Universidad de Granada

Department of Computer Science and Artificial Intelligence

Doctoral Program in Computer Science and Information Technology

Adversarial decision and optimization-based models

Doctoral Thesis

Pablo José Villacorta Iglesias

Advisors:

David Alejandro Pelta

José Luis Verdegay Galdeano

La memoria titulada **"Adversarial decision and optimization-based models"**, que presenta D. Pablo José Villacorta Iglesias para optar al grado de Doctor en Informática, ha sido realizada en el Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada, bajo la dirección de los doctores D. David Alejandro Pelta y D. José Luis Verdegay Galdeano, miembros del mismo departamento.

Hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, Noviembre de 2015

Pablo José Villacorta Iglesias

David Alejandro Pelta

José Luis Verdegay Galdeano

-A mi familia-

(To my family)

Contents

Resumen **vii**

Abstract **xix**

1 Introduction **1**

1.1 Objectives and structure of the thesis 4

2 Background **9**

2.1 Game theory 10

2.2 Security games 13

2.3 Search games 15

2.4 Imitation games 16

2.4.1 An adversarial imitation model 16

2.4.2 On the unsuitability of game-theoretic equilibrium . . 19

2.4.3 Strategies of the agents 21

3 Theoretical analysis of expected payoff with interpretable strategies **23**

3.1 Calculation of the expected payoff 24

3.1.1 Notation and payoff matrix definition 24

3.1.2 Expected payoff of R-k-B 25

3.1.3 Expected payoff of Proportionally Random 27

3.1.4 Expected payoff of B-k-R 29

3.2 Computational experiments and results 32

3.2.1 Results for Proportionally Random 33

3.2.2 Results for R-k-B 36

3.2.3 Results for B-k-R 38
3.3 Conclusions 40

4 Forgetting as a way to avoid deception 41
4.1 Simplified version of the model 42
4.2 Behaviour of the agents 43
4.2.1 Static Mixed Strategy for Agent S. 44
4.2.2 Dynamic Mixed Strategy for Agent S 45
4.2.3 A generalized notation for the expected payoff 49
4.3 Forgetting as a way to avoid deception 51
4.3.1 Expected payoff with limited memory 52
4.4 Experiments and results 54
4.4.1 Experimental settings 54
4.4.2 Results 56
4.5 Conclusions 68

5 Considering statistical dependence between actions and events 71
5.1 A model with statistical dependence 72
5.2 Behaviour of the agents 75
5.2.1 Static mixed strategy for S under statistical dependence 75
5.2.2 Dynamic mixed strategy for S under statistical dependence 78
5.3 Experiments and results 84
5.3.1 Static vs dynamic strategies: performance comparison 91
5.3.2 On the influence of the number of periods 92
5.4 Conclusions 94

6 Estimation of Fuzzy Stationary Probabilities of a Markovian patrolling strategy 97
6.1 Markov chains 98
6.1.1 Related work 100
6.2 A method to compute fuzzy stationary probabilities 101
6.2.1 Fuzzy numbers 102

6.2.2 Fuzzy transition probabilities from observations 103

6.2.3 Fuzzy Markov chains and restricted matrix multiplication 104

6.2.4 User-specified fuzzy transition probabilities 105

6.2.5 Computation of fuzzy stationary probabilities 106

6.3 The FuzzyStatProb package 110

6.3.1 Implementation issues 112

6.4 Application example 116

6.4.1 Departing from a sequence of observations 118

6.4.2 Departing from user-specified fuzzy transition probabilities 124

6.4.3 On the reduction of uncertainty with more observations 126

6.5 Conclusions 131

7 A patrolling model for a UAV protecting an area against terrestrial intruders 133

7.1 Problem statement 134

7.2 State of the art 135

7.3 Methodology 137

7.3.1 Game-theoretic Formulation 137

7.3.2 Mathematical Representation of Trajectories to Compute Capture Probabilities 145

7.4 Towards an approximate solving strategy 148

7.4.1 Components of a population-based metaheuristic . . . 150

7.5 Conclusions 152

8 Conclusions and Future Work 159

8.1 Conclusions 159

8.2 Future Work 161

References 163

Agradecimientos

En estos tres años y medio he tenido buenos y malos momentos (afortunadamente, muchos más de los primeros que de los segundos) que han marcado un período de mi vida muy gratificante. Está claro que no podría haber alcanzado esta meta en solitario. Por lo tanto, quiero dedicar unas líneas a todos aquellos que han contribuido a que esta tesis finalmente vea la luz.

Mis primeras palabras de agradecimiento son para mis directores, Curro y David, por darme la oportunidad de incorporarme al grupo de Modelos de Decisión y Optimización (MODO), y por guiarme con paciencia y con comentarios clarificadores. Gracias a ellos aprendí, poco a poco, que en este viaje que supone una tesis doctoral, era yo quien debía encontrar por mí mismo el camino para convertirme en un verdadero investigador, y no en un estudiante obediente. La libertad que me han dado durante estos años para elegir los temas, enfoques y métodos que quería utilizar ha contribuido a desarrollar mis habilidades y, sin duda, tendrá un fuerte impacto en mi carrera.

También debo expresar mi agradecimiento a otros miembros del grupo MODO que me han acompañado durante este trayecto y que han influido de manera importante en mi vida académica. Quiero dar las gracias a la Prof. María Teresa Lamata (Maite), Directora del grupo, por su espíritu constructivo y por dedicarme siempre buenas palabras y grandes dosis de sentido común. En segundo lugar, al Dr. Carlos Cruz, por su apoyo y ánimo incondicionales, y por brindarme una visión de la investigación y, sobre todo, de la vida que sólo una persona de su experiencia en ambas puede tener. En tercer lugar, al Dr. Antonio Masegosa, con quien he compartido muchas horas de interesantes discusiones y logros, por su inspiradora capacidad de trabajo y por su disponibilidad para ayudarme con cualquier tema en cualquier momento. Finalmente, a todos los miembros actuales y pasados del grupo MODO, con quienes he compartido reuniones, viajes, congresos y cafés: Juanra, Nacho, Elio, Blanca, Virgilio y Pavel.

He de mencionar al Ministerio de Educación que ha financiado parcialmente este doctorado a través de una beca FPU que también posibilitó mi estancia de investigación en los Estados Unidos. Asimismo, estoy profundamente agradecido al Prof. Milind Tambe, de la University of Southern California en Los Ángeles, por aceptarme durante tres meses en su grupo, el Teamcore Research Group on Agents and Multi-agent Systems, y a todos los miembros del mismo por acogerme tan amistosamente, donarme su tiempo sin tener por qué, y por contribuir a crear una atmósfera que me hizo sentir casi como en casa. No son solamente excelentes investigadores sino grandes personas.

Desde el punto de vista personal, tengo la suerte de contar con buenos amigos que me han distraído lo suficiente como para concentrarme mejor en la vuelta al trabajo tras los momentos de desconexión que he compartido con ellos. Cuando vuelvo la vista atrás y me doy cuenta de que algunos de ellos me han acompañado durante quince o incluso veinte años de mi vida, me siento todavía más afortunado.

Finalmente, esta tesis está dedicada a toda mi familia. A mis padres, por su sacrificio y por proporcionarme siempre lo mejor; a todos mis tíos y tías, especialmente a Juan y Natalia por quererme como a un nieto; a mis primos, por su paciencia con su primillo más pequeño. Y por supuesto, a mi prima y madrina, la Dra. Natalia Navas, que me mencionó en los agradecimientos de su tesis, allá por el año 1996. En aquel entonces, ninguno de los dos imaginábamos que casi veinte años después publicaríamos juntos.

Esta tesis ha sido financiada parcialmente por los proyectos P07-TIC-02970 y P11-TIC-8001 de la Junta de Andalucía, TIN2011-27696-C02-01 del Ministerio de Economía y Competitividad, GENIL-PYR-2014-9 del CEI-BioTIC (Universidad de Granada), TIN2008-01948 y TIN2008-06872-C04-04 del Ministerio de Ciencia e Innovación, y por la beca FPU referencia AP-2010-4738 del Ministerio de Educación.

Resumen

Este resumen contiene una versión en español de los capítulos 1 (introducción) y 8 (conclusiones), y ha sido incluido para cumplir con la normativa de tesis doctorales de la Universidad de Granada.

Introducción

La toma de decisiones nos rodea. Todo el mundo lleva a cabo elecciones a diario, desde el mismo momento en que nos despertamos por la mañana: levantarse al instante o permanecer un rato en la cama; qué comida desayunar, coger el autobús o ir a pie al trabajo, etc. Por supuesto, las consecuencias de las decisiones que acabamos de mencionar no son demasiado importantes, más allá de ganar algunos kilos si tomamos un desayuno poco saludable, o llegar tarde al trabajo si el autobús que elegimos tomar se retrasa. Los factores que tenemos en cuenta para estas decisiones son bastante simples, dado que las consecuencias de cada elección están bien definidas. Sin embargo, en el mundo de los negocios, los directivos toman decisiones que tienen importantes consecuencias en el futuro de su propia empresa (en términos de beneficios, posicionamiento en el mercado o políticas de empresa) que a su vez influyen en la organización de la compañía y repercuten en los empleados. El número de factores que un director general debe tener en cuenta es grande y desgraciadamente, con frecuencia son difíciles de medir o incluso de darse cuenta de su existencia.

La toma de decisiones es, por tanto, una actividad inherentemente humana. En la actualidad existe un interés creciente en incorporar capacidades propias de los humanos a los sistemas informáticos. Los *Smart systems* [7] pueden

definirse como sistemas autónomos o claborativos que aúnan sensores, actuadores, informática y comunicaciones para ayudar a los usuarios o a otros sistemas a realizar una tarea. Por su propia naturaleza estos sistemas combinan funcionalidades. No sólo unen múltiples tecnologías sino que están fuertemente orientados a las aplicaciones concretas, en sectores como transporte, energía, salud, seguridad, manufacturación, etc.

Están compuestos de sensores para adquisición de señal, elementos transmisores de información a una unidad central que toma decisiones inteligentes basándose en la información disponible, y actuadores que llevan a cabo las acciones adecuadas según la decisión. La importancia de los Smart systems queda demostrada por su inclusión en los objetivos del programa Horizonte 2020 de la Unión Europea, en la convocatoria ICT-2 (Integración de Smart Systems [1]). Una de sus principales características es precisamente la toma de decisiones, que utiliza la información suministrada por los sensores para intentar tomar la mejor decisión, tal como haría una persona. La pregunta es, ¿cómo se determina la mejor decisión?

En el caso (ideal) de que un decisor (a) sea capaz de tener en cuenta perfectamente todas las consecuencias que puedan derivarse de una decisión (decisor perfectamente informado) y pueda cuantificarlas, (b) esté al tanto de todas las posibles alternativas para escoger, y (c) sea capaz de realizar todos los cálculos necesarios de una manera perfectamente racional, entonces la elección óptima será la acción cuya consecuencia sea la preferida del decisor por encima de todas las demás consecuencias asociadas a las demás acciones. Aún podrían discutirse algunos detalles de esta definición, como la manera de definir las preferencias, de cuantificarlas o si todos estos elementos serían percibidos y cuantificados de la misma forma por otros decisores que se encontrasen ante la misma situación.

Además, la mayoría de las situaciones que requieren de un proceso riguroso de toma de decisiones presentan incertidumbre en muchos aspectos: una acción puede tener asociadas diferentes consecuencias que pueden darse o no con cierta probabilidad. La naturaleza probabilística de este proceso dio lugar a la disciplina conocida como *Análisis de Decisión*, que estudia desde una perspectiva formal cómo la gente debería tomar las decisiones [81].

Los problemas estudiados por el Análisis de Decisión consideran que sólo hay un decisor involucrado en la decisión. Sin embargo, tomar una decisión cuando sabemos que existe alguien que observa y reacciona a nuestra elección constituye una situación bastante diferente al caso en el que no existe ningún adversario. Por ejemplo, podríamos preferir ocasionalmente una alternativa sub-óptima orientada a causar confusión en el adversario, de manera que sea más difícil para él predecir nuestras propias decisiones en el futuro. Esto llevaría a un mayor beneficio para nosotros a largo plazo.

Consideremos el siguiente ejemplo. Un hombre de negocios va caminando a su trabajo todas las mañanas. Se trata de un hombre adinerado que posee una gran compañía con importantes beneficios anuales, así que es un potencial objetivo para un grupo terrorista que necesite financiación. Todos los días se marcha a las 8 de la mañana y sigue el mismo camino hasta su trabajo, no demasiado lejos. Sabe que en este caso, caminar es más rápido que ir en coche, y que su ruta es la más corta, por lo que puede afirmarse que su elección es la mejor para él. Sin embargo, si el hombre supiera que está siendo vigilado por alguien que intenta aprender sus costumbres, entonces debería cambiar deliberadamente su comportamiento para hacerlo más impredecible. Por ejemplo, podría tomar caminos alternativos ocasionalmente, o ir en coche o en autobús de vez en cuando, sin ningún patrón definido, a pesar de que todas estas elecciones son aparentemente peores dado que se gasta más tiempo en ellas. A pesar de todo, el resultado es que será más difícil secuestrarle en un día cualquiera, lo cual es sin duda la consecuencia que nuestro hombre de negocios prefiere.

De este ejempo se deduce que la manera en la que tomamos decisiones es diferente cuando sabemos que existe un adversario cuyas acciones afectan a nuestros beneficios. En su forma más general, este tema es conocido como decisión en presencia de adversarios y constituye el centro de la presente tesis. El razonamiento en presencia de adversarios trata en gran medida sobre la comprensión de la mente y las acciones de nuestro oponente. Es de interés para una gran variedad de problemas donde los actores son conscientes de la existencia del otro, y saben que están compitiendo contra otro cuyos objetivos son generalmente contrarios. El objetivo de este tipo de análisis es encontrar estrategias óptimas que tengan en cuenta no sólo nuestras propias preferencias sino también las creencias y preferencias de los oponentes según

nosotros las percibimos.

Esta situación se da en muchos campos de la vida real, especialmente (aunque no únicamente) en la lucha antiterrorista y en la prevención del crimen [49, 70]. Ciertamente, la amenaza del terrorismo ha incrementado la inversión y el interés en el desarrollo de técnicas y herramientas computacionales para el razonamiento en presencia de adversarios, sobre todo para seguridad nacional [43]. Pueden encontrarse también aplicaciones menos dramáticas en los juegos de ordenador donde el usuario juega el papel de adversario frente a los personajes manejados por el ordenador, provistos de capacidades inteligentes de razonamiento con el fin de mejorar la calidad, la dificultad y la capacidad adaptación del juego, las cuales unidas proporcionan una mejor experiencia para el jugador. Otra utilidad similar a la anterior es el desarrollo de sistemas inteligentes de entrenamiento de personas. Este tipo de tecnologías persiguen la predicción de las estrategias del oponente, la anticipación a sus planes, y el reconocimiento de engaños. Los ámbitos de aplicación incluyen también los negocios, transacciones, conflictos entre gobiernos, finanzas, deportes de equipo (por ejemplo RoboCup), juegos como el póker, etc [48].

En términos simples, un adversario es una entidad cuyos objetivos son opuestos a los nuestros, y que es capaz de influenciar las ganancias que obtenemos de nuestras decisiones eligiendo y ejecutando sus propias acciones. Este tipo de interacción competitiva entre dos actores ocurre en gran cantidad de situaciones y puede analizarse con un abanico variado de técnicas [48]. Muchas están íntimamente ligadas a la Inteligencia Artificial (modelado basado en agentes, heurísticas, planificación, exploración de árboles, aprendizaje automático) pero también a la Ingeniería del Conocimiento (cómo representar y manejar el conocimiento sobre el adversario y el entorno) y la Investigación Operativa, con énfasis en la Teoría de Juegos. Varios problemas pueden formalizarse como juegos competitivos y resolverse utilizando la Teoría de Juegos o técnicas relacionadas. Revisaremos algunos de ellos, como los juegos de búsqueda, los juegos de seguridad y los problemas de patrullaje en el capítulo de Background.

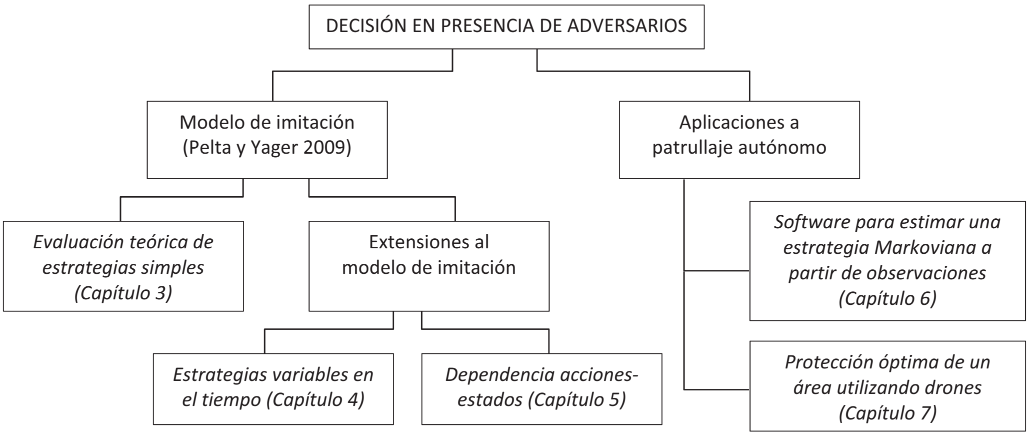

DECISIÓN EN PRESENCIA DE ADVERSARIOS
Modelo de imitación (Pelta y Yager 2009)
Aplicaciones a patrullaje autónomo
Evaluación teórica de estrategias simples (Capítulo 3)
Extensiones al modelo de imitación
Software para estimar una estrategia Markoviana a partir de observaciones (Capítulo 6)
Estrategias variables en el tiempo (Capítulo 4)
Dependencia acciones-estados (Capítulo 5)
Protección óptima de un área utilizando drones (Capítulo 7)

Figure 1: Estructura de la tesis.

