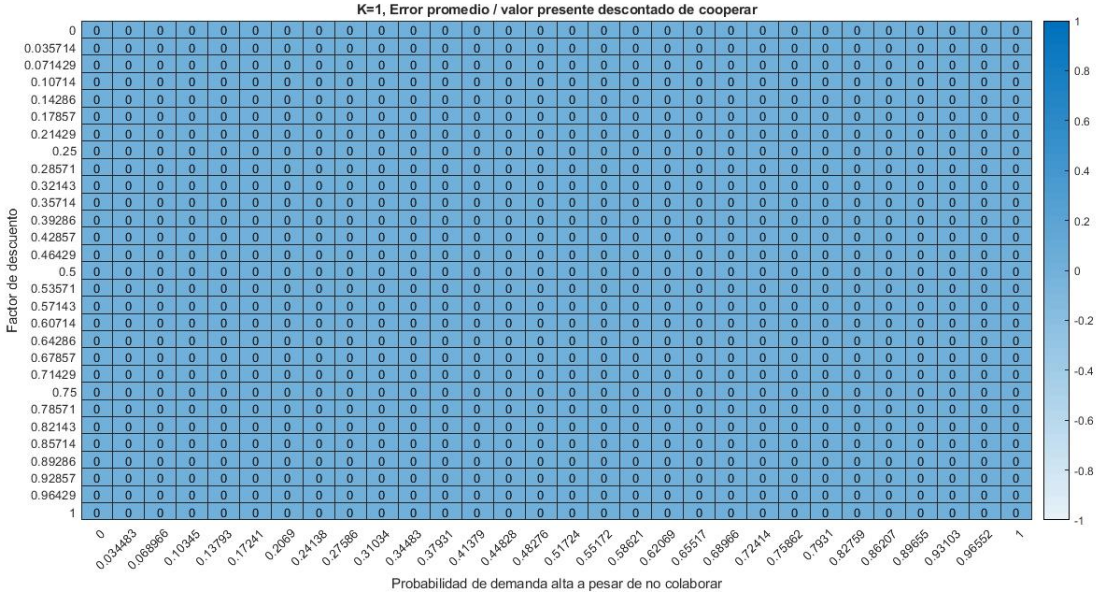


Figura F.22: Tasa de error en la obtención de los valores de continuación en ejemplo 1, $K = 0$

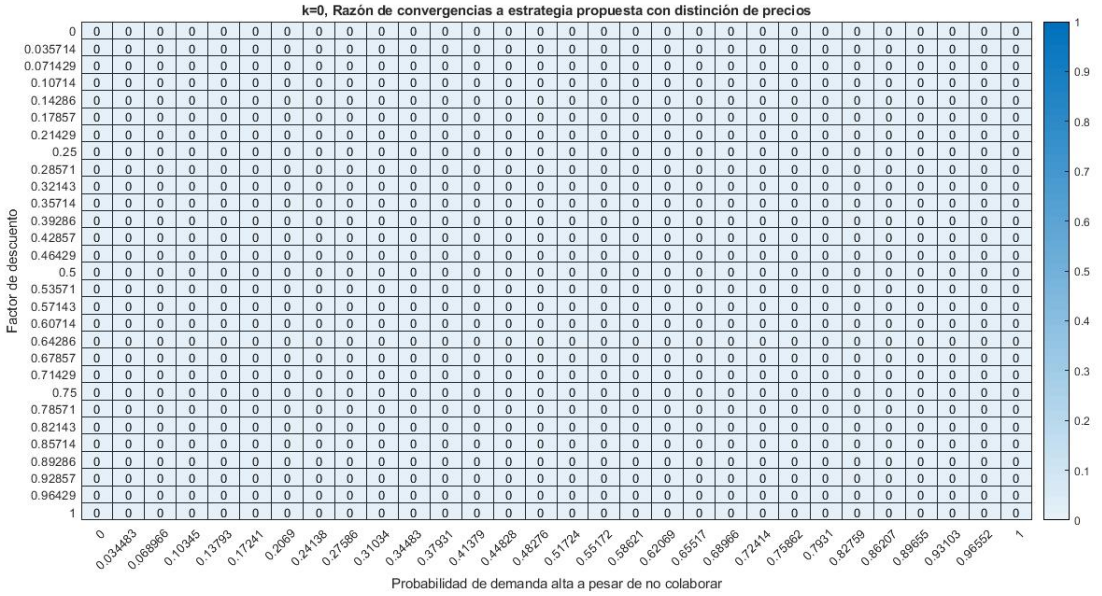


Figura F.23: Tasa de convergencia a la estrategia propuesta en ejemplo 1, $K = 0$