Objetivos y estructura de la tesis

Teniendo en cuenta todo esto, el **objetivo general** de esta tesis es el análisis y diseño de modelos de decisión y optimización en presencia de adversarios que sean capaces de modelar situaciones como las descritas anteriormente. Vamos a llevar a cabo estudios teóricos y a proponer aplicaciones prácticas relacionadas con los juegos de imitación, los juegos de seguridad y los modelos de patrullaje.

Los **objetivos específicos** son los siguientes:

1. Recopilar las referencias relevantes en el campo de los modelos de decisión con adversarios y sus aplicaciones.

2. Desarrollar nuevos modelos y extensiones.

3. Proponer nuevas estrategias para estos modelos que tengan en cuenta la componente temporal.

4. Evaluar las estrategias desde un punto de vista teórico y empírico.

5. Proponer ejemplos de aplicación.

Para lograr estos objetivos, la tesis se ha organizado como muestra la Figura 1. Hemos abordado la decisión en presencia de adversarios desde dos puntos de vista diferentes. En primer lugar, nos hemos centrado en un modelo abstracto de imitación publicado en [67]. La situación consiste en un agente S que entra en repetidas ocasiones en una decisión de conflicto con un agente T, que es capaz de observar el comportamiento de S, aprender de él y utilizar lo aprendido para tomar decisiones que pueden cambiar (posiblemente disminuir) la ganancia de S con el fin de incrementar la suya propia. El trabajo citado estudia el balance entre el grado de confusión introducido mediante decisiones sub-óptimas y los beneficios que se pueden obtener de ellas a largo plazo. Los autores concluyen que las estrategias aleatorizadas son útiles para esto. Revisamos el modelo en el capítulo 2 junto con otros conceptos necesarios para entender el resto de la tesis y algunos problemas relacionados.

Hemos publicado tres artículos que tratan sobre este modelo o extensiones al mismo, en conexión con los objetivos 2, 3 y 4. En consecuencia, hay tres capítulos dedicados al modelo de imitación. El capítulo 3, que contiene el material publicado en [92], explica cómo obtener una expresión teórica de la ganancia del agente S cuando utiliza estrategias simples de decisión que habían sido propuestas en [67]. La expresión es validada mediante simulaciones computacionales, como indican los objetivos 3 y 4. Los capítulos 4 y 5 también se centran en la obtención de una expresión matemática del pago en dos extensiones diferentes del modelo de imitación, en consonancia con los objetivos 2 a 4. El material de estos capítulos está publicado en [95] y [94], respectivamente. Más concretamente, el capítulo 4 se centra en una versión simplificada del modelo y estudia el caso en el que el agente S utiliza una distribución de probabilidad optimizada sobre sus acciones. Después, se demuestra que cambiar en ciertos instantes la distribución de probabilidad empleada, tras una serie de encuentros con el adversario, puede ser beneficioso. Por su parte, el capítulo 5 extiende el modelo original introduciendo dependencia estadística entre la decisión tomada por S en el momento actual y el conjunto de pagos que podrán obtenerse como resultado de las decisiones del siguiente instante. De modo similar, se calculan expresiones matemáticas del pago de estrategias de decisión optimizadas para esta nueva situación.

En segundo lugar, hemos desarrollado dos trabajos de aplicación de la decisión en presencia de adversarios. El primero de ellos [96], correspondiente al capítulo 6, explica cómo mejorar la información obtenida a partir de la observación de un agente del cual se sabe que se encuentra en una situación frente a un adversario. Dicha situación es el patrullaje de un área para protegerla frente a intrusiones. Presentamos para ello un procedimiento matemático y un paquete software que lo implementa, siguiendo los objetivos 4 y 5. El segundo de ellos, aún no publicado (capítulo 7), describe un nuevo modelo de patrullaje para un vehículo aéreo no tripulado que vuela sobre un área industral para protegerla de intrusos terrestres. A diferencia del primero, este modelo está concebido específicamente para un agente que va volando sobre el terreno sin limitaciones de movimiento. Propondremos una solución basada en teoría de juegos bajo el marco general de los juegos de seguridad. Por tanto, aquí hacemos referencia a los objetivos 2 a 5. Parte de este trabajo se completó durante la estancia breve de investigación realizada por el autor en el Teamcore Research Group on Agents and Multi-agent Systems, Computer Science Department, University of Southern California, Los Ángeles (EEUU).

Finalmente, el capítulo 8 está dedicado a conclusiones generales, así como a algunas posibles líneas futuras que pueden investigarse en relación con los diferentes modelos descritos a lo largo de la tesis.

Los resultados obtenidos durante el desarrollo de esta tesis han dado lugar a un conjunto de publicaciones que se detalla a continuación.

Capítulo 3:

P. Villacorta y D. Pelta (2010). Evolutionary design and statistical assessment of strategies in an adversarial domain. Proc. of the IEEE Conf. on Evolutionary Computation, pp. 2250 - 2256.

P.J. Villacorta y D.A. Pelta (2012). Theoretical Analysis of Expected Payoff in an Adversarial Domain. **Information Sciences** 186(1): 93-104.

Capítulo 4:

P. Villacorta y D. Pelta (2011). Expected payoff analysis of dynamic mixed strategies in an adversarial domain. Proc. of the IEEE Symposium on Intelligent Agents. IEEE Symposium Series on Computational Intelligence, 116 - 122.

P.J. Villacorta, D.A. Pelta y M.T. Lamata (2013). Forgetting as a way to avoid deception in a repeated imitation game. **Autonomous Agents and Multi-Agent Systems** 27(3): 329-354.

Capítulo 5:

P.J. Villacorta, L. Quesada y D. Pelta (2012). Automatic Design of Deterministic Sequences of Decisions for a Repeated Imitation Game with Action-State Dependency. Proc. of the IEEE Conf. on Computational Intelligence y Games, 1 - 8.

P.J. Villacorta y D.A. Pelta (2015). A repeated imitation model with dependence between stages: decision strategies y rewards. **Int. Journal of Applied Mathematics and Computer Science** 25 (3): 617 - 630.

Capítulo 6:

P.J. Villacorta y J.L. Verdegay (2015). FuzzyStatProb: an R Package for the Estimation of Fuzzy Stationary Probabilities from a Sequence of Observations of an Unknown Markov Chain. **Journal of Statistical Software**. En prensa.

Conclusiones y trabajos futuros

Conclusiones

Esta tesis se ha centrado en el estudio y desarrollo de modelos de decisión en presencia de adversarios y su aplicación a problemas prácticos. Los objetivos eran los siguientes:

1. Recopiar las referencias relevantes en el campo de los modelos de decisión con adversarios y sus aplicaciones.
2. Desarrollar nuevos modelos y extensiones.
3. Proponer nuevas estrategias para estos modelos que tengan en cuenta la componente temporal.
4. Evaluar las estrategias desde un punto de vista teórico y empírico.
5. Proponer ejemplos de aplicación.

Con respecto al objetivo 1, el capítulo de background contiene una revisión de las referencias importantes en varios campos, con énfasis en las que guardan mayor relación con los modelos que hemos propuesto.

Con respecto al objetivo 2, se ha logrado siguiendo los dos puntos de vista mencionados en la introducción:

- Por un lado, hemos continuado investigando sobre el modelo de imitación [67], para el cual hemos desarrollado dos extensiones. En la primera de ellas, hemos mostrado la importancia de contar con un modelo apropiado del adversario, así como el impacto negativo que tiene sobre el pago de un agente hacer suposiciones erróneas acerca del adversario. Esta extensión considera explícitamente la componente temporal de una estrategia con el fin de cambiar la aleatorización que guía el movimiento de uno de los agentes. Mientras que esto incrementa el pago total cuando el adversario se ajusta a la concepción que tenemos de él, también puede suponer importantes pérdidas cuando esto no ocurre. En la segunda extensión, hemos introducido dependencia estadística entre la decisión actual tomada por uno de los agentes y las circunstancias (modeladas como el conjunto de pagos que se podrán obtener) de la siguiente decisión que ese agente tenga que tomar. Hemos mostrado que el mismo concepto de estrategias aleatorizadas que varían en el tiempo también resulta ventajoso en este nuevo escenario.

- Por otro lado, hemos desarrollado un modelo con adversarios para una situación real en el contexto de los juegos de seguridad: un dron que protege un área frente a intrusos. La propuesta toma en consideración de manera realista las limitaciones físicas de los agentes, tales como la duración de la batería del dron y su velocidad, la existencia de grafos separados de movimiento para el dron y para los intrusos terrestres, el radio de percepción limitado del dron (que depende de la cámara a bordo), etc. Hemos explicado las dificultades del cálculo del equilibrio de Stackelberg en nuestro modelo debido a que el problema de optimización que resulta, pese a ser lineal, es intratable en la práctica debido al alto número de estrategias existentes. Se ha sugerido la utilización de técnicas heurísticas para remediarlo.

En lo que respecta al objetivo 3, se han investigado nuevas estrategias que no sólo son aleatorizadas sino que cambian la aleatorización a lo largo del tiempo, lo cual se ha demostrado beneficioso para llevar a engaño al adversario y conseguir un mayor pago. Esto confirma que es posible la manipulación estratégica cuando la única información disponible para el adversario consiste en observaciones sobre las acciones pasadas. Por lo tanto, cuando un agente se sabe observado, es ventajoso para él modificar la manera en que toma las decisiones para causar engaño y obtener un mayor pago a largo plazo, tal y como se ha logrado con las estrategias que varían en el tiempo.

En todas las publicaciones hemos analizado el rendimiento de las estrategias tanto desde un punto de vista teórico como empírico, con el fin de validar las expresiones analíticas que se habían deducido, tal y como nos habíamos marcado en el objetivo 4. Esto ha demostrado también que no es necesario llevar a cabo simulaciones para comparar dos estrategias aleatorizadas si podemos obtener correctamente el pago esperado que se obtendrá con dicha estrategia aplicando conceptos de la teoría de la probabilidad. Comparar el pago esperado es una manera mucho más fiable de determinar qué estrategia funciona mejor.

Finalmente, en relación al objetivo 5, hemos propuesto dos aplicaciones: por un lado, el modelo de patrullaje para un dron mencionado anteriormente, y por otro, una nueva técnica matemática para mejorar la información obtenida mediante las observaciones de un agente que está patrullando un área siguiendo una estrategia aleatorizada Markoviana. Estas observaciones constituyen la percepción que el intruso tiene acerca del defensor cuando el primero observa al segundo con el objetivo de aprender su aleatorización, tal como se suele asumir en los juegos de seguridad. Hemos mostrado que utilizando conceptos de Lógica Difusa y Números Difusos, el intruso puede tener una mejor aproximación de las verdaderas probabilidades que guían el movimiento del agente que patrulla, ya que un número difuso es más informativo porque incorpora la incertidumbre que rodea a una cantidad. Hemos implementado un paquete software de código abierto con el método propuesto y lo hemos incorporado a un repositorio público para que esté a disposición de la comunidad investigadora.

Trabajos futuros

Tanto el juego de imitación como los modelos de patrullaje aún precisan de mayor investigación. En el modelo de imitación, no se han investigado algunas técnicas que podrían dar buenos resultados, tales como el uso de mecanismos de aprendizaje más sofisticados para el agente T (por ejemplo, aprendizaje por refuerzo) que le facilitarían aprender la estrategia de S. Además, el juego de imitación podría abordarse como un problema de clasificación para el que se pueden aplicar algoritmos de aprendizaje automático para determinar cuál de entre las posibles alternativas va a ser elegida a continuación por el agente S. Por último, deben aún examinarse aplicaciones reales de este modelo, especialmente orientadas al ámbito de los juegos de ordenador. Una mejor predicción de las acciones del usuario llevará a adversarios más inteligentes que mejorarán la experiencia del juego por parte del usuario.

Con respecto al modelo de patrullaje para el drone, aún quedan varias cuestiones abiertas. La dificultad matemática del problema de optimización que debe resolverse para calcular el equilibrio de Stackelberg abre la puerta a la aplicación de técnicas de optimización aproximada, como algoritmos bioinspirados u otras metaheurísticas, que aún no han sido aplicadas a juegos de seguridad. La magnitud de los espacios de acciones de ambos jugadores requieren que se considere solamente un subconjunto de estrategias porque el problema completo no puede resolverse debido a las limitaciones físicas sobre la memoria RAM disponible. Este problema se da con frecuencia en los juegos de seguridad cuando se intentan escalar para escenarios reales. Aún deben realizarse experimentos para comprobar qué metaheurísticas en concreto resuelven mejor este tipo de problemas y cómo adaptarlas al dominio de los juegos de seguridad. Dado el creciente número de problemas que están siendo abordados como juegos de seguridad en la actualidad, si el uso de metaheurísticas se demuestra efectivo para el problema del dron, puede llevar a su adopción en otros modelos que presentan problemas similares de escalabilidad.

Por último, no se ha modelado la incertidumbre sobre las limitaciones físicas de los agentes, como la velocidad real del dron y la velocidad que asumimos para los posibles intrusos, o la duración exacta de la batería. Ninguno de estos parámetros puede ser conocido con exactitud, por lo que parece natural modelarlos como números difusos, y recurrir a enfoques existentes de programación lineal difusa para calcular el equilibrio de Stackelberg. Esto no se ha hecho hasta ahora en juegos de seguridad.

Abstract

Decision making is all around us. Everyone makes choices everyday, from the moment we open our eyes in the morning. Some of them do not have very important consequences in our life and these consequences are easy to take into account. However, in the business world, managers make decisions that have important consequences on the future of their own firm (in terms of revenues, market position, business policy) and their employees. In these cases, it is difficult to account for all the possible alternatives and consequences and to quantify them. Decision making tools such as Decision Analysis are required in order to determine the optimal decision.

Furthermore, when several competing agents are involved in a decision making situation and their combination of actions affect each other's revenues, the problem becomes even more complicated. The way an agent makes a decision and the tools required to determine the optimal decision change. When we are aware of someone observing and reacting to our behavior, one might occasionally prefer a sub-optimal choice aimed at causing confusion on the adversary, so that it will be more difficult for him to guess our decision in future encounters, which may report us a larger benefit. This situation arises in counter-terrorist combat, terrorism prevention, military domains, homeland security, computer games, intelligent training systems, economic adversarial domains, and more.

In simple terms, we define an adversary as an entity whose aims are somehow inversely related to ours, and who may influence the profits we obtain from our decisions by taking his/her own actions. This kind of competitive interaction between two agents fits a variety of complex situations which can be analyzed with a number of techniques ranging from Knowledge Engineering and Artificial Intelligence (agent-based modeling, tree exploration, machine learning) to Operational Research, with an emphasis on Game Theory.

The objective of this thesis is the analysis and design of adversarial decision and optimization-based models which are able to represent adversarial situations. We are going to conduct theoretical studies and propose practical applications including imitation games, security games and patrolling domains. More precisely, we first study a two-agent imitation game in which one of the agents does not know the motivation of the other, and tries to predict his decisions when repeatedly engaging in a conflict situation, by observing and annotating the past decisions. We propose randomized strategies for the agents and study their performance from a theoretical and empirical point of view. Several variants of this situation are analyzed. Then we move on to practical applications. We address the problem of extracting more useful information from observations of a randomized Markovian strategy that arises when solving a patrolling model, and propose a mathematical procedure based on fuzzy sets and fuzzy numbers for which we provide a ready-to-use implementation in an R package. Finally, we develop an application of adversarial reasoning to the problem of patrolling an area using an autonomous aerial vehicle to protect it against terrestrial intruders, and solve it using a mix of game-theoretic techniques and metaheuristics.

Chapter 1

Introduction

Decision making is all around us. Everyone makes choices everyday, from the moment we open our eyes in the morning: whether to get up at once or stay in bed for a while, what to have for breakfast, whether to take the bus or walk to work, and so on. Of course, the consequences of the aforementioned decisions are not very important in our life beyond the fact that we might put on some weight if our daily breakfast is unhealthy, or arrive late to work if the bus is delayed. The factors we take into account for these decisions are quite simple, since the consequences of each choice are well defined. However, in the business world, managers make decisions that have important consequences on the future of their own firm (in terms of revenues, market position, business policy) which have a direct influence on the organization of the company and thus, on their employees. The number of factors that a CEO must take into account is large and, unfortunately, it is often difficult to quantify them or even to realize about their existence.

Decision making is, thus, an inherently human ability. Nowadays, there is a growing interest on incorporating human-like capabilities to automated systems. Smart systems [7] can be defined as autonomous or collaborative systems which bring together sensing, actuation, informatics and communications to help users or other systems perform a role. By their very nature these systems combine functionalities. They may extract multiple functionalities from a common set of parts, materials, or structures. Smart Systems not only unite multiple technologies, they are strongly tailored to application sectors such as transport, energy, healthcare, security and safety, manufacturing, and so on.

They are composed of sensors for signal acquisition, elements to transmit the information to a central unit which makes intelligent decisions based on the available information, and actuators that perform the adequate actions according to the decision. The importance of Smart systems is proved by their inclusion in the objectives of the Horizon 2020 programme of the European Union, in the call ICT-2 (Smart Systems Integration [1]). One of the main features of these systems is their decision making module, which accounts for the information provided by the sensors and tries to make the best decision, just as a person would do. A question arises here: how to determine the "best" decision?

In the (ideal) case that a decision maker (a) is able to perfectly account for all the consequences that may arise from a decision (i.e. he is fully informed) and measure them, (b) is aware of all his alternatives, and (c) is able to make all the necessary calculations in a perfectly rational way, then the optimal choice would be the action whose consequence is preferred by the decision maker among all the other consequences associated to the rest of actions. Some details could still be discussed in this definition, such as how to define the preferences, how they should be measured and whether this concept is perceived equally by all the decision makers. Moreover, most situations that require a serious decision making process pose uncertainty in many ways: an action may have different consequences that may or may not arise with some probability. The probabilistic nature of this process gave birth to the discipline known as *Decision Analysis*, which studies from a formal perspective how people *ought to* make decisions [81].

The problems addressed by Decision Analysis consider that only one decision maker is involved in the decision. However, making a decision when we are aware of someone observing and reacting to our behavior is quite different from the case when no adversary exists. For instance, one would occasionally prefer a sub-optimal choice aimed at causing confusion on the adversary, so that it will be more difficult for him to guess our decision in future encounters. This in turn leads to a larger benefit in the long term.

Consider the following example. A business man walks to work every morning. He owns a well-established company with large revenues per year so he is a potential target for a terrorist group in need of funding. He always leaves home at 8 a.m. and follows the same path to his office, which is not very far. He knows that walking is faster than taking the car, and that his route is the shortest, so we can say it is the optimal choice for him. However, if he is aware that he is under surveillance by someone trying to learn his habits, he should deliberately change his behaviour to become more unpredictable and keep the adversary guessing. For instance, he might occasionally try alternative (longer) paths, or go by car or by bus from time to time (with no pattern), even though these choices are apparently worse in the short term as they take more time. However, as a result, it will be more difficult to kidnap him on an arbitrary day, which is the preferred consequence.

From this example, it follows that the way we make decisions differs when we are aware of an adversary whose actions can affect our benefits. In its most general meaning, this topic is known as adversarial decision making and it constitutes the focus of the present thesis. Adversarial reasoning is largely about understanding the mind and actions of one's opponent. It is relevant to a broad range of problems where the actors are aware of each other, and they know they are contesting at least some of the other's objectives. The goal of this kind of analysis is to find optimal strategies taking into account not only one's preferences but also the beliefs and preferences of the opponents as perceived by oneself, which does not always match the true adversarial preferences.

This situation arises in many areas of real life, with particular (but not exclusive) interest in counter-terrorist combat and crime prevention [49, 70]. Certainly, the threat of terrorism has fueled the investments and interest in the development of computational tools and techniques for adversarial reasoning, mostly oriented to homeland defense [43]. Horizon 2020 includes a topic called *Safe societies* funded with 1.7M euros within the area of Social Challenges, and states that one of its objectives is to "fight against crime, traffic and terrorism, including understanding and fighting against its underlying ideas and beliefs". We can find less dramatic applications as well in fields like computer games where the user is the adversary and the computer characters are provided with adversarial reasoning features to enhance the quality, difficulty and adaptivity of the game, which together improve the gaming experience. Another orientation of this is the development of intelligent training systems. Nowadays there is a quest for technologies aiming at opponent strategy prediction; plan recognition; deception discovery and planning; among others, and which apply not only to security or computer games but also to business, transactions, government vs government conflicts, economic adversarial domains, team sports (e.g., RoboCup), competitions (e.g., Poker), etc. [48].

In simple terms, we define an adversary as an entity whose aims are somehow inversely related to ours, and who may influence the profits we obtain from our decisions by taking his/her own actions. This kind of competitive interaction between two agents fits a variety of situations which can be analyzed with a number of techniques [48]. They are closely related to Artificial Intelligence (agent-based modeling, heuristics, planning, tree exploration, machine learning) but also to Knowledge Engineering (how to capture and represent the knowledge about the adversary and the environment) and Operational Research, with emphasis in Game Theory. Several adversarial problems can be formalized as a competitive game and solved by Game Theory or related approaches. We will review some of them, namely search games, security games and optimal autonomous patrolling, in the Background chapter.

1.1 Objectives and structure of the thesis

With this in mind, the **general objective** of this thesis is the analysis and design of adversarial decision and optimization-based models which are able to represent situations like the one described above. We are going to conduct theoretical studies and propose practical applications including imitation games, security games and patrolling domains. The **specific objectives** are the following:

1. To compile the relevant references in the field of adversarial models and applications.

2. To develop new models and extensions.

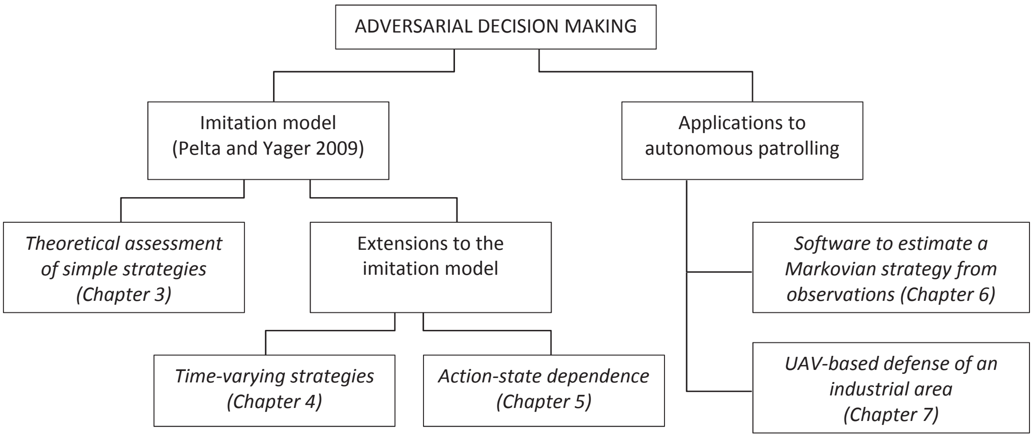

ADVERSARIAL DECISION MAKING
Imitation model (Pelta and Yager 2009)
Applications to autonomous patrolling
Theoretical assessment of simple strategies (Chapter 3)
Extensions to the imitation model
Software to estimate a Markovian strategy from observations (Chapter 6)
Time-varying strategies (Chapter 4)
Action-state dependence (Chapter 5)
UAV-based defense of an industrial area (Chapter 7)

Figure 1.1: Structure of the thesis.

3. To propose new strategies for the models, explicitly including a temporal component.

4. To assess the strategies from a theoretical and empirical point of view.

5. To propose examples of application.

In order to accomplish these objectives, the thesis has been organized as shown in Figure 1.1. We have tackled adversarial decision making from two different points of view. Firstly, we have focused on an abstract imitation model published in [67]. The situation consists of an agent S who repeatedly engages in a conflicting situation with an adversary T, who is able to observe our behavior, learn from it and use it to make decisions that may change (possibly decrease) our gains in order to increase his/her own benefits. This work addresses the balance between the degree of confusion induced with suboptimal decisions and the benefits they may have in the long term. The authors conclude that randomized strategies are beneficial for this purpose. This model is reviewed in chapter 2 along with the main concepts needed to understand the rest of the thesis and related problems.

We have published three articles dealing with the aforementioned model and extensions, in connection with objectives 2, 3 and 4. Accordingly, there are three chapters devoted to the imitation model. Chapter 3, which contains the material published in [92], explains how to obtain a theoretical expression of the gains of agent S when using simple decision strategies in the model. The expression is then validated with computational simulations, addressing objectives 3 and 4. Chapters 4 and 5 also focus on the computation of a mathematical expression of the payoff on two different extensions of the imitation model, addressing objectives 2 to 4. The material in these chapters has been published in [95] and [94], respectively. More precisely, chapter 4 focuses on a simplified version of the model and studies the case when agent S uses an optimized probability distribution over its actions. Then it demonstrates that changing the probability distribution along the time, after a number of encounters against T, can be beneficial. Chapter 5 extends the original model by introducing statistical dependence between the current decision made by S and the set of payoffs attainable at the next encounter, and provides mathematical expressions of the payoff of optimized decision strategies for this new situation.

Secondly, we have developed two works about concrete applications of adversarial decision making. The first of them [96], corresponding to Chapter 6, explains how to improve the information obtained from the observation of an agent who is known to be engaged in an adversarial situation, namely patrolling an area to prevent intrusions. A mathematical procedure and a software package which implements it are presented, addressing objectives 4 and 5. The second, not published yet (chapter 7), describes a novel adversarial patrolling model for an Unmanned Aerial Vehicle (UAV) flying over an industrial area to protect it against terrestrial intruders. Differently from the first, this framework was specifically conceived for a flying agent with no movement restrictions. We present a game-theoretic solution to the model in the setting of Security Games. Therefore, here we address objectives 2 to 5. The proposal combines a Mixed Integer Linear Program with metaheuristics that make the problem solvable. This work has been partially done during a short research stay done by the candidate within the Teamcore Research Group on Agents and Multi-agent Systems, Computer Science Department, University of Southern California, Los Angeles (USA).

Finally, chapter 8 is devoted to the general conclusions of the thesis, and outlines some future research lines that could be investigated in connection with the different models described in this work.

The results obtained along the development of this thesis have been published in the articles mentioned below.

Chapter 3:

P. Villacorta and D. Pelta (2010). Evolutionary design and statistical assessment of strategies in an adversarial domain. Proc. of the IEEE Conf. on Evolutionary Computation, pp. 2250 - 2256.

P.J. Villacorta and D.A. Pelta (2012). Theoretical Analysis of Expected Payoff in an Adversarial Domain. **Information Sciences** 186(1): 93-104.

Chapter 4:

P. Villacorta and D. Pelta (2011). Expected payoff analysis of dynamic mixed strategies in an adversarial domain. Proc. of the IEEE Symposium on Intelligent Agents. IEEE Symposium Series on Computational Intelligence, 116 - 122.

P.J. Villacorta, D.A. Pelta and M.T. Lamata (2013). Forgetting as a way to avoid deception in a repeated imitation game. **Autonomous Agents and Multi-Agent Systems** 27(3): 329-354.

Chapter 5:

P.J. Villacorta, L. Quesada and D. Pelta (2012). Automatic Design of Deterministic Sequences of Decisions for a Repeated Imitation Game with Action-State Dependency. Proc. of the IEEE Conf. on Computational Intelligence and Games, 1 - 8.

P.J. Villacorta and D.A. Pelta (2015). A repeated imitation model with dependence between stages: decision strategies and rewards. **Int. Journal of Applied Mathematics and Computer Science** 25 (3): 617 - 630.

Chapter 6:

P.J. Villacorta and J.L. Verdegay (2015). FuzzyStatProb: an R Package for the Estimation of Fuzzy Stationary Probabilities from a Sequence of Observations of an Unknown Markov Chain. **Journal of Statistical Software**. In press.

Chapter 2

Background

We have explained in the introductory chapter that, in single-agent real world's decisions, it is very difficult to account for all the possible alternatives and consequences of a decision, and even when they can be successfully captured, an optimal decision process requires the application of mathematical abstractions (provided by Decision Analysis) that are not simple. We also stated that adversarial decision making requires a different way of making decisions. When an adversary is aggregated to the situation, everything becomes much more complicated. The untidy nature of the problem remains, but we also have to consider how the adversary's decisions may affect our benefits. In other words, we need to predict how the adversary will decide, and adapt our decision accordingly. In order to do this prediction, we have to represent and manage the adversary's beliefs, intentions, and so on; we want, in one word, to construct a *model* of the adversary.

Many computational solutions exist to determine or anticipate the state, intent and actions of one's adversary in an environment where one strives to effectively counter the adversary's actions. Topics such as belief and intent recognition, opponent strategy prediction, plan recognition, deception discovery, deception planning and strategy generations must be considered. One of the disciplines that contributes to adversarial decision making is Game Theory. It is a mature field that is traditionally focused on the rigorous study of problems in which two or more actors strive to achieve their respective goals while interacting in a certain domain.

However, many real problems studied by adversarial reasoning are far less tidy and well-defined than those studied by Game Theory (for instance, chess). As stated in [48],

> The set of possible inputs can be very large and ill-defined, the dynamics might not be completely known or can contain random elements. Furthermore, in contrast to chess, one seldom knows the true state of the system; in fact, the players typically have only imperfect observations of some portion of the domain in which the game is being played. Thus, a tremendous gap exists between problems that can be rigorously solved and the real-world problems that need to be solved.
>
> To fill this immense gap, many tools outside of the traditional realm of game theory must be brought to bear. As this volume demonstrates, practical adversarial reasoning calls for a broad range of disciplines, including but no limited to stochastic processes, artificial intelligence planning, cognitive modeling, robotics and agent theory, robust control theory, and machine learning.

In this chapter, we provide some basics of Game Theory and describe some of its applications to real adversarial decision making situations as they are closer to the concrete models we have studied in the thesis. Of course, as the fragment above states, many other adversarial scenarios exist in real-world military domains, military planning, domestic security, law enforcement and antiterrorism that require the application of tools different than Game Theory, but they are out of the scope of this dissertation. The interested reader may refer to [48] for further detail.

2.1 Game theory

In this section we introduce some basic definitions of Game Theory that are going to be used along the thesis, especially in chapter 7. As stated in [64],

> Game theory is a bag of analytical tools designed to help us understand the phenomena that we observe when decision-makers

> interact. The basic assumptions that underlie the theory are that decision-makers pursue well-defined exogenous objectives (they are rational) and take into account their knowledge or expectations of other decision-makers' behavior (they reason strategically).

A way to synthesize this is that a game is a decision problem (which means we need to compute the optimal decision) in which there is more than one decision-maker (called players) and each player's actions affect the others [74]. It started in a formal way in 1944 with the book of von Neumann and Morgenstern [98]. The concepts used in the next chapters are more closely related to strategic-form games (also called normal-form games) so we will define these. A strategic game is a model of interactive decision-making in which each decision-maker chooses his plan of action once and for all, and these choices are made simultaneously. The model consists of (see [64]):

- A finite set N of players.
- For each player $i \in N$, a non-empty set A_i of actions available to that player[1]. Let $A = \times_{j \in N} A_j$. We will refer to an element $a \in A$ as an action profile, $a = (a_j)_{j \in N}$, also called an outcome of the game.
- For each player $i \in N$, a utility function $u_i : A \to \mathbb{R}$. If $u_i(a) \geqslant u_i(b)$ we say that player i prefers outcome a to b. The real values given by this function are called the payoffs or utilities of the game.

With these elements, a game can be formalized as a tuple $\langle N, (A_i), (u_i)_i \rangle$. We show an example of a two-player strategic-form game in Fig. 2.1. Player 1 has an action set $A_1 = \{T, B\}$ while player 2 has $A_2 = \{L, R\}$. The numbers inside the cells are the payoffs each player receives when the outcome of the game matches that cell. When player 1 chooses T and player 2 chooses L, the outcome of the game is (T, L) so player 1 gets a payoff of w_1 and player 2 gets w_2. Formally, we can say that $u_1(T, L) = w_1$ and $u_2(T, L) = w_2$.

The solution of a game prescribes the optimal choice for each player and is called an *equilibrium*. Several types of equilibria can be defined depending

[1] In our models we always consider A discrete and finite, but in general this is not necessarily the case.

	L	R
T	w_1, w_2	x_1, x_2
B	y_1, y_2	z_1, z_2

Table 2.1: A two-player strategic-form game.

on the circumstances of the game, but the most common type is known as Nash equilibrium. A *Nash equilibrium* [64] of a strategic game $\langle N; (A_i); (u_i)_i \rangle$ is a profile $a^* \in A$ of actions with the property that for every player $i \in N$ we have

$$u_i(a^*_{-i}, a^*_i) \geqslant u_i(a^*_{-i}, a_i) \qquad \forall a_i \in A_i$$

Thus for a^* to be a Nash equilibrium it must be that no player i has an action yielding an outcome that he prefers to that generated when he chooses a^*_i, given that every other player j chooses his equilibrium action a^*_j. In other words, no player can profitably deviate on his own if the other players abide to the Nash equilibrium.