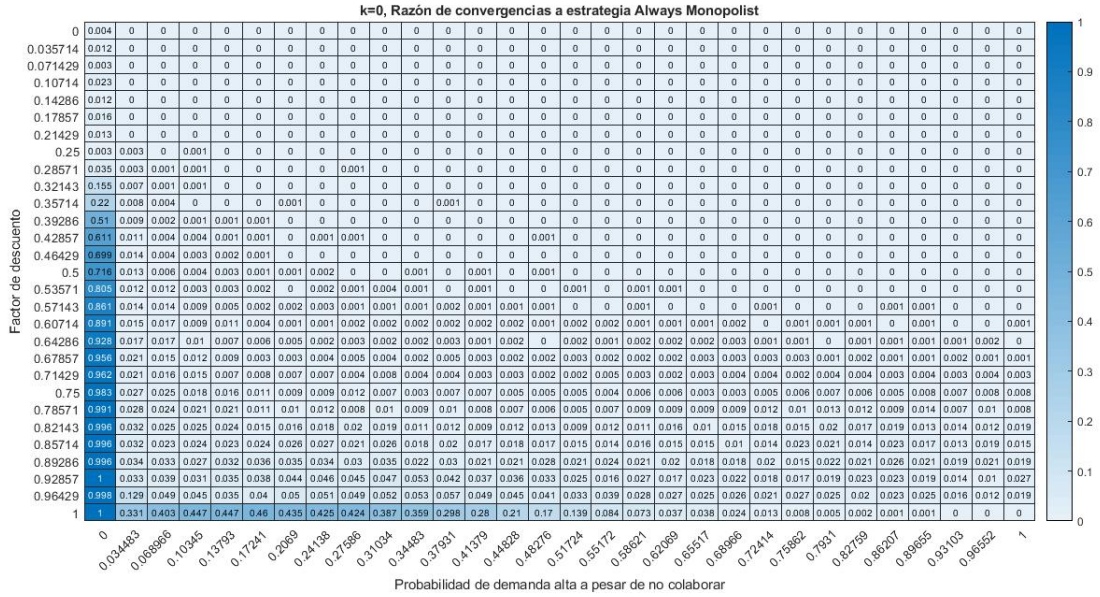


Figura F.24: Tasa de convergencia a estrategia always monopolist en ejemplo 1, $K = 0$

Anexo G

Código y uso

En esta sección se adjuntan los códigos utilizados. Se adjunta un vinculo de una revisión para facilitar su uso[1] y para acceder a las versiones descargables[2] en formato .m. Partes redundantes han sido excluidas para disminuir la longitud del anexo así como los códigos de las experiencias 1,2 y 3, estos se encuentran disponibles en el enlace para eventuales revisiones. [3]