The existence of a Nash equilibrium cannot be guaranteed for any game. However, there is a kind of Nash equilibrium that always exists. First, let us define the set $\Delta(A_i)$ as the set of probability distributions over the elements of A_i. An element $\sigma \in \Delta(A_i)$ is called a *mixed strategy* [64] of player i, and it is a probability distribution over the actions available to that player. An element of A_i (i.e. an action) is also known as a *pure strategy* as opposed to the former. With this, the probability of a given action profile or outcome of the game, $(a_j)_{j \in N}$, when the players use mixed strategies can be computed as $\prod_{j \in N} \alpha_j(a_j)$, where $\alpha_j(\cdot)$ is the probability that player i gives to an action in his mixed strategy. This can be done because the randomizations of the players are independent of each other.

If the players use mixed strategies, the payoff that a player considers is called the expected payoff for that player. If we call α to the set of the randomizations applied by all players, then the expected payoff for player i when the players use α can be defined as the sum of the payoff attained by player i for each feasible action profile, weighted by the probability that the action profile actually arises:

$$U_i(\alpha) = \sum_{a \in A} \left(\prod_{j \in N} \alpha_j(a_j) \right) u_i(a)$$

Taking into account the aforementioned new definitions, a mixed extension to a strategic-form game [64] can be defined as a tuple $\langle N, \Delta(A_i), (U_i)_i \rangle$ in which the actions available to the players are actually mixed strategies, i.e. randomizations over pure strategies, and the utility functions turn into expected utility functions. Analogously, a mixed strategy Nash equilibrium is a Nash equilibrium of the mixed extension. It can be proved that every finite strategic-form game has a mixed strategy Nash equilibrium [64], which constitutes one of the most important results in Game Theory.

Note that for a player to compute the Nash equilibrium it is necessary to know the payoffs of the rest of the players. This fact motivates the non game-theoretic treatment we give to some models proposed along the thesis.

2.2 Security games

The use of AI techniques combined with Game Theory to address real world security problems has proven very useful as demonstrated by the applicability of the solutions found, resulting in sophisticated systems successfully deployed for instance in the United States, probably the country which devotes most resources to homeland security issues. The field called Security Games [84] has emerged from the application of Game Theory to security domains [29]. These problems deal with finding the best resource allocation of the defender in order to protect a number of targets from an adversary aimed at attacking them. The number of resources is limited so that not all the targets can be protected all the time. Each target has two numbers associated to it which stand for the benefits and/or losses when it is successfully attacked.

The aforementioned description can be easily modeled as a Stackelberg game, a game-theoretic model in which one of the players (in this case, the defender) acts as the leader and the other (the attacker) plays the role of the follower who first observes and learns the leader's strategy, and then attacks after careful planning. Stackelberg games were first introduced to model leadership and commitment [99], and are now widely used to study security problems, from robbers and police [11], missile defense systems [21], computer network security [53, 8], and terrorism [76]. The solution to the game is the Strong Stackelberg equilibrium (SSE) -also called leader-follower equilibrium-, which prescribes a randomized strategy for the leader and a deterministic strategy for the follower that are optimal for both in terms of the long-run payoff they provide.

Some successful examples of deployed systems based on Security Games are the ARMOR software for placing randomized checkpoints at Los Angeles International Airport [68], working since August 2007; the GUARDS software for protecting the whole US airport network at national scale [69]; the IRIS software for allocating special counter-terrorism personnel in US flights [44]; the PROTECT system to help patrolling the maritime infrastructure, currently in use in Boston and New York [79]; the TRUSTS system for fare inspection in transit systems [102]; and more recently, the PAWS system for protecting animal wildlife against poachers [101], deployed in Uganda's Queen Elizabeth National Park and a protected area in Malaysia. Some challenges posed by the preceding systems when they have to be deployed in real settings include the advances in scalable algorithms for computing the SSE in very large action spaces, the management of uncertainty about the adversary's abilities, the computation of robust solutions for non-fully rational adversaries, and the development of mathematical models of bounded rationality and their validation in experiments with human people.

Patrolling models. Some of the aforementioned works deal with patrolling problems. Patrolling games can be viewed as a subclass of Security games in which the role of the defender, played either by a human or a robot, moves along a perimeter or an area to protect it against intrusions. Originally, patrolling models were studied with other techniques different than Game Theory, although they were aimed at maximizing some protection or coverage metric, which ultimately requires solving a mathematical optimization problem. The first publications date back to 2003 and 2004. The first theoretical analysis of the patrolling problem in a multi-agent setting was given in [27], although this approach, like many others that followed, is non adversarial but frequency-based. A number of works dealing with border patrolling have been published by Kraus et al.; see [33, 5] and references therein. The authors propose randomized strategies for multiagent patrolling of a linear perimeter and analyze their theoretical properties. Like the former, such works are non-adversarial as they do not explicitly consider the adversary's preferences to attack some segments of the perimeter instead of some other; the first of these works is frequency-based, while the second is aimed at maximizing the *probability of penetration detection* (*ppd*). The authors have also used physical agents in multi-agent coordination tasks [4]. When patrolling models are addressed considering the payoffs for attacking each location, the situation usually becomes a Security Game, as in [11]. More details on recent patrolling works can be found in section 7.2.

2.3 Search games

A mathematical framework developed much earlier than security games is known as *Search Games*. In them, one player acts as the seeker and the other plays the role of the hider. The game can take place in a continuous space (either in a compact region, or in an unbounded area) or in a graph. The players move in the area (in some cases the hider is immobile) with no restrictions except for maximum speed constraints. Whenever the distance between them gets smaller than a detection radius, the hider is assumed to be captured and the game ends. The actions of the players are continuous trajectories within the search space, modeled as functions mapping a time instant to a spatial position, and there is a payoff associated to every pair of trajectories of the players. The seeker has no prior knowledge about the hider's starting position. The game is solved using game-theoretic and control theory concepts, mainly the minimax strategy.

The situation captured by a Search Game is more general than that of a Security Game. The mathematical theory was developed in the early seventies, as part of the intense military research that the United States were carrying at that time. Search games are first described in the last chapter of [41] and they are explained in detail in [9]. The main difference with security games is that in a Search game, we assume the existence of an intruder in the area being protected, so the patrollers' mission is to capture him in an optimal way (most often, within the shortest time).

A search game can also be implemented by a set of physical autonomous agents, as done in pursuit evasion works like [89], which uses both aerial and terrestrial autonomous vehicles. A lot of works have been published dealing with mathematical aspects of pursuit-evasion games and extensions, like those where the patrollers do not know their own environment and make a model of it while looking for the evader. Some are also solved using game-theoretical concepts [89]. Other variants of the game, namely defending a concrete target in a continuous environment, are solved with concepts from control theory [52]. A lot of simulation studies have been conducted for discrete variants and visibility-based versions; see [18] and references therein.

2.4 Imitation games

Imitation games have been studied from a formal perspective in works by McLennan and Tourky [57, 58, 59]. They are relevant for several reasons. Despite being a seemingly simple version of a non-cooperative two-personal game, they have proven as complex as a general game. It is shown in [58] that many problems related to the computation of an equilibrium in an imitation game are NP-complete, although [57] proves that a mixed-strategy Nash equilibrium always exists in one-shot imitation games. Recall that this computation requires that a player knows the payoff matrices of both players.

However, all of these works focus on theoretical complexity issues and the analogy of Nash equilibrium problem with other, apparently non-related ones such as proving Kakutani's fixed-point theorem [57]. Further, none of these works explicitly considers repeated games.

2.4.1 An adversarial imitation model

Here we expose a strategic situation in which, differently from many other studies, the agents do not have perfect knowledge of the game being played. To be precise, we present a finitely repeated imitation game where the imitator has no explicit access to the preferences of the other agent, as will be explained later. This makes our situation more realistic than traditional equilibrium-based approaches since we are not looking for the most rational strategy in a perfect-knowledge situation but in one that is based on repeated empirical observations which are likely to be deceptive.

We depart from the simple adversarial model introduced by Pelta and Yager [67]. It consists of two agents or entities S and T (the adversary) seeking to maximize their rewards, which are inversely related. The agents engage in a game where, given an external stimulus or event, they must issue a response. Agent T wants to mimic the responses of S (acting as an imitator), while agent S tries to avoid being imitated. When this scenario is repeated many times, i.e. situations of repeated conflicting encounters arise, then the situation becomes complex as the participants have the possibility to learn the other's strategy. We can see this as a repeated *imitation* game, where the imitator T learns from the actions taken by S in the past. The more frequently T imitates S correctly, the smaller is the reward of S.

When an agent S knows he is being observed by another agent T trying to learn from his behaviour, he should adopt some counter-measure to avoid this intrusion in his cognitive process. In that case, S should not choose his actions based just on his own preferences but must take into account the presence of the adversary and the fact that his behaviour should not be invariant and clear. A defense against T is to make decisions that are intended to confuse him, although S's immediate reward can be diminished. Agent S wants to force the presence of uncertainty in order to confuse the adversary while its payoff is as less affected as possible, using randomized strategies [67, 92, 65]. The authors' focus is on designing decision strategies for S that minimize the losses due to correct guesses or to non-optimal responses.

The model presented in [67] is based on two agents S and T (the adversary), a set of possible events $E = \{e_1, e_2, \ldots, e_n\}$ issued by the external environment (represented as a third agent R), and a set of potential responses or actions $A = \{a_1, a_2, \ldots, a_m\}$ associated with every event. For simplicity, we assume the set of actions is the same for all types of events. We have a payoff table $P_{n\times m} = (p_{ij})$ which collects the payoffs that S can obtain. In this table,

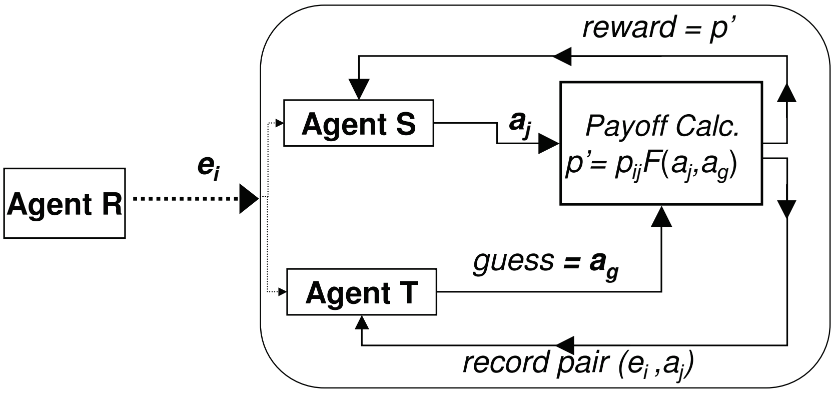


Figure 2.1: Graphical representation of the model. Events e_i are issued by agent R while the responses a_j are issued by agent S.

Algorithm 1 Sequence of steps in the model.

```
for $l = 1$ to $L$ do
    A new input $e_i$ arises.
    Agent $T$ "guesses" an action $a_g$
    Agent $S$ determines an action $a_j$
    Calculate payoff for $S$ as a function of $p_{ij}, a_g, a_j$
    Agent $T$ records the pair $e_i, a_j$
end for
```

$p_{ij} \in [0, 1]$ is the payoff for S when he chooses action a_j in response to event e_i, and T does not guess this action. The payoff table is known by S but not by T. This is a key aspect of the model and a reason why it is not strictly in the framework of imitation games.

The agents play repeatedly a simultaneous imitation game. At each encounter (also called *step*), the agents are presented an event e_i and they must select an action at the same time and without knowing what the other agent will do. If S selects action a_j and T matches that choice (i.e. T was successful in predicting S's action), then T gets a payoff of 1 and S gets 0. Otherwise, S gets p_{ij} and T gets 0. Fig. 2.1 shows a depiction of the model.

Agent T is watching agent S in order to learn from his actions. His aim is to reduce agent's S payoff by guessing which action he will take as a response to the event which arises at each step of the game. Algorithm 1 describes the steps of the model, with L being the length of the sequence of events (i.e. the number of steps they play repeatedly). The reward calculation for S at each step is defined as:

$$p' = p_{ij} \;\; F(a_j, a_g) \text{ where } F(a,b) = \begin{cases} 0 & \text{if } a = b \\ 1 & \text{otherwise} \end{cases} \tag{2.1}$$

At each step, after the payoff has been calculated, agent T is informed of what S had chosen, and then T "records" the pair (e_i, a_j) in his own memory. This information can be used in the future by T to make his predictions. The memory in which agent T keeps records of the actions taken by S is modeled as an observation matrix O, with dimensions $n \times m$. O_{ij} stores the number of times that, in the past, agent S took action a_j when the input was e_i.

2.4.2 On the unsuitability of game-theoretic equilibrium

A note on the use of game theory tools should be done here. We will first describe why one might be tempted to model the situation described above as a Stackelberg game in which the players do not need to know each other's payoffs to compute an equilibrium. Then we will explain why this approach is not correct in our opinion as the assumptions required by a Stackelberg game do not hold.

Apparently the model could be seen as an instance of a Stackelberg game described in section 2.2. In these kinds of games, one of the players (the leader) has a strategic advantage over the other player (the follower) so he is able to *commit* to a strategy that he explicitly reveals to the follower in a credible way that guarantees he will not change the strategy he has committed to [30]. Committing to a strategy does not imply to reveal which action will be played by the leader (this is only the case if the commitment is to a pure strategy), since the strategy may be itself a randomization over the available actions (which is known as a *mixed* strategy [64]). In that case, the leader just reveals the probability distribution he is using to select an action. Security games employ this concept because it is assumed that the leader, using a randomized strategy, implicitly reveals it because the follower observes the leader's behaviour for long enough before engaging in conflict. However, note that the situation presented in this section requires that the follower, in this case agent T, gives a response at each turn, i.e. at the same time that he is recording observations. The reason why one may consider that this kind of game fits our problem is the fact that agent T is able to record observations of the behaviour of S in the past. To some extent this can be considered as if agent S is revealing his strategy.

As mentioned before, the solution concept seems to be strong Stackelberg equilibrium which maximizes the leader's expected payoff while at the same time the follower maximizes his own payoff, i.e. the follower's strategy is the best one for him, given the strategy imposed by the leader. This equilibrium is usually computed using bilinear programming tools and states that the best strategy for the follower is always pure [30]. Further, in order for this equilibrium to be computed, it is not necessary that the follower knows the leader's payoffs but only the leader's mixed strategy. We may think this is what happens in our problem: agent S knows both players' payoffs, and T does not have access to S's payoffs but might take the past observations (relative frequency) as probabilities.

Having stated the reasons for which a leader-follower game seems initially suitable, now we discuss why, in our opinion, such choice is not appropriate. In our imitation model, the only way to implicitly reveal a mixed strategy is through the repeated observations recorded by agent T. The best strategy for the follower is to choose, for each event, the action that the leader selects with highest probability. If agent T takes the observed relative frequencies as probabilities, then his best response is to choose the action that he has observed more times in the past -we call *Most Frequent* (MF) to this strategy- since it reports the highest expected payoff to the follower.

However, the fact that the only clue about S's strategy are empirical observations motivates that agent S, anticipating the best-responser he may be facing, can use a deterministic, alternating strategy that systematically avoids consecutively selecting the same action twice when that action is already the most frequently used in the past for a concrete event (and therefore, T might select it as a prediction the next time that event arises). For this reason, the leader-follower equilibrium is not feasible, since it always prescribes a pure strategy for the follower that could be exploited by the leader by always selecting alternatively the best and the second-best actions, avoiding all the time a correct guess.

2.4.3 Strategies of the agents

Strategy of agent T. Recall that our game is repeated along time and S could learn to avoid the predictions when they are made on a deterministic basis. Thus T's strategy should also be mixed. The explanation for this behaviour is that observing S's past behaviour is not equivalent to considering S is implicitly revealing his actual behaviour which, for instance, could be based on some other complicated, non-randomized behaviour rules that are difficult to perceive but do not actually constitute a mixed strategy. In other words, S does not make a true commitment and, as a result, T has no reason to consider S's past behaviour to be credible for the future and trust it.

Finally, note that Nash equilibrium to mixed strategies is unfeasible since T does not know S's payoff and therefore T cannot compute such equilibrium[2]. For these reasons, we assume that agent T employs a different mixed strategy called *Proportional to Frequency* (PF) instead of MF. PF exhibited good performance according to the empirical results presented in [67]. PF means that the probability of selecting an action a_j as a prediction to an event e_i is proportional to O_{ij}, i.e. to the number of times (in percentage) such action was observed to be chosen by S in the past. In other words, the probability that T selects an action as a response to an event is actually the relative frequency with which S did the same in the past. As can be seen, it is a randomized strategy and because of that, it is safer than MF. In the forthcoming chapters, PF will be the strategy employed by T.

We must point out that constructing a model for T deserves further investigation (see for instance [107] and [55]), possibly involving learning techniques.

[2] The use of uncoupled dynamics that do not need any knowledge of the adversary's payoff to eventually converge to mixed equilibrium under some conditions is proposed as future work in the conclusions section.

Strategies of agent S. From the point of view of S, the following strategies were considered in [67]:

- **Random among k Best actions (R-k-B)**: Randomly select an action from the k best ones.
- **Proportionally Random (PR)**: The probability of selecting an action is proportional to its payoff.
- **Best of k selected Randomly (B-k-R)**: Select the best action (in terms of payoff) from a set of k possible actions selected randomly.

Chapter 3

Theoretical analysis of expected payoff with interpretable strategies

In the preceding chapter, a simple adversarial model presented in [67] was reviewed. In such work, some decision strategies for both agents were proposed and empirically evaluated using stochastic simulations. Here, our aim is to analyze such strategies from a theoretical point of view so that they can be evaluated without running a simulation. In turn, this will lead to a faster and more exact way of comparing strategies, it will facilitate comparisons with new strategies investigated in further research and will allow to understand the impact of certain components of the model. Some other recent work has also been done on this model, following a heuristic (evolutionary) approach to automatically design decision strategies [66, 90].

With this in mind, the objectives of this chapter are: (a) to provide analytical expressions for the expected payoff, based in concepts of probability theory; (b) to show the agreement between the theoretical expected payoff and the empirical results; and (c) to discuss how the strategy used and the definition of the payoff matrix affect the results.

This chapter is organized as follows. The behaviour of the agents is formalized in the remainder of this subsection. Section 3.1 presents an explanation of the details and assumptions needed to obtain the expressions of the expected payoff (with and without adversary) for each strategy, both in a general case and also in the particular conditions in which our experiments have been conducted. Section 3.2 provides graphical representations of the results in order to contrast the expected and empirical payoff, and also explains the behaviour of each strategy in terms of the payoff matrices employed. Finally, section 3.3 discusses the benefits of the results and provides new research lines in this direction.

Agents' strategies. We will analyze the strategies described at the end of section 2.4.2, namely PF (Proportional to Frequency) for agent T, and the three strategies R-k-B, PR and B-k-R for agent S. The strategy *Best action*, which always chooses the action with the largest payoff in a deterministic way, will not be evaluated theoretically because it was proved in [67] to be the worst one. The *Random* strategy (completely random) is a particular case of R-k-B in which k equals the total number of possible actions.

3.1 Calculation of the expected payoff

In this section, we will provide analytical expressions to calculate the expected payoff for S when using the the previous strategies. The basic assumptions are: agent T uses the *proportional to frequency* strategy, and the payoff matrix has a specific structure, which is defined below in the following section.

3.1.1 Notation and payoff matrix definition

Due to the need to include a certain amount of confusion (randomness) in the strategies for adversarial domains, the payoff matrix plays a key role in the calculation of the expected payoff.

Intuitively, when all the actions have quite similar payoffs, then it is not very important if a sub-optimal action is chosen for a given input. However, it becomes much more problematic if the actions' payoffs are very different, because all the sub-optimal actions provide payoffs that are considerably lower than the best one, thus leading to serious losses.

In this chapter, we will define the payoff matrices in terms of a parameter α standing for the gap between the best payoff and the rest. All matrices tested are square (the same number of stimuli and responses, namely K_{max}). Let p_{ij} be the elements of the payoff matrix P described in section 2.4.1. Every row of P is a permutation of the payoffs in the set $Y = \{1$, 1-α, 1-2α, ... 1-(K_{max}-1)$\alpha\}$. The repetition of the same value is not allowed within a row of the matrix, in order to simplify the mathematical expressions. We will refer to these values (not necessarily in this order) as $r_i, i = 1, \ldots, K_{max}$. Of course $r_i > 0$, $\forall\, i$. Let r^i be the i-th greatest value of Y. Under these considerations, matrices generated with lower values of α will have more similar values in each row, whereas higher values of α lead to very different payoffs in the same row.

Finally, let X be the random variable associated with the occurrence of every stimulus, and let $P(X = e_t)$ be a (discrete) probability distribution over the stimuli.

3.1.2 Expected payoff of R-k-B

The strategy *Random among k best actions* chooses one action (with uniform probability) from a set composed by the k best actions for the current input, being k a parameter of the strategy ($k \leqslant K_{max}$). Each candidate action can be selected with probability $1/k$, while any action that is not among the k best actions will never be chosen. Let $p_t^1, \ldots, p_t^{K_{max}}$ be the values of row t of the payoff matrix, sorted in descending order (i.e. $p_t^i > p_t^j$ when $i < j$). This is just a notation convention to indicate which the highest payoffs in a row are. Then, when there is no adversary, the total payoff E for agent S after a sequence of L inputs can be expressed as a function of the k value used in the *R-k-B* strategy, as follows:

$$E_{RkB}(k) = L \sum_{t=1}^{K_{max}} \left(P(X = e_t) \sum_{i=1}^{k} \frac{1}{k} p_t^i \right) \tag{3.1}$$

We have considered two different random events in this formula. First, the random variable T, associated to the probability that a given stimulus arises. In the original model, this distribution was considered uniform, so $P(X =$

$e_t) = 1/K_{max}$ $\forall t$ when the problem instance has K_{max} different stimuli. However, the above expression was generalized to consider any probability distribution for the stimuli of the sequence. Second, after a stimulus arises, any of the k best actions can be chosen with the same probability, $1/k$. As both events are independent, the probability that they occur simultaneously is the product of the probabilities. So, we have:

$$\begin{aligned} E_{RkB}(k) &= L \sum_{t=1}^{K_{max}} \left(\frac{1}{K_{max}} \sum_{i=1}^{k} \frac{1}{k} p_t^i \right) \\ &= \frac{L}{K_{max}\ k} \sum_{t=1}^{K_{max}} \sum_{i=1}^{k} p_t^i \qquad (3.2) \end{aligned}$$

Due to the way we defined the payoff matrices, the set of values in every row is the same, so after sorting, p_t^i are the same regardless of the row t. As a consequence we have

$$\sum_{t=1}^{K_{max}} \sum_{i=1}^{k} p_t^i = K_{max} \sum_{i=1}^{k} r^i \qquad (3.3)$$

for any $k \in [1, K_{max}]$, with r^i being the values of any row of the matrix, sorted in decreasing order.

Every row in the matrix has the values $1, 1-\alpha, 1-2\alpha, \ldots, 1-(K_{max}-1)\alpha$, so the following expression can be proven by mathematical induction:

$$\sum_{i=1}^{k} r^i = \sum_{i=1}^{k} (1-(i-1)\alpha) = k - \alpha \left(\frac{k^2-k}{2} \right) \qquad (3.4)$$

Considering these simplifications, the expression in Eq. 3.1 yields

$$E_{RkB}(k) = \frac{L}{k} \left(k - \alpha \left(\frac{k^2-k}{2} \right) \right) \qquad (3.5)$$

which represents the expected payoff for agent S without adversary.

When dealing with an adversary we also have to consider the probability of not being guessed in order to obtain a non zero reward for that action. The *R-k-B* strategy implies that, theoretically, after a certain number of inputs, any of the k best actions has been chosen the same number of times and the rest have never been chosen. Thus, we can suppose that the observed frequency for each of the k best actions is the same after a long-enough input sequence, so the probability that agent T, who is using a strategy proportional to frequency, guesses one of them is also $1/k$. Hence the probability of not being guessed is $(k-1)/k$. The probability of simultaneously choosing an action and not being guessed can be calculated as the product of the probabilities of both events since they are independent. Thus it is enough to include the factor $(k-1)/k$ (probability of not being guessed) in the general expression of Eq. 3.1, obtaining:

$$\begin{aligned} E_{RkB}(k) &= L \sum_{t=1}^{K_{max}} \left(P(X=e_t) \sum_{i=1}^{k} \frac{1}{k} \frac{k-1}{k} p_i^t \right) \\ &= L \frac{k-1}{k^2} \sum_{t=1}^{K_{max}} \left(P(X=e_t) \sum_{i=1}^{k} p_i^t \right) \qquad (3.6) \end{aligned}$$

Finally, considering simplifications due to a uniform distribution for the stimuli and the particular form of our payoff matrices (recall 3.3 and 3.4), we have:

$$E_{RkB}(k) = L \frac{k-1}{k^2} \left(k - \alpha \left(\frac{k^2-k}{2} \right) \right) \qquad (3.7)$$

which represents the expected payoff for agent S when using the *Random among k best actions* in the presence of an adversary.

3.1.3 Expected payoff of Proportionally Random

In this strategy (henceforth called PR), the probability of choosing an action is proportional to its payoff. Let z_{ti} be the probability of agent S choosing action i as a response to stimulus t using strategy PR. By definition of PR we have

$$z_{ti} = \frac{p_{ti}}{\sum_{k=1}^{K_{max}} p_{tk}} \qquad (3.8)$$

The total expected payoff, thus, can be calculated as the sum of the expected payoff for all the possible actions. This idea yields the following general expression of the expected payoff E_{PR} after a sequence of L inputs when there is no adversary.

$$\begin{aligned} E_{PR} &= L \sum_{t=1}^{K_{max}} \left(P(X = e_t) \sum_{i=1}^{K_{max}} z_{ti} p_{ti} \right) \\ &= L \sum_{t=1}^{K_{max}} \left(P(X = e_t) \sum_{i=1}^{K_{max}} \frac{p_{ti}}{\sum_{k=1}^{K_{max}} p_{tk}} p_{ti} \right) \qquad (3.9) \end{aligned}$$

Notice that this strategy does not require any additional parameter like the preceding *R-k-B* strategy.

Now let us consider again the definition of z_{ti}. Theoretically, after a certain number of inputs, say L (suppose L is *long enough*), action i will have been chosen z_{ti} L times. As the adversary agent T always uses a strategy proportional to the observed frequency to make his prediction, the probability that T guesses that action correctly can be calculated as the number of times he has observed that action when the input was t, divided by the total number of observations, L. In other words, the probability that T chooses action i is $\frac{z_{ti}L}{L} = z_{ti}$. Thus, the probability that T does not guess correctly is 1-z_{ti}.

Finally, the expected payoff for action i when dealing with an adversary can be calculated as the basic payoff of action i, p_{ti}, weighted by the probability of simultaneously choosing action i, z_{ti} and by the probability of not being predicted properly, 1-z_{ti}. Once again, both events are independent so the probability that they happen simultaneously is the product of the probabilities of each individual event. In other words, we just have to incorporate to Eq. 3.9 the factor 1-z_{ti}, which yields

$$\begin{aligned} E_{PR} &= L \sum_{t=1}^{K_{max}} \left(P(X = e_t) \sum_{i=1}^{K_{max}} z_{ti}(1 - z_{ti}) p_{ti} \right) \qquad (3.10) \\ &= L \sum_{t=1}^{K_{max}} \left(P(X = e_t) \sum_{i=1}^{K_{max}} \frac{p_{ti}}{\sum_{k=1}^{K_{max}} p_{tk}} \left(1 - \frac{p_{ti}}{\sum_{k=1}^{K_{max}} p_{tk}} \right) p_{ti} \right) \end{aligned}$$

Again, we can simplify this expression due to a uniform distribution for the stimuli and the particular form of our payoff matrices. Eq. 3.4 still holds and the whole inner summation is constant regardless of the row t because all the rows have a permutation of the same set of payoffs. This yields

$$\begin{aligned}
E_{PR} &= L\frac{1}{K_{max}}\sum_{t=1}^{K_{max}}\left(\sum_{i=1}^{K_{max}}\frac{p_{ti}}{\sum_{k=1}^{K_{max}}p_{tk}}\left(1-\frac{p_{ti}}{\sum_{k=1}^{K_{max}}p_{tk}}\right)p_{ti}\right)\\
&= L\frac{1}{K_{max}}K_{max}\left(\sum_{i=1}^{K_{max}}\frac{r_i}{\sum_{k=1}^{K_{max}}r_k}\left(1-\frac{r_i}{\sum_{k=1}^{K_{max}}r_k}\right)r_i\right)\\
&= L\left(\sum_{i=1}^{K_{max}}\frac{r_i}{C}\left(1-\frac{r_i}{C}\right)r_i\right) \qquad (3.11)
\end{aligned}$$

with r_i being the values of any row of the payoff matrix but now in no specific order. The sum of payoffs C is defined as: $C = K_{max} - \alpha(\frac{K_{max}^2 - K_{max}}{2})$

3.1.4 Expected payoff of B-k-R

Firstly, this strategy randomly chooses k different (*candidate*) actions, and secondly, the best of such actions is finally selected as a response to a given stimulus. Here, k is a parameter of the strategy.

Let a_i with $i = 1, \ldots, K_{max}$ be one of the candidate actions agent S can choose to answer to stimulus t, and let p_{ti} be the corresponding payoff. Action a_i will finally be chosen if, and only if, the following two conditions are met:

1. action a_i must be chosen as one of the k candidate actions

2. the candidate set must not contain any other action whose payoff is greater than p_{ti}, because if that were the case, then action a_i would not be selected as the *Best-among-k* candidate actions.

Any action will appear in the set of candidate actions with probability k/K_{max}. When a certain action a_i has been chosen for the candidate set, it will be finally selected as the actual response if, and only if, there are no better actions in the candidate set. The probability that this occurs can be obtained as the quotient of favorable cases divided by the total number of feasible cases.

The number of feasible cases is the number of non-sorted combinations of $k-1$ actions (the rest of the actions, apart from action a_i that is being studied) taken from the total set of remaining actions, which has $K_{max} - 1$ elements because action a_i has already been picked. The mathematical concept of *combination* captures this notion. By definition, the number of b-combinations (each of size b) from a set with a elements (size a) is the binomial coefficient:

$$comb(a, b) = \binom{a}{b} = \frac{a!}{b!(a-b)!}$$

so the number of feasible cases we need is $\binom{K_{max}-1}{k-1}$.

The number of cases that are favorable to action a_i can be computed as follows. A set of k candidate actions is favorable to action a_i if all the actions in the set have lower payoffs than a_i. The number of favorable cases is the number of favorable combinations, i.e. combinations of (k-1) actions taken only from the subset of actions that are worse than a_i.

The restriction of using only actions that are worse than a_i is the key to calculating the number of favorable combinations. The number of actions that have a better payoff than p_{ti} can be expressed as a function $B : \mathbb{R} \to \mathbb{N}$ which takes payoffs into naturals, so the number of actions that are worse can be calculated as $K_{max} - 1 - B(p_{ti})$. As we will show, we use this function to make the next expressions valid for any payoff matrix (although our particular matrices lead to simpler expressions). So, the number of favorable combinations is $\binom{K_{max}-1-B(p_{ti})}{k-1}$.

Once we have calculated the probability of each action being chosen, we can use this to obtain the general expression of the expected payoff of B-k-R for a sequence of L inputs when there is no adversary:

$$E_{BkR}(k) = L \sum_{t=1}^{K_{max}} \left(P(X = e_t) \sum_{i=1}^{K_{max}} \frac{k}{K_{max}} \cdot \frac{\binom{K_{max}-1-B(p_{ti})}{k-1}}{\binom{K_{max}-1}{k-1}} p_{ti} \right) \tag{3.12}$$

Again, we can simplify the above formula as has been done in the previous cases. If the stimulus follows a uniform probability distribution, then $P(X = e_t) = 1/K_{max} \forall t \in \{1, \ldots, K_{max}\}$. Also, if the payoff matrix has a permutation of the same set of values in each row, as in our particular payoff matrices, then the result of the inner summation is constant regardless of the value of t, so the outer summation can be ignored whilst the inner can be substituted by K_{max} multiplied by the same inner summation but now ignoring index t and using r_i. These are the same transformations we have already carried out in the preceding sections.

$$\begin{aligned} E_{BkR}(k) &= L\frac{K_{max}}{K_{max}}\sum_{i=1}^{K_{max}} \frac{k}{K_{max}} \frac{\binom{K_{max}-1-B(r_i)}{k-1}}{\binom{K_{max}-1}{k-1}} r_i \\ &= L\frac{k}{K_{max}}\sum_{i=1}^{K_{max}} \frac{\binom{K_{max}-1-B(r_i)}{k-1}}{\binom{K_{max}-1}{k-1}} r_i \qquad (3.13) \end{aligned}$$

Two more simplifications are possible for our particular matrices. First, we can suppose again that the payoffs are sorted in decreasing order (from best to worst), so $r_i > r_j$ when $i < j$. This supposition does not change the result of the summation; it just re-sorts its terms. Using this fact and considering that payoff values are not repeated within a row, function B becomes trivial: $B(r_i) = i-1$, with $i = 1, \ldots, K_{max}$. The reader should note that r_1 is the payoff of the best action, so there are 0 actions better than it, and $r_{K_{max}}$ is the worst action because all the $K_{max} - 1$ remaining actions are better. Also, as every row is a permutation of $\{1, 1-\alpha, \ldots, 1-(K_{max}-1)\alpha\}$, then $r_i = 1 - (i-1)\alpha$. This leads to the final expression of the expected payoff without an adversary:

$$E_{BkR}(k) = L\frac{k}{K_{max}}\sum_{i=1}^{K_{max}} \frac{\binom{K_{max}-1-(i-1)}{k-1}}{\binom{K_{max}-1}{k-1}} (1-(i-1)\alpha) \qquad (3.14)$$

Finally, we can apply the same reasoning used in the previous section in order to calculate the probability of not being guessed. If z_{ti} represents the probability that agent S chooses an action, then it also represents the probability of being guessed when T uses a strategy proportional to the observed frequency. So we need to introduce the factor $1 - z_{ti}$ into the expressions above. It should be recalled that, in strategy *B-k-R*, we have

$$z_{ti} = \frac{k}{K_{max}} \frac{\binom{K_{max}-1-B(p_{ti})}{k-1}}{\binom{K_{max}-1}{k-1}}$$

so expression 3.12 turns into the following expression in order to include this probability of not being guessed:

$$E_{BkR}(k) = L \sum_{t=1}^{K_{max}} \left(P(X = e_t) \sum_{i=1}^{K_{max}} z_{ti}(1 - z_{ti})p_{ti} \right) \tag{3.15}$$

Using our particular matrix,

$$z_i' = \frac{k}{K_{max}} \cdot \frac{\binom{K_{max}-1-(i-1)}{k-1}}{\binom{K_{max}-1}{k-1}}$$

with z_i' being the probability of choosing action a_i regardless of the row (remembering that every row is a permutation of the same set of values). Then expression 3.14 becomes:

$$E_{BkR}(k) = L \sum_{i=1}^{K_{max}} z_i'(1 - z_i')(1 - (i - 1)\alpha) \tag{3.16}$$

which represents the expected payoff for agent S when using strategy *B-k-R* in the presence of an adversary.