Código G.1: Dilema del prisionero k=1

```

k=1                         % Matlab R2017b
clc                         % Limpiamos consola
clear                       % Limpiamos espacio de trabajo

cd 'C:\Users\Hbley\OneDrive\Documentos\Tesis\Codigo\DilemaDelPrisionero\CondDeman(K=0)\Nueva version'
rng(1111)          ;

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Variables que se pueden cambiar a gusto

nvx     = 30       ;           % La cantidad de elementos en la grilla (nvx*nvy) incrementar este valor solo
nvy     = 29       ;           % incrementa la cantidad total de juegos (el unico costo es tiempo el cual es lineal a (nvx*nvy))

parj    = 1000     ;           % Para cada elemento de la grilla de 30*29 se hacen parj*dim2*otra simulaciones, cada una consta de t iteraciones
otra    = 1        ;           % Incrementar la variable otra solo incrementa el tiempo multiplicandolo por su valor

t       = 50000    ;           % Cantidad de iteraciones por simulacion
P       = 4        ;           % Partidor de matrices, debe ser divisor de t. Reduce la memoria requerida, pero incrementa la demora.
b       = 1000     ;           % Cantidad de veces que se tiene que repetir un perfil en el ultimo periodo para declararlo convergente
cep     = 1        ;           % Cortar exploración? 0 no, 1 si
bd      = 10    + b ;           % Cantidad de veces que no se explora con certeza al final de la exploracion
ver     = 1        ;           % Verificar equilibrio y obtener Q-loss? 0 no, 1 si
exc     = 0        ;           % Exploración constante? 0 no (ie, exp con decaimiento exponencial), 1 si

XvarMIN= 0.025      ;
XvarMAX= 0.250      ;
YvarMIN = 0         ;
YvarMAX = 2*10^(-4) ;
Xnom    = 'Alpha'   ;
Ynom    = 'Beta'    ;

 % Cambiar estas variables implica cambiar el código
Nj  = 2            ;           % Cantidad de Jugadas posibles  (Cambiarla aun implica cambiar el calculo de la variable Pago)
JUS = 2            ;           % Cantidad de Jugadores         (Cambiarla implica cambiar el calculo de pagos)

 %%%%%%%%%%  %%%%%%%%%%

tP  = t/P          ;

XvarV = ((0 : 1 : nvx - 1)  /(nvx-1))' .* (XvarMAX  - XvarMIN )+ XvarMIN;    % Vector de alphas
YvarV = ((0 : 1 : nvy - 1)  /(nvy-1))' .* (YvarMAX  - YvarMIN )+ YvarMIN;    % Vector de betas

Result= zeros(7,length(XvarV),length(YvarV))    ;                        % Almacenador de resultados finales

```

[1] https://youtu.be/dVIx3g-HZWA.

[2] Los códigos utilizados en la tesis se pueden encontrar en https://1drv.ms/u/s!AhB7Nqt5qiCKg5gIhI7_ftE6NIGwPA?e=Wvdpcv.

[3] Adicionalmente los papers citados se encuentran aqui: https://1drv.ms/u/s!AhB7Nqt5qiCKg508P7qc7OJZ3zMSAg?e=kwNnGx.

```
le  =   length(XvarV)                       ;
lk  =   length(YvarV)                       ;

parfor ii= 1 : 1 : le*lk                                    % Comienzan iteraciones paralelas
    %En estas lineas hacemos las llamadas al indice ii (para evitar comunicaciones innecesarias)
   tic

   [indx1,indx2] = ind2sub([le lk], ii);    % Pasamos de un indice lineal del parfor a un indice como la gente

   epsii   =    -0.5       ;              % Valor de perdida de desvio asociado al juego
   kksii   =       2       ;              % Valor de ganancia por desvio asociada al juego
   a   =   XvarV(indx1)    ;
   beT =   YvarV(indx2)    ;
   peC =   0.05            ;              % Probabilidad de explorar en caso de exploración constante

   delta   = 0.9           ;              % Factor de descuento intertemporal
   mon     = 1./((1-delta));              % Calculamos el valor monopolico descontado (A mano)
   ecn     = 0./((1-delta));              % Valor de utilidad descontada equilibrio de nash

   for tt=1:1:otra
       %Declarammos estas variables vacias para evitar problemas de clasificación de parfor

      I       = zeros(1,parj,JUS,1) ;      % Matriz de indices de elecciones
      II      = zeros(1,parj,JUS,1) ;      % Matriz de indices de elecciones post exploración
      aaa     = zeros(1,parj,JUS,Nj);      % Indicatriz de si se jugó la jugada por el jugador
      aar     = zeros(1,parj,JUS,Nj);      % Indicatriz del rival de aaa
      aa1     = zeros(1,parj,JUS,Nj);      % Indicatriz de si el jugador 1 juega (x < Nj)
      aa2     = zeros(1,parj,JUS,Nj);      % Indicatriz del jugador 2
      aa1h    = zeros(1,parj,JUS,Nj);      %  aa1 del periodo anterior (k=1)
      aa2h    = zeros(1,parj,JUS,Nj);

      Pago    = zeros(1,parj,JUS,1) ;      % Matriz de utilidades dado II
      PagoEf  = zeros(1,parj,JUS)   ;      % Matriz de utilidades percibidas (pagadas)
      ddd     = zeros(1,parj)       ;      % Matriz Contador de equilibrio en subjuego perfecto
      eee     = zeros(1,parj)       ;
      fff     = zeros(1,parj)       ;      % Matriz Contador de Q-loss

      Qef     = zeros(1,parj,JUS)   ;      % Se registra el valor descontado efectivo
      Resul   = zeros(7,le,lk)      ;      %
      ult     = 0                   ;

       % Variables que crecen con k
      Q       = zeros(1,parj,JUS,Nj,Nj,Nj)                ;    % Matriz Q (al incrementar k se debe agregar ",Nj,Nj")
      matriz  = zeros(1,parj,JUS,1 ,Nj,Nj)                ;    % Matriz transitoria, para eleccionar la celda de la matriz Q relativa a las jugadas de cada jugador en parj,JUS

       % Comienzan las iteraciones en P porciones de tamaño tP
      for p = 1 : 1 : P
          % Seteamos la exploración para esta porción de iteraciones
          T    = ((p-1)*tP + 1) : 1 : (p)*(tP)                ;    % Vector de T según cada bloque
          Azar =  rand(tP,parj,2) <    exc.*peC.*ones(tP,parj,2) +...     ;    % Indicatriz de si se prefiere explorar
                        +(1-exc).*repmat(exp(-beT .*T'),[1 parj 2]) ;    % exploración de decaimiento exponencial

          if  (p)*(tP) > t-bd-1 && cep == 1
              if ult==0
                  Azar((t-bd-1 )- (p-1)*(tP):end ,:,:)    = 0          ;    % No se explora en las ultimas iteraciones
                  ult=1                                                ;
              else
                  Azar(:,:,:) = 0                                      ;
              end
          end

          Elij    =   randi(Nj,tP,parj,2)                          ;

          for i = ((p-1)*tP + 1) : 1 : (p)*(tP)