3.2 Computational experiments and results

In the previous sections we provide analytical expressions for the expected payoff that agent S can achieve using the decision strategies presented above, with and without an adversary.

In this section, we will assess the impact of the payoff matrix on the final payoff achieved by agent S. From a set of matrices, we will calculate the expected payoff using the theoretical expressions and the effective payoff calculated through simulations and then check the agreement between both results.

We will test several 20x20 matrices (i.e. $K_{max} = 20$). This means that the values of each row in a given matrix are in the set $\{1, 1-\alpha, \ldots, 1-19\alpha\}$. To avoid negative payoffs, we need $1 - 19\alpha > 0$ so the following inequality holds:

$$1 - 19\alpha \geqslant 0 \iff \alpha \leqslant 0.052 \tag{3.17}$$

A full factorial design of the experiments was performed, where the factors and their levels are the following:

- First factor: $\alpha \in \{0.001, 0.010, 0.020, 0.030, 0.040, 0.050\}$, which means there are six different payoff matrices to be tested with each strategy.
- Second factor: agent S strategy = $\{PR, R\text{-}k\text{-}B, B\text{-}k\text{-}R\}$.
- Third factor: parameter $k \in [2, 3, \ldots, 20]$. This is only considered when using the strategies *R-k-B* and *B-k-R*.

In the third factor, we omitted $k = 1$ for two reasons: first, in *B-K-R* it is equivalent to a completely random strategy (which is already included in *R-K-B* with k=K_{max}), and second, in *R-K-B* it is equivalent to always choosing the best action, which is the worst strategy according to the results presented in [67].

Every combination of α, strategy and k value (if applies) has been evaluated theoretically and empirically. In this last case, we performed 2000 runs (simulations) of Algorithm 1 where the length of the input sequences employed was $L = 250$. Inputs were considered independent and they were generated using a uniform distribution. As the best response to any stimulus always returns a payoff of 1, the maximum possible payoff after a 250-input sequence is precisely 250. This would be the ideal situation (if there was no adversary). The payoff assigned to each combination is the average payoff over the 2000 simulations.

3.2.1 Results for Proportionally Random

The empirical and theoretical results regarding the expected payoff as a function of the α value (with and without adversary) are shown in Fig. 3.1. Empirical results are shown with markers while theoretical ones are displayed with lines. The first element to notice is the perfect agreement between the analytical expressions and empirical results in both cases: with and without adversary.

The relation between α and payoff is the second element to notice. As the former increases, the latter decreases. The reasons for this are the following. As α grows the payoff matrix is less balanced (the differences among payoffs are greater). Using this decision strategy, when a sub-optimal action is selected, the contribution to the total payoff is lower than that which could

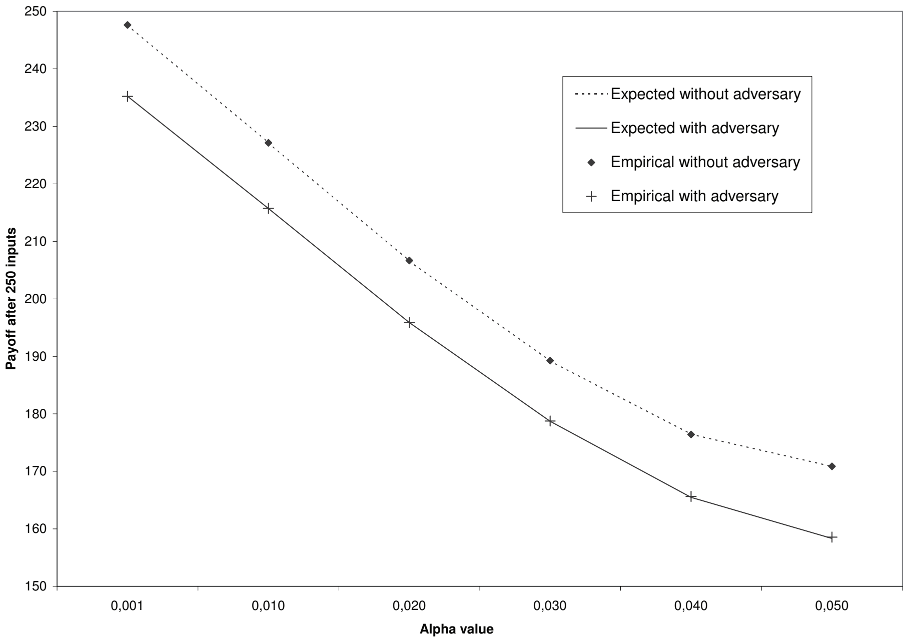


Figure 3.1: Expected and empirical payoff for Proportionally Random

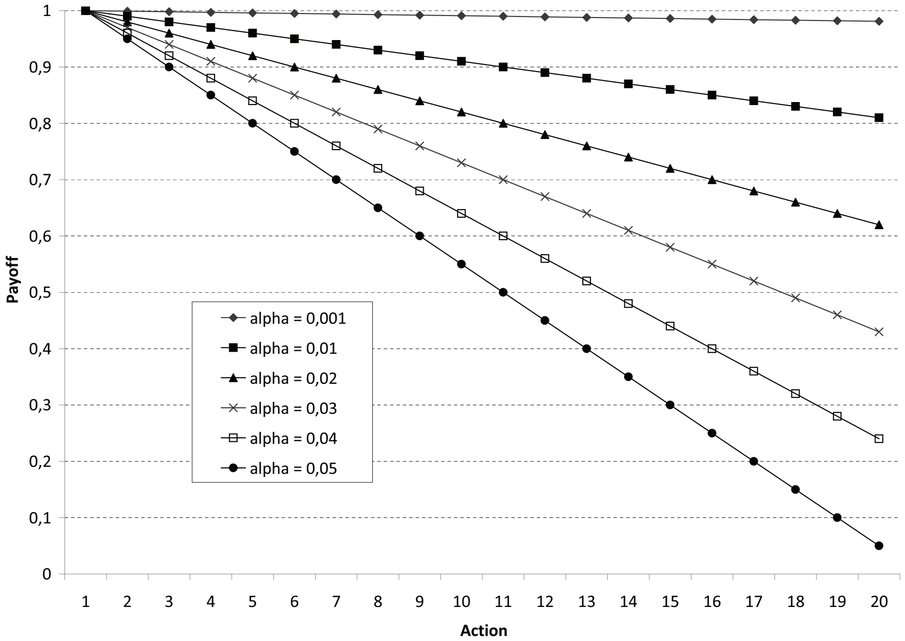


Figure 3.2: Payoff per action in every matrix tested

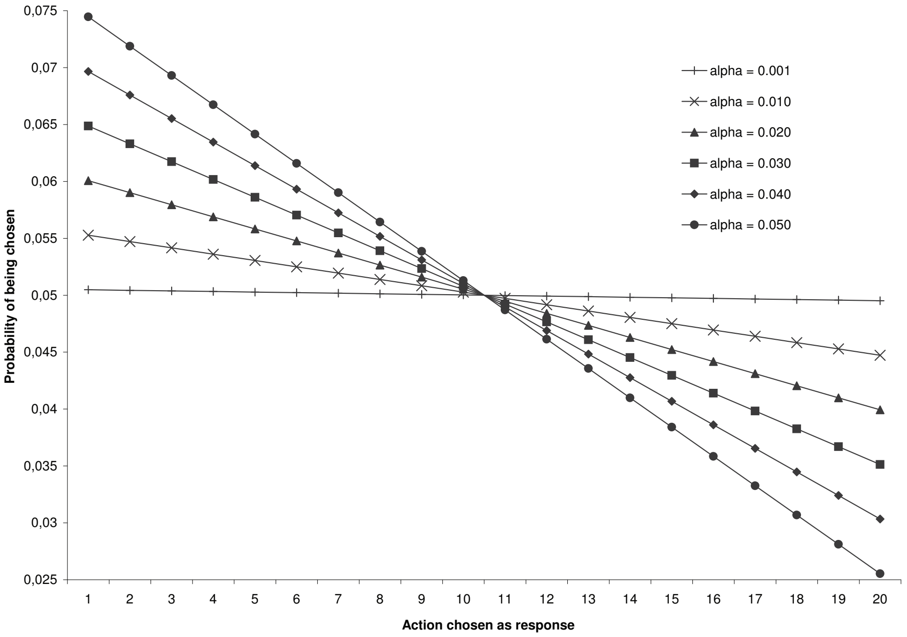


Figure 3.3: Probability distribution of the actions for different α values in PR

be achieved selecting the best alternative. In other words, the cost (potential loss in payoff) of choosing a sub-optimal solution is higher. This is clearly observed in Fig. 3.2 where the payoff assigned to every action is shown. Although the best action always returns a payoff of 1, the other actions' payoffs are diminished as the α value increases.

In turn, as the payoff matrix is less balanced, the probability of choosing one of the best actions grows while the probability of choosing one of the worst actions diminishes. Fig. 3.3 shows this phenomenon. Each line represents the probabilities of choosing each of the twenty possible actions with a different payoff matrix. Here, action 1 is the best while action 20 is the worst. The almost horizontal line corresponds to the matrix generated with $\alpha = 0.001$ (payoffs and in turn, probabilities are well balanced), while the most-sloped diagonal line corresponds to $\alpha = 0.050$ (probabilities are very unbalanced). It should be noted how the probabilities of choosing one of the 10 best actions tend to grow while the rest tend to diminish.

Now, the situation is as follows. The best actions have more chances of being selected but, from the point of view of the adversary, they have

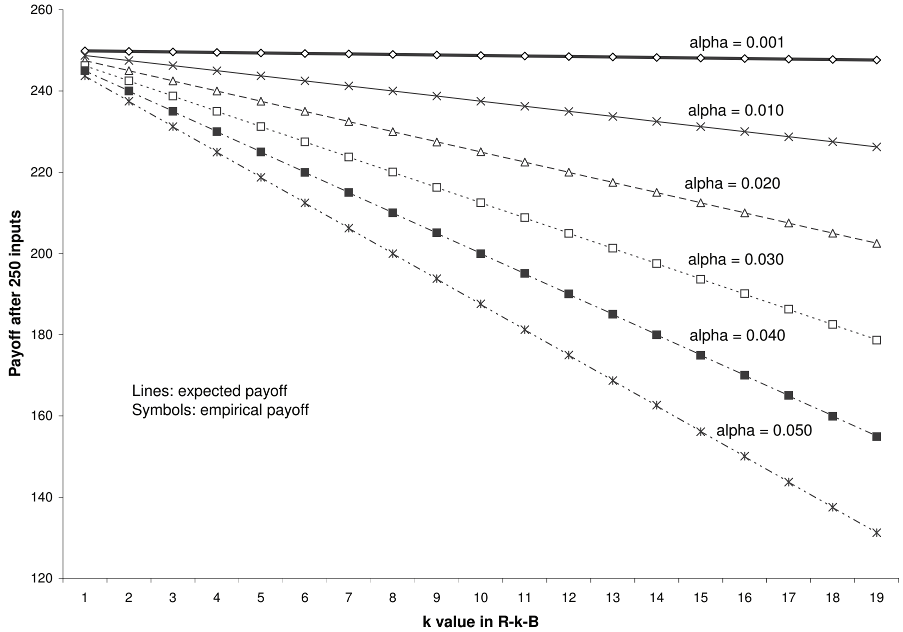


Figure 3.4: Expected and empirical payoff for R-k-B without adversary

been more frequently observed. In this situation it would be easier for the adversary to properly predict the action that S will choose.

3.2.2 Results for R-k-B

In order to analyze R-k-B strategy we would also need to take into account the value of the parameter k.

In Fig. 3.4 we present the expected and empirical payoff attained using this strategy (without adversary), as a function of k for every payoff matrix tested. Again, we can first observe the perfect agreement between theoretical expressions and empirical results. It is also clear that for a given α, the payoff depends linearly on the value of k, with a higher slope as α increases. As k increases, a highly-randomized strategy is obtained, so the chances of selecting sub-optimal responses also increase.

When α is low, this is not very problematic, but when α increases, the selection of sub-optimal actions with low payoffs leads to substantial losses.

When dealing with an adversary, the situation is very different as can be

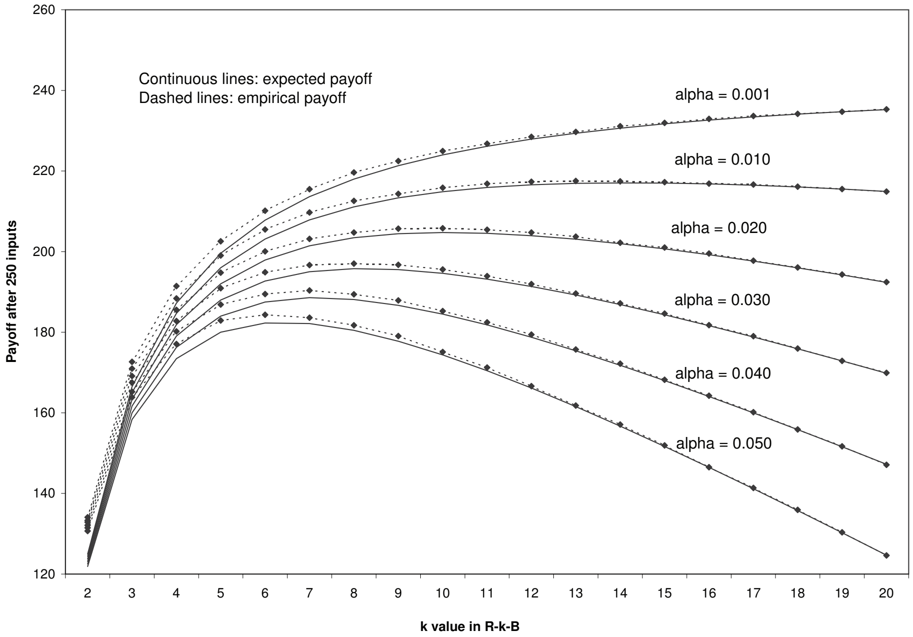


Figure 3.5: Expected and empirical payoff for R-k-B with adversary

observed in Fig. 3.5.

Again, the agreement between the empirical and theoretical approaches for the payoff calculation is high. In this case, the minor differences between the expected (dashed line) and the empirical (continuous line) payoff at the beginning of each curve can be explained in terms of probabilities. Probability estimations only hold when the experiments are repeated a great number of times[1].

Here, partially-randomized strategies may be better because they make our behaviour more unpredictable, although sub-optimal actions do not report so much benefit. When the values of the matrix are very similar, increasing k is always better because it results in a more unpredictable behaviour while keeping the total payoff very close to the optimal. When α is high and the values of the matrix are not as similar, a more random behaviour is not

[1] We have repeated the experiments with a longer input sequence (1000 stimuli) and these differences become smaller, which means that the formula holds when the input sequence is *large enough*. Results are not shown as they do not substantially contribute to the analysis.

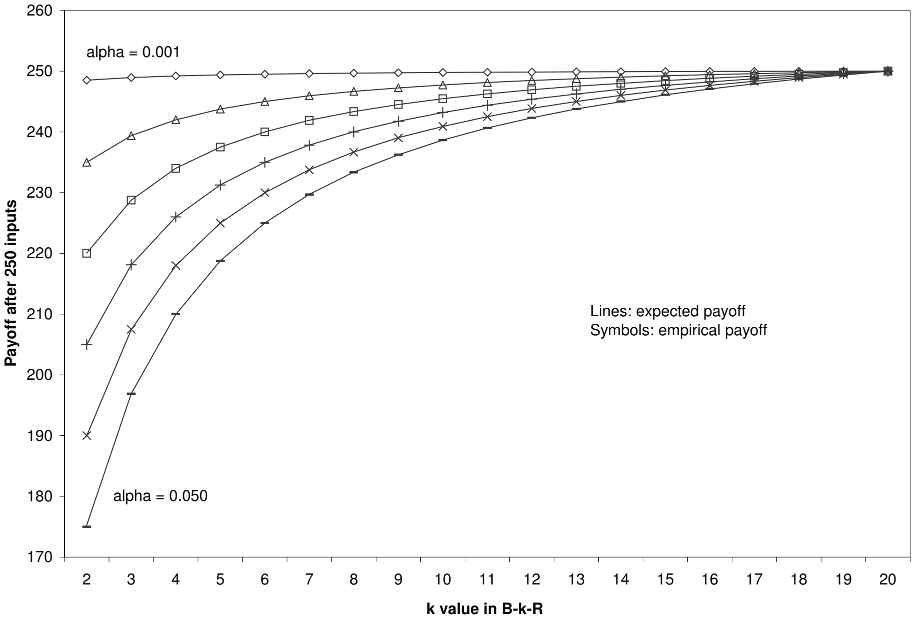


Figure 3.6: Expected and empirical payoff for B-k-R without adversary

always the best alternative because the losses due to sub-optimal actions are important, yet hard to guess. In this situation, the precise amount of randomness needed can be calculated as the exact value for k that maximizes the total payoff for a given payoff matrix. That is the point where the payoff-vs-k curve reaches its absolute maximum and, as can be seen in Fig. 3.5.