              in  =   i-((p-1)*tP )                                ;    % Indice i ajustado para calzar en las matrices truncadas de la porcion p

              [M,I] = max(sum(sum(Q.*matriz,6),5),[],4);

              II(1,:,:)     = I.*(1 - Azar(in,:,:)) + Elij(in,:,:).*Azar(in,:,:);      % Precio jugado considerando post exploracion

              for xhx = 1:1:Nj
                  aaa(1,:,:,xhx) = (II(1,:,:)==xhx);                   % Indicatriz si el jugador JJ jugó en 1 la jugada xhx
              end
              aar(1,:,1,:) = aaa(1,:,2,:);    aar(1,:,2,:) = aaa(1,:,1,:);           % Indicatriz del rival a JJ


              Pago(1,:,:) = aaa(1,:,:,1).*aar(1,:,:,1) + aar(1,:,:,2).*aaa(1,:,:,1).*epsii + aar(1,:,:,1).*aaa(1,:,:,2).*(kksii);
              PagoEf(1,:,:) = 0;

              aa1(1,:,:,:)        =   repmat(aaa(1,:,1,:),1,1,2,1);
              aa2(1,:,:,:)        =   repmat(aaa(1,:,2,:),1,1,2,1);

               % Actualizamos matriz de decision Q

              matriz =                    reshape(  aa1(1,:,:,:),[1,parj,JUS,1,Nj,1])...
                                       .*reshape(  aa2(1,:,:,:),[1,parj,JUS,1,1,Nj]);
```

```
143
144
145
146
147
148
149
150          M2= max(sum(sum(Q.*matriz,6),5),[],4);
151
152          Matrix  =   sum(matriz(1,:,:,1,:),5) > 0;
153
154          for inn = 1 : 1 : Nj ^ 2
155             [da1,da2]                          = ind2sub([Nj Nj], inn);
156
157             Q(1,:,1,da1,:,:)                   = Q(1,:,1,da1,:,:) + matriz(1,:,1,:,:,:).*aaa(1,:,1,da1).*aaa(1,:,2,da2).*(-a.*Q(1,:,1,da1,:,:) + a.*(Pago(1,:,1)+delta*M2(1,:,1,:)));  % Jugadores actualizan pago en
       ↪ matriz
158             Q(1,:,2,da2,:,:)                   = Q(1,:,2,da2,:,:) + matriz(1,:,2,:,:,:).*aaa(1,:,1,da1).*aaa(1,:,2,da2).*(-a.*Q(1,:,2,da2,:,:) + a.*(Pago(1,:,2)+delta*M2(1,:,2,:)));  % Jugadores actualizan pago en
       ↪ matriz
159             PagoEf(1,:,:)                      = PagoEf(1,:,:)    + Matriz(1,:,:,:)    .*aaa(1,:,:,da1).*aar(1,:,:,da2).*Pago(1,:,:);
160          end
161
162          % Actualizamos matriz Qef (utilidad descontada percibida)
163          Qef(1,:,:) = PagoEf(1,:,:) + delta * Qef(1,:,:);
164          if i == t-b
165             [M,IFh]     =   max(Q(1,:,:,:,:,:),[],4)    ;
166          end
167          % Verificamos convergencias
168          if i > t-b
169
170             [M,IF]  =   max(Q(1,:,:,:,:,:),[],4)                    ;
171
172             eee(:,:) = eee(:,:) + prod((IFh(1,:,1,:)==IF(1,:,1,:)).*(IFh(1,:,2,:)==IF(1,:,2,:)),4);
173
174             % Aqui verificamos si se resuelve el problema de programación dinámica para k > 0
175
176             if i == t && ver == 1                          % Se resuelve si es resolviendo el problema de programación dinánica.
177                v       =   zeros(1,parj,JUS,1)            ;   % Valores de continuación de cada estado
178                VV      =   zeros(1,parj,JUS,2,Nj,Nj)      ;   % Valores de continuacion dada cada jugada
179                ER      =   flip(IF,3)                     ;   % Fijamos estrategia del rival
180
181                % barremos cada uno de los juegos hechos en esta instancia y obtenemos el valor de continuación
182                for idm = 1:1:parj
183
184                   Vc1a11 = sym('Vc1a11')                                  ; %
185                   Vc1a12 = sym('Vc1a12')                                  ; %
186                   Vc1a21 = sym('Vc1a21')                                  ; %
187                   Vc1a22 = sym('Vc1a22')                                  ; %
188
189                   Vc2a11 = sym('Vc2a11')                                  ; %
190                   Vc2a12 = sym('Vc2a12')                                  ; %