3.2.3 Results for B-k-R

In this strategy, we will also consider the results with and without adversary, and also taking into account all the possible values for the parameter k. It should be remembered that the strategy randomly selects k actions and then it chooses the best one available.

As expected, when there is no adversary, it is always better to increase the k value because it enlarges the candidate set, which allows to consider more actions and thus to choose the best among them (see Fig. 3.6). If k equals K_{max}, then all the actions are always selected within the candidate set, thus

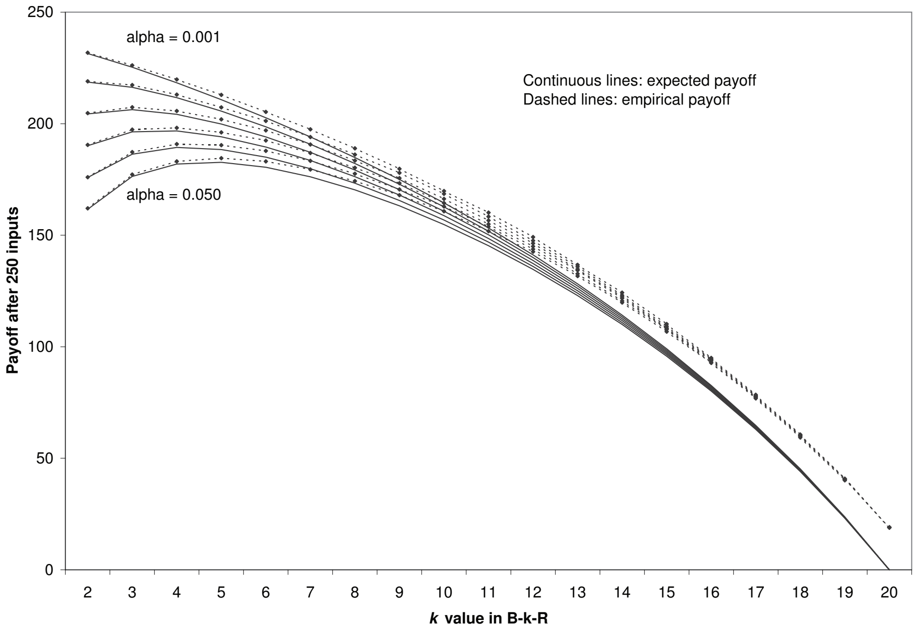


Figure 3.7: Expected and empirical payoff for B-k-R with adversary

the strategy degenerates into always selecting the best action. Obviously, this strategy works perfectly only when no adversary is present. On the contrary, if k is low, then the set of candidate actions becomes very randomized, and it may happen quite often that good actions are not selected. When this occurs, losses become more significant, especially if the differences between the payoffs are large (i.e. when α is large). This fact explains the reasons for the different slopes in the curves.

When the adversary is included in the game, the results are those shown in Fig. 3.7. We can observe that as the value of k increases, the expected payoff is always slightly less than the empirical one. This difference can be easily explained. At the beginning of an experiment, when the observation matrix used by agent T only contains zeros, the first prediction made by T is totally random and possibly wrong. This can happen once per each different input, i.e. up to K_{max} times because the observation matrix has K_{max} rows that are initialized to all 0. As a result, agent S may attain some *extra* payoff at the first steps of each experiment. If the length of the input sequence is large enough, this effect has less influence on the total payoff.

However, it can be clearly appreciated in the extreme case, $k = K_{max}$, which is equivalent to always choosing the best action, as explained above. The expected payoff for S in this case is 0 because agent T should always be able to properly guess our action, but the empirical payoff is around K_{max} due to the phenomenon of the initial steps. It should be recalled that the best action has a payoff of 1 and the observation matrix has K_{max} independent rows. The same phenomenon also explains the differences shown in Fig. 3.5. It also shows that the impact of the payoff matrix decreases as k increases.

3.3 Conclusions

In this chapter, we have studied decision strategies in an adversarial model from a theoretical point of view, providing analytical expressions for the expected payoff when the adversary is present or is not. An interesting feature of the model studied is that the assumptions are minimal. We observed an almost perfect agreement between theoretical and empirical results, so the expressions can be used to evaluate the strategies without running a simulation, thus allowing to analyze the impact of model features (such as number of alternatives/stimuli), or to make comparisons with other strategies faster. In addition, we have discussed the role of payoff matrices in the results. Under the particular definition we used, it is clear that such matrices have a great impact in the payoff obtained. More studies are needed using different types of matrices, for example, where the sum of the payoffs in every row is the same, or where different alternatives with the same reward can exist. Game-theoretic studies of equilibrium may also be useful, specially those that make use of particular equilibrium concepts for repeated games.

The understanding of these basic features will allow to develop better strategies for this simple adversarial model (see [91] for more sophisticated dynamic strategies that make use of the same ideas about the expected payoff) and can pave the way to study variations that are yet not considered but realistic, such as adversaries not taking decisions (only observing), or sequence of stimulus with certain correlations (between the last response and the next random stimulus) and so on. Some of these aspects are already under study.

Chapter 4

Forgetting as a way to avoid deception

In the previous chapter, we provided analytical expressions for the expected payoff, showed the agreement between the theoretical expected payoff and the empirical results, and discussed how the strategy used and the definition of the payoff matrix affect the results. However, everything was done assuming that agent S's strategy did not change along the time, and that the imitator had an unlimited observation memory. These two conditions will now be challenged.

The aim of this chapter is twofold. Firstly, we will propose and analyze randomized decision strategies for agent S that are not constant along the time, but change at certain time steps of the repeated game. We tackle the strategy design as a constrained non-linear optimization problem whose solution gives both the exact moment at which agent S must change and the new strategy he must use. Secondly, we will evaluate such strategies in a different scenario in which agent T forgets the oldest observations, in order to test if this is benefitial or not for T. This will test if the strategies presented in previous work are severely affected by wrong assumptions about the adversary or not. To be precise, we want to answer the following questions:

1. Do the results obtained with the analytical expressions match those obtained by empirical simulations?

2. Do dynamic mixed strategies outperform a static mixed strategy in

terms of expected payoff?

3. Does a limited memory have an impact over the expected payoff? Is it beneficial or detrimental for agent T to forget the oldest observations?

The chapter is organized as follows. Section 4.1 describes the main characteristics and components of the model used and discusses the suitability of game theoretic equilibrium concepts. Section 4.2 is a review of the material presented in [91], and deals with the need of randomized strategies, explains a static mixed strategy for agent S, and also introduces the concept of dynamic, time-changing decision strategies as opposite to the former static strategy. The analytical expression of the expected payoff attained by S when using both kinds of strategies is explained step by step. An optimization process is also introduced here to obtain the best dynamic strategy under certain assumptions. Section 4.3 modifies the assumptions made in the preceding section to deal with an adversary with a limited observation memory, which yields a generalized expression of the expected payoff. In Section 4.4 we describe the computational experiments performed and the results obtained. They are aimed at checking the validity of the theoretical expressions with empirical simulations and comparing the performance of static and dynamic strategies with both unlimited and limited observation memory. Finally, Section 4.5 is devoted to discussions and further work.

4.1 Simplified version of the model

The model is a simplified version of the one depicted in 2.1. We will consider no external events or, equivalently, only one type of event. The reason is that the events of the model were independent so that the actions and predictions about one type of event did not affect the strategy that should be adopted for other types of events, so we can focus on the study of a single type of event. Hence, S's payoff table is now a set of payoffs $P = \{p_1, p_2, ..., p_m\}$ associated to the m actions. Accordingly, T's memory is an observation vector $O = \{o_1, ..., o_m\}$ that he updates after each encounter. Element o_j stands for the number of times that, in the past, agent S has chosen action a_j. Agent T may take into account this information for future decisions. The

Table 4.1: Payoff matrix of the simultaneous game played at each step. Showing payoffs for S and T for each possible outcome.

		T				
		a_1	a_2	a_3	$\dots$	a_m
S	a_1	$(0, 1)$	$(p_1, 0)$	$(p_1, 0)$	$\dots$	$(p_1, 0)$
	a_2	$(p_2, 0)$	$(0, 1)$	$(p_2, 0)$	$\dots$	$(p_2, 0)$
	a_3	$(p_3, 0)$	$(p_3, 0)$	$(0, 1)$	$\dots$	$(p_3, 0)$
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	a_m	$(p_m, 0)$	$(p_m, 0)$	$(p_m, 0)$	$\dots$	$(0, 1)$

game mechanics are the same already described: at each step, if S selects action a_j and T matches that choice (i.e. T was successful in predicting S's action), then T gets a payoff of 1 and S gets 0. Otherwise, S gets p_j and T gets none.

In this chapter, a slightly more formal treatment is adopted to show a comparison with the game-theoretic Nash equilibrium (section 4.4.2) if the payoffs were of common knowledge, i.e. if agent T was a clairvoyant adversary. Therefore, the game outcomes and payoffs are summarized in the payoff matrix of the simultaneous game played at each step, shown in Table 4.1. Notice that T is equally interested in guessing any action because all the correct guesses report the same payoff to him, while S does have an incentive for choosing some actions instead of some others because the payoff he may attain when not guessed is higher.

4.2 Behaviour of the agents

The strategy that agent T will be using (Proportional to Frequency) has been described in section 2.4.2. In Sections 4.2.1 and 4.2.2 we review the approaches published in [91] for agent S to emphasize the assumptions that allowed to compute such decision strategies. Sections 4.2.3, 4.3 and 4.4 present a completely novel study on how a change on certain adversarial conditions can affect the performance of decision strategies that were supposed to be optimal under the original assumptions when these do not hold anymore.

4.2.1 Static Mixed Strategy for Agent S.

Agent S could use a totally deterministic strategy that always select the action with the highest payoff. However, this would be very easy to learn for agent T so he would quickly predict this behavior correctly after a short number of repetitions. S could also employ a totally random strategy that would select an action in a totally random way. This behaviour would be very hard to learn from observations but, on the other hand, the payoff attained would be low because bad actions (i.e. those with low payoff) may be selected with the same probability than best actions.

As stated before, a mixed strategy is a set of weights representing a probability distribution over the actions. When a player has to do a movement, he uses this probability distribution to choose his action. In our model, we are interested in the *best randomization* or, in other words, the set of weights that lead to the highest payoff when playing against agent T. From now we will use the expression *set of weights* to refer to a probability distribution over the actions of the model. Thus such weights are in [0, 1] and their sum is 1.

With these weights, it is possible to calculate the expected payoff for a given player, which is the sum of all the possible outcomes of the game weighted by the probability that each outcome eventually arises and by the payoff that outcome reports to the player. In the adversarial model we are dealing with, this means that we can weight each payoff in vector P by the probability that agent S eventually gets that payoff. This probability can be computed as the product of the probabilities of two independent events happening simultaneously. Agent S will attain payoff p_j if two conditions hold:

i Agent S must select action a_j as a response. This probability is noted α_j, although we will refer to it as the *weight* used by S to select a_j.

ii In addition, S will only get the payoff p_j if agent T does not successfully predict his response. Actually this probability is independent of (i) if we consider it as a conditional probability that is conditioned to the fact that S has already selected action a_j.

This (conditional) probability can be computed as follows. In case agent S is using a non-variant weight α_j (a *static* mixed strategy) during a sequence of L repetitions of the game, then the probability that agent T does not guess his actions if T uses PF strategy is (1 - α_j). The explanation is as follows. After L repetitions, since agent S uses α_j, then action a_j would have been selected $L \cdot \alpha_j$ times, and this is what agent T sees in O_j. The probability that T selects action a_j as a prediction is

$$P_{guess}(j) = \frac{O_j}{\sum_{j=1}^{m} O_j} = \frac{L{\cdot}\, \alpha_j}{L} = \alpha_j \tag{4.1}$$

with m being the number of actions available. The probability of not being guessed correctly is then $1 - P_{guess}(j) = 1 - \alpha_j$.

Taking into account the probabilities of the two conditions described above yields the following expression of the expected payoff for agent S after a sequence of L encounters when he uses weights α_j to select his actions:

$$EP_{\text{static}} = L \sum_{j=1}^{m} \alpha_j (1 - \alpha_j) p_j \tag{4.2}$$

If we want to maximize the expected payoff, we have to maximize expression 4.2 by computing the values of the optimal weights α_j. This can be achieved using numerical methods for constrained optimization. The optimization problem can be formalized as follows.

$$\max_{\{\mathrm{\alpha_j}\}} \sum_{j=1}^{m} \alpha_j{\cdot}\, (1 - \alpha_j){\cdot}\, p_j \tag{4.3}$$

subject to:

$$\begin{aligned} \sum_{j=1}^{n} \alpha_j &= 1 \\ \alpha_j &\geqslant 0 \end{aligned} \tag{4.4}$$

4.2.2 Dynamic Mixed Strategy for Agent S

In the previous section we described a static strategy for agent S. It was static in the sense that the same set of weights is used by S to make decisions in the repeated encounters. The static mixed strategy described above can be viewed as one single period, because the weights computed by S do not change along time. Now the idea is to define several periods and calculate the optimal mixed strategy for every period. The *length* of a period is the duration of the period, i.e. the number of consecutive encounters of the game during which agent S will use the same mixed strategy.

For a given period, the set of optimal weights is different from that of other periods. Let N_h be the length of the h-th period. The next example illustrates this concept. Suppose that we have a sequence of length $L = 100$ encounters. Then, we can define for instance 4 periods of length $N_1 = 30$, $N_2 = 10$, $N_3 = 20$ and $N_4 = 40$. Fig. 2 shows a depiction of a dynamic mixed strategy.

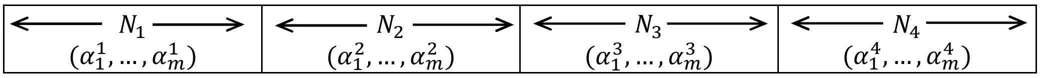

N_1
$(\alpha_1^1, \ldots, \alpha_m^1)$
N_2
$(\alpha_1^2, \ldots, \alpha_m^2)$
N_3
$(\alpha_1^3, \ldots, \alpha_m^3)$
N_4
$(\alpha_1^4, \ldots, \alpha_m^4)$

Figure 4.1: Example of a dynamic strategy with 4 different periods. The letters inside each rectangle represent the length of that period and the mixed strategy to be used in that period.

In order to calculate the best randomization under this scenario, we need to apply constrained optimization methods to compute the values of the optimal weights for each period, along with the optimal length of every period. In principle, we should also compute the optimal number of periods but this, as will be seen later, requires solving independent optimization problems since the number of periods affects the total number of variables of the problem. For this reason, the number of periods is assumed to be known. In section 4.4.2 we will study the influence of this parameter.