191                   Vc2a21 = sym('Vc2a21')                                  ; %
192                   Vc2a22 = sym('Vc2a22')                                  ; %
193
194                   eq1a11 = Vc1a11 ==(IF(1,idm,1,1,1,1)==1).*(ER(1,idm,1,1,1,1)==1).*(  1   + delta.*Vc1a11)...
195                               + (IF(1,idm,1,1,1,1)==1).*(ER(1,idm,1,1,1,1)==2).*(epsii + delta.*Vc1a12)...
196                               + (IF(1,idm,1,1,1,1)==2).*(ER(1,idm,1,1,1,1)==1).*(kksii + delta.*Vc1a21)...
197                               + (IF(1,idm,1,1,1,1)==2).*(ER(1,idm,1,1,1,1)==2).*(  0   + delta.*Vc1a22);
198
199                   eq1a12 = Vc1a12 ==(IF(1,idm,1,1,1,2)==1).*(ER(1,idm,1,1,1,2)==1).*(  1   + delta.*Vc1a11)...
200                               + (IF(1,idm,1,1,1,2)==1).*(ER(1,idm,1,1,1,2)==2).*(epsii + delta.*Vc1a12)...
201                               + (IF(1,idm,1,1,1,2)==2).*(ER(1,idm,1,1,1,2)==1).*(kksii + delta.*Vc1a21)...
202                               + (IF(1,idm,1,1,1,2)==2).*(ER(1,idm,1,1,1,2)==2).*(  0   + delta.*Vc1a22);
203
204                   eq1a21 = Vc1a21 ==(IF(1,idm,1,1,2,1)==1).*(ER(1,idm,1,1,2,1)==1).*(  1   + delta.*Vc1a11)...
205                               + (IF(1,idm,1,1,2,1)==1).*(ER(1,idm,1,1,2,1)==2).*(epsii + delta.*Vc1a12)...
206                               + (IF(1,idm,1,1,2,1)==2).*(ER(1,idm,1,1,2,1)==1).*(kksii + delta.*Vc1a21)...
207                               + (IF(1,idm,1,1,2,1)==2).*(ER(1,idm,1,1,2,1)==2).*(  0   + delta.*Vc1a22);
208
209                   eq1a22 = Vc1a22 ==(IF(1,idm,1,1,2,2)==1).*(ER(1,idm,1,1,2,2)==1).*(  1   + delta.*Vc1a11)...
210                               + (IF(1,idm,1,1,2,2)==1).*(ER(1,idm,1,1,2,2)==2).*(epsii + delta.*Vc1a12)...
211                               + (IF(1,idm,1,1,2,2)==2).*(ER(1,idm,1,1,2,2)==1).*(kksii + delta.*Vc1a21)...
212                               + (IF(1,idm,1,1,2,2)==2).*(ER(1,idm,1,1,2,2)==2).*(  0   + delta.*Vc1a22);
213
214
215                   eq2a11 = Vc2a11 ==(IF(1,idm,2,1,1,1)==1).*(ER(1,idm,2,1,1,1)==1).*(  1   + delta.*Vc2a11)...
216                               + (IF(1,idm,2,1,1,1)==1).*(ER(1,idm,2,1,1,1)==2).*(epsii + delta.*Vc2a21)...
217                               + (IF(1,idm,2,1,1,1)==2).*(ER(1,idm,2,1,1,1)==1).*(kksii + delta.*Vc2a12)...
218                               + (IF(1,idm,2,1,1,1)==2).*(ER(1,idm,2,1,1,1)==2).*(  0   + delta.*Vc2a22);
219
220                   eq2a12 = Vc2a12 ==(IF(1,idm,2,1,1,2)==1).*(ER(1,idm,2,1,1,2)==1).*(  1   + delta.*Vc2a11)...
221                               + (IF(1,idm,2,1,1,2)==1).*(ER(1,idm,2,1,1,2)==2).*(epsii + delta.*Vc2a21)...
222                               + (IF(1,idm,2,1,1,2)==2).*(ER(1,idm,2,1,1,2)==1).*(kksii + delta.*Vc2a12)...
223                               + (IF(1,idm,2,1,1,2)==2).*(ER(1,idm,2,1,1,2)==2).*(  0   + delta.*Vc2a22);
224
225                   eq2a21 = Vc2a21 ==(IF(1,idm,2,1,2,1)==1).*(ER(1,idm,2,1,2,1)==1).*(  1   + delta.*Vc2a11)...
226                               + (IF(1,idm,2,1,2,1)==1).*(ER(1,idm,2,1,2,1)==2).*(epsii + delta.*Vc2a21)...
227                               + (IF(1,idm,2,1,2,1)==2).*(ER(1,idm,2,1,2,1)==1).*(kksii + delta.*Vc2a12)...
228                               + (IF(1,idm,2,1,2,1)==2).*(ER(1,idm,2,1,2,1)==2).*(  0   + delta.*Vc2a22);
229
230                   eq2a22 = Vc2a22 ==(IF(1,idm,2,1,2,2)==1).*(ER(1,idm,2,1,2,2)==1).*(  1   + delta.*Vc2a11)...
231                               + (IF(1,idm,2,1,2,2)==1).*(ER(1,idm,2,1,2,2)==2).*(epsii + delta.*Vc2a21)...
232                               + (IF(1,idm,2,1,2,2)==2).*(ER(1,idm,2,1,2,2)==1).*(kksii + delta.*Vc2a12)...
233                               + (IF(1,idm,2,1,2,2)==2).*(ER(1,idm,2,1,2,2)==2).*(  0   + delta.*Vc2a22);
234
235                   [Sc1a11 Sc1a12 Sc1a21 Sc1a22] = vpasolve(eq1a11, eq1a12, eq1a21, eq1a22,[Vc1a11 Vc1a12 Vc1a21 Vc1a22])  ;
236                   [Sc2a11 Sc2a12 Sc2a21 Sc2a22] = vpasolve(eq2a11, eq2a12, eq2a21, eq2a22,[Vc2a11 Vc2a12 Vc2a21 Vc2a22])  ;
```

```
                    v(1,idm,1,1,1,1)=double(Sc1a11);v(1,idm,1,1,1,2)=double(Sc1a12);
                    v(1,idm,1,1,2,1)=double(Sc1a21);v(1,idm,1,1,2,2)=double(Sc1a22);

                    v(1,idm,2,1,1,1)=double(Sc2a11);v(1,idm,2,1,1,2)=double(Sc2a12);
                    v(1,idm,2,1,2,1)=double(Sc2a21);v(1,idm,2,1,2,2)=double(Sc2a22);

                end

                VV(1,:,1,1,:,:)=  (ER(1,:,1,1,:,:)==1).*(  1  +  delta.*v(1,:,1,1,1,1)) + (ER(1,:,1,1,:,:)==2).*(epsii + delta.*v(1,:,1,1,1,2));
                VV(1,:,2,1,:,:)=  (ER(1,:,2,1,:,:)==1).*(  1  +  delta.*v(1,:,2,1,1,1)) + (ER(1,:,2,1,:,:)==2).*(epsii + delta.*v(1,:,2,1,2,1));

                VV(1,:,1,2,:,:)=  (ER(1,:,1,1,:,:)==1).*(kksii + delta.*v(1,:,1,1,2,1)) + (ER(1,:,1,1,:,:)==2).*(  0  +  delta.*v(1,:,1,1,2,2));
                VV(1,:,2,2,:,:)=  (ER(1,:,2,1,:,:)==1).*(kksii + delta.*v(1,:,2,1,2,1)) + (ER(1,:,2,1,:,:)==2).*(  0  +  delta.*v(1,:,2,1,2,2));

                [MI,vI]     = max(VV(1,:,:,:,:,:),[],4)            ;   % Vemos si habria desvio en un periodo
                ddd=prod(IF(1,:,:)== vI(1,:,:),3)                  ;
                fff=sum(abs(M(1,:,:) - MI(1,:,:)),3)./JUS          ;

              end

            end

            % Guardamos historia para la proxima iteracion




            aa1h(1,:,:,:)  =  aa1(1,:,:,:);
            aa2h(1,:,:,:)  =  aa2(1,:,:,:);
          end
        end
        % Contador de estrategias


        % Guardamos los resultados acumulados de esta simulación
        Resul(4,indx1,indx2) = sum(fff,2)       ;
        Resul(3,indx1,indx2) = sum(ddd,2)       ;
        Resul(2,indx1,indx2) = sum(eee==b,2)    ;
        Resul(1,indx1,indx2) = (squeeze(sum(Qef(1,:)/JUS,2))-ecn)/(mon-ecn);
        Result  = Result + Resul./(parj*otra)   ;
    end
    toc                                             % Se termina de medir el tiempo
end

% Borramos variables inutiles y guardamos
clearvars -except Result XvarV YvarV Xnom Ynom k t                  % Borramos todas las varialbes EXCEPTO estas
save('K' + string(k) + "t" + string(t) + XvarV + Ynom)

% Result=round(Result*1000)/1000;

% Razón de utilidad                                         % 1 monopolica, 0 bertrand
Dk1HRU = heatmap(string(XvarV),string(YvarV),squeeze(Result(1,:,:))');
Dk1HRU.Title = 'K='+string(k)+', Razón de la utilidad promedio / valor presente descontado de cooperar';
Dk1HRU.XLabel = Xnom;
Dk1HRU.YLabel = Ynom;
Dk1HRU.ColorLimits = [0 1];

% Convergencias a estrategias                               % Cantidad de veces que se convergió a algún perfil
Dk1HCE = heatmap(string(XvarV),string(YvarV),squeeze(Result(2,:,:))');
Dk1HCE.Title = 'K='+string(k)+', Razón de convergencias a estrategias';
Dk1HCE.XLabel = Xnom;
Dk1HCE.YLabel = Ynom;
Dk1HCE.ColorLimits = [0 1];

% Eps                                                       % Cantidad de veces que se llega a equilibrio
Dk1eps = heatmap(string(XvarV),string(YvarV),squeeze(Result(3,:,:))');
Dk1eps.Title = 'K='+string(k)+', Razón de convergencias a equilibrio en sub juego';
Dk1eps.XLabel = Xnom;
Dk1eps.YLabel = Ynom;
Dk1eps.ColorLimits = [0 1];

% Q loss                                                    % Diferencia entre el Q obtenido y l calculado
Dk1HQL = heatmap(string(XvarV),string(YvarV),squeeze(Result(4,:,:))');
Dk1HQL.Title = 'K='+string(k)+', Error promedio / valor presente descontado de cooperar';
Dk1HQL.XLabel = Xnom;
Dk1HQL.YLabel = Ynom;
```