The objective function of such constrained optimization problem is the expression of S's expected payoff for a dynamic strategy, which can be computed as follows. Let $(\alpha_1^h, ..., \alpha_m^h)$ be the set of weights that agent S uses to choose an action during the h-th period. Then, within a given period, α_j^h represents the probability that S selects action a_j. The difficult part of the expression we need is computing the probability of not being guessed, which is the same within a period but different from one period to another. After the first period of length, say, N_1, the observation vector O has the following values (absolute frequencies of the responses given in the past by S):

$$O = \begin{pmatrix} N_1 \cdot \alpha_1^1 & \dots & \dots & N_1 \cdot \alpha_m^1 \end{pmatrix}$$

Clearly, the probability of not being guessed P_{NG} during the first period is (1-α_j^1), according to the same explanation given in Section 4.2.1. This reasoning becomes more complicated when considering the observation vector at the end of the second period, whose length is N_2:

$$O = \begin{pmatrix} N_1 \cdot \alpha_1^1 + N_2 \cdot \alpha_1^2 & \dots & \dots & N_1 \cdot \alpha_m^1 + N_2 \cdot \alpha_m^2 \end{pmatrix}$$

According to these values, the probability *at the end of the second period* that agent T selects action a_j as a prediction is

$$P_{guess}(j) = \frac{O_j}{\sum_{j=1}^{m} O_j} = \frac{N_1 \cdot \alpha_j^1 + N_2 \cdot \alpha_j^2}{N_1 + N_2}$$

so the probability of not being guessed *at the end* of the second period is

$$P_{NG}(j) = 1 - P_{guess}(j) = \frac{N_1(1 - \alpha_j^1) + N_2 \cdot (1 - \alpha_j^2)}{N_1 + N_2}$$

What happens in the middle, i.e. *during* the second period? The probability of not being guessed changes at every step because the number of times each response has been observed by T varies along time. This variation can be modeled as follows. At a certain step s of the second period (s is measured from the beginning of the period, so $0 \leqslant s \leqslant N_2$), the probability that T correctly predicts response a_j is

$$P_{guess}(j, s) = \frac{O_j}{\sum_{j=1}^{m} O_j} = \frac{N_1 \cdot \alpha_j^1 + s \cdot \alpha_j^2}{N_1 + s} \tag{4.5}$$

Notice we have added a second argument to this probability to emphasize that it also depends on the step s of the simulation. The probability of not being guessed is then

$$P_{NG}(j, s) = 1 - P_{guess}(j, s) = \frac{N_1(1 - \alpha_j^1) + s(1 - \alpha_j^2)}{N_1 + s}$$

As stated before, notice that this probability changes at every step within a period. Now, it is possible to generalize this reasoning to obtain the probability of not being guessed at step s of the h-th period ($0 \leqslant s \leqslant N_h$):

$$P_{NG}(j,h,s) = 1 - P_{guess}(j,h,s) = \frac{\sum_{k=1}^{h-1} N_k(1-\alpha_j^k) + s(1-\alpha_j^h)}{\sum_{k=1}^{h-1} N_k + s}$$

Argument h points out the period to which s belongs, although this is just a formalism that will be eliminated below. The expression of the total expected payoff with H periods of length N_h is a generalization of Eq. 4.2, using 4.6 as the probability of not being guessed:

$$EP_{\text{dynamic}} = \sum_{h=1}^{H}\sum_{s=1}^{N_h}\sum_{j=1}^{m} \alpha_j^h \cdot \frac{\sum_{k=1}^{h-1} N_k(1-\alpha_j^k) + s(1-\alpha_j^h)}{\sum_{k=1}^{h-1} N_k + s} p_j \tag{4.6}$$

The next expression should be also verified: $\sum_{k=1}^{H} N_h = L$. Recall that L is known in advance by both agents.

Once again, the reader should note that this expression is valid only when T uses the strategy Proportional to Frequency. Further, S must know in advance the length of the sequence (the exact number of steps of the complete simulation) to add the last constraint to the optimization process.

With this approach, the number of unknown parameters is greater than that of static mixed strategies. Recall that the number of periods H is given by the user in advance, but the optimal length of each period N_h is unknown. In addition, instead of computing only m weights as in the static mixed strategy, we have to compute $m \cdot H$ weights (H sets of weights) plus the H lengths of the periods N_h for $h = 1, ..., H$ which are integer values. The optimization problem in this case can be formulated as follows:

$$\max_{\{\mathrm{\alpha_{ij}^h}\}\cup\{\mathrm{N_i^h}\}} \sum_{h=1}^{H}\sum_{s=1}^{N_h}\sum_{j=1}^{m} \alpha_j^h \cdot \frac{\sum_{k=1}^{h-1} N_k(1-\alpha_j^k) + s(1-\alpha_j^h)}{\sum_{k=1}^{h-1} N_k + s} p_j \tag{4.7}$$

subject to:

$$\sum_{j=1}^{m} \alpha_j^h = 1, \quad h = 1, ..., H$$

$$\alpha_j^h \geqslant 0, \quad j = 1, ..., m \quad h = 1, ..., H$$

$$\sum_{h=1}^{H} N_h = L \tag{4.8}$$

As explained before, the number of unknown variables to be optimized when we are computing a dynamic strategy with H periods is $m \cdot H + H = H(m + 1)$. Therefore, a different optimization problem with a different number of variables is generated for each H value (see Fig. 4.1). The way to find the best choice for H is to solve the optimization problem for different H values and accept the strategy that provides the highest payoff for S.

4.2.3 A generalized notation for the expected payoff

In this part we will rewrite expression 4.6 with a slightly different notation that simplifies and generalizes the previous one and that will be employed in Section 4.3.1.

For a given step d of the simulation, let $PCD(d) : N \to N$ be the index (starting in 1) of the period that precedes the one to which d belongs, or 0 if d belongs to period 1. The step d is measured from the beginning of the simulation, so in this case $1 \leqslant d \leqslant L$. The name *PCD* stands for *preceding.* Let $H(d)$ be the period to which step d belongs. $H(d)$ and $PCD(d)$ can be defined in terms of the lengths of the periods as follows:

$$PCD(d) = \begin{cases} max\{k \in N : \sum_{h=1}^{k} N_h < d\} & \text{if } d > N_1 \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

$$H(d) = PCD(d) + 1$$

Now let $Y(d)$ be the number of steps measured from the beginning of period $H(d)$:

$$Y(d) = d - \sum_{k=1}^{PCD(d)} N_k \tag{4.10}$$

Fig. 4.2 depicts the meaning of these functions when applied for instance to step $d = 224$ belonging to the third period of a strategy whose 5 first periods have length 100, 90, 112, 76 and 120.

Finally, let $OBS(d, j)$ be the value of cell O_j after d steps. It can be expressed in terms of the weights used by S in his decisions, as follows. The

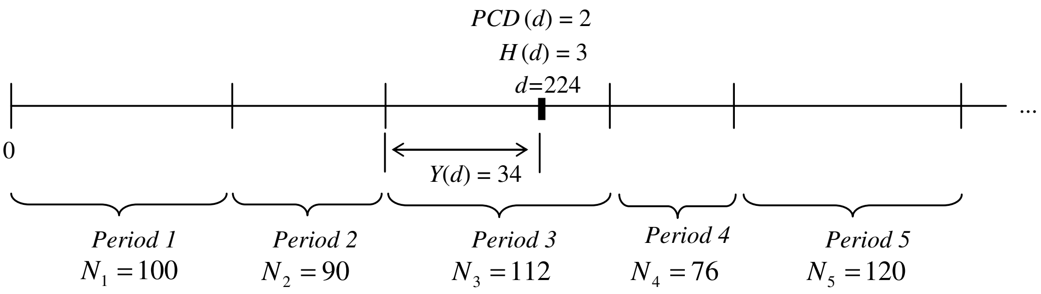


Figure 4.2: Example of dynamic strategy and the notation explained in this section

otherwise part was written to make this function also valid for the case of limited memory, as explained later.

$$OBS(d,j) = \begin{cases} \left(\sum_{k=1}^{PCD(d)} N_k \alpha_j^k\right) + Y(d)\alpha_j^{H(d)} & \text{if } d > 0 \\ \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

As can be seen, expression 4.11 is the numerator of 4.5, generalized for an arbitrary number of steps d. The total number of observations in the observation vector after d steps is exactly d:

$$\begin{aligned} \sum_{j=1}^{m} OBS(d,j) &= \left(\sum_{j=1}^{m}\sum_{k=1}^{PCD(d)} N_k \alpha_j^k\right) + \sum_{j=1}^{m} Y(d)\alpha_j^{H(d)} = \\ &= \left(\sum_{k=1}^{PCD(d)} N_k \sum_{j=1}^{m} \alpha_j^k\right) + Y(d)\sum_{j=1}^{m}\alpha_j^{H(d)} = \\ &= \left(\sum_{k=1}^{PCD(d)} N_k\right) + d - \left(\sum_{k=1}^{PCD(d)} N_k\right) = d \end{aligned}$$

so the probability of being guessed after d steps when choosing action a_j can be rewritten as $\frac{OBS(d,j)}{d}$ if $d > 0$. In the first encounter ($d = 0$), nothing has been observed yet by agent T so we assume he makes a prediction in a totally random way. The probability of being guessed is thus $1/m$ in this case. With these functions, expression 4.6 can be rewritten as

$$\begin{aligned} EP_{\text{dynamic}}(d) &= \sum_{j=1}^{m} \alpha_j^1 (1 - 1/m) p_j \\ &+ \sum_{s=2}^{d} \sum_{j=1}^{m} \alpha_j^{H(s)} \left(1 - \frac{OBS(s-1,j)}{s-1}\right) p_j \qquad (4.12) \end{aligned}$$

where d is the number of steps of the simulation, provided that we have defined enough periods to cover, at least, that number of steps.

4.3 Forgetting as a way to avoid deception

The key point of the optimized mixed strategies presented in [91] is that they intend to manipulate the observations made by T so that S has long periods to safely choose very good actions without being guessed. In other words, they were exploiting the fact that T considers all the observations done in the past and gives all of them the same importance. This means that, if T observed a very frequent action many times in the past, and uses Proportional to Frequency, then it is difficult for him to predict something different in the future unless he observes a different action approximately as many times as the first one. We can see this phenomenon as a sort of "inertia" which makes T difficult to recover after a series of observations of one very frequent action. For that reason, S manipulates with its actions T's observation vector to his convenience, and in fact the optimization process tells S the best way to do this at every step.

As mentioned in Sections 4.2.1 and 4.2.2, expressions 4.2 and 4.6 consider that T is playing proportional to frequency with an unlimited memory, i.e. T would be annotating every action a_j forever. We hypothesize that a limited memory for T would be beneficial for T (or harmful for S) because it would attenuate or even eliminate this inertial behaviour that makes T easy to manipulate. This way T can be thought of as a more intelligent adversary that only takes into account the most recent observations and simply forgets older ones.

4.3.1 Expected payoff with limited memory

The following reasoning can be applied to obtain the expression of the expected payoff when T uses PF but has limited memory. As stated before, limited memory means that the number of observations T can store in his memory is limited to a certain number. When the memory is full and a new action a_j is to be stored, the oldest observation is simply deleted. Suppose that the oldest observation was action a_t, then the deletion consists in decreasing the value O_t in one unit. This idea can be applied to obtain an expression of the expected payoff as follows. Suppose that we have observed $N_1 + s$ steps, and that agent S was using a mixed strategy of two periods of lengths N_1 and N_2. Then, recall that after $N_1 + s$ steps, the observation vector of T will look like

$$O = \begin{pmatrix} N_1 \cdot \alpha_1^1 + s \cdot \alpha_1^2 & \dots & \dots & N_1 \cdot \alpha_m^1 + s \cdot \alpha_m^2 \end{pmatrix}$$

Suppose that now we add a new observation. We do not know what exact action S will do in the next step, but we can model the behaviour using the probability that the action is each of the possible ones. So after one additional step, the same row will now look like

$$O = \begin{pmatrix} N_1 \cdot \alpha_1^1 + (s+1) \cdot \alpha_1^2 & \dots & \dots & N_1 \cdot \alpha_m^1 + (s+1) \cdot \alpha_m^2 \end{pmatrix}$$

Notice that we have not added 1 unit to any component, but we have added α_j^2 to every component of the vector. Now let us see what happens if now T *forgets* the first observation he made, at stage, say, s_0. This means he will forget an observation that was made when S was in the first period of his strategy, i.e. S was using the set of weights $\{\alpha_1^1, ..., \alpha_m^1\}$ at that moment. Thus the probability that the action observed at step s_0 was a_1 is α_1^1, the probability that it was a_2 is α_2^1,... and the probability that it was a_m is α_m^1. Since we do not know what really happened at that step because we are modeling an expected payoff, we cannot subtract 1 to any concrete component of the observation vector. What we should do is *subtract the probability* that the action observed in that moment was each of the possible actions, just in an analog way that we added that probability to every component of the vector at the time when we considered a new observation. Thus the observation vector will look like

$$O = \Big(\; (N_1 - 1)\cdot \alpha_1^1 + (s+1)\cdot \alpha_1^2 \quad \ldots \quad \ldots \quad (N_1 - 1)\cdot \alpha_m^1 + (s+1)\cdot \alpha_m^2 \; \Big)$$

It is important to note that the weights that are subtracted correspond to those used by S at the moment of the observation that is being deleted. This means that it is necessary to compute which set of weights was being used by S at the moment of the observation. Function 4.11 was designed for this purpose, because it calculates what was in the observation vector at a certain step, no matter how many different periods were employed from the beginning to that step. If the capacity of the memory is limited, it is enough to delete everything that had been observed up to a certain step corresponding to the one before the oldest step that is still stored in memory. Thus the content of O_j after step d ($1 \leqslant d \leqslant L$) of the simulation with an observation memory limited to mem steps ($d > mem$) can be expressed as $OBS(d,j) - OBS(d-mem,j)$. The case where $d-mem \leqslant 0$ is nonsense but was added to the definition of function OBS to enable a general expression for the expected payoff. Obviously, $\forall j, OBS(d{-}mem, j) = 0$ when $d \leqslant mem$, meaning that T has observed nothing before the beginning of the simulation. Limited memory can be seen as a sliding window, as shown in Fig. 4.3.

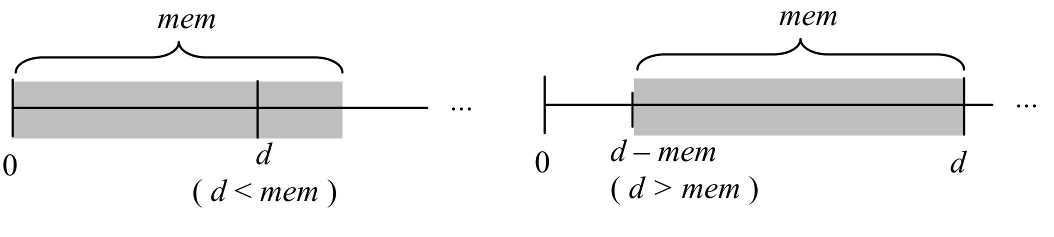


Figure 4.3: Depiction of limited memory in a temporal line at step s of the sequence in two cases: when d is smaller (left) and greater (right) than the capacity of the memory. In the former case, agent T still remembers everything from the beginning of the simulation but in the latter, he only remembers the mem last responses of S

Summarizing, expression 4.12 can be generalized to account for limited observation memory of mem steps as follows.

$$EP_{\lim}(d, mem) = \sum_{j=1}^{m} \alpha_j^1 (1 - 1/m) p_j +$$

$$\sum_{s=2}^{d} \sum_{j=1}^{m} \alpha_j^{H(s)} \left(1 - \frac{OBS(s-1, j) - OBS(s-1-mem, j)}{\min\{s-1, mem\}}\right) p_j \qquad (4.13)$$

4.4 Experiments and results

Recall that the experiments we have conducted are aimed at answering the following questions already posed at the beginning of the chapter:

1. Do the results obtained with the analytical expressions match those obtained by empirical simulations?

2. Do dynamic mixed strategies outperform a static mixed strategy in terms of expected payoff?

3. Does a limited memory have an impact over the expected payoff? Is it beneficial or detrimental for agent T to forget the oldest observations?

4.4.1 Experimental settings

The parameter configuration of the model instance that has been used in the empirical evaluation of strategies was the following:

- Number of different actions: $m = 5$.

- Number of encounters: $L = 500$.

- Payoff vectors: 15 different vectors were tested. The payoffs of every vector are summarized in Table 4.2.

Table 4.2: The 15 payoff vectors used in the experiments, separated in 3 groups according to the payoff structure, and the maximum total payoff attainable by S after 500 encounters

Vector	Payoffs					Maximum reward possible after 500 encounters
V_1	1	0.9	0.95	0.8	0.85	500
V_2	0.8	0.9	0.6	0.7	1	500
V_3	1	0.85	0.7	0.4	0.55	500
V_4	1	0.6	0.8	0.4	0.2	500
V_5	0.25	0.01	0.5	1	0.75	500
V_6	1.1	0.95	0.9	1.05	1	550
V_7	1.2	1	1.1	0.9	0.8	600
V_8	1.3	1	1.15	0.85	0.7	650
V_9	1.2	1.4	1	0.8	0.6	700
V_{10}	1.5	1	0.75	1.25	0.5	750
V_{11}	0.8	0.6	0.4	1.5	1	750
V_{12}	0.8	0.6	0.4	1.75	1	875
V_{13}	0.8	0.6	0.4	2	1	1000
V_{14}	0.8	0.6	0.4	2.25	1	1125
V_{15}	0.8	0.6	0.4	2.5	1	1250

Evaluation of a strategy. To compare empirical and theoretical results for every payoff vector, we did the following. For each payoff vector, the numerical values p_j are substituted in the theoretical expression as well as the number of periods H that is required in case of dynamic mixed strategies (recall that we tested H varying from 1 to 4), and the optimization algorithm is then executed. The algorithm returns the values of the optimal strategy for that payoff vector. Once we have such values defining the strategy, we evaluate it theoretically by substituting them in the theoretical expressions 4.2 (if it is a static strategy) or 4.6 and 4.13 (if it is dynamic), to obtain the expected payoff. In addition to this, we also use that strategy in 100 independent empirical simulations with the current payoff vector, and take the average of the executions to compare it with the theoretical expected payoff.

Optimization algorithm. The *NMaximize* command of the *Mathematica* software package was used to solve the constrained optimization problems of Eq. 4.3 and 4.7. The maximization it carries out is not analytical but employs approximate heuristic methods that depend on the initial solution. However, the same command was used for both the static and dynamic strategies so this phenomenon affects both families of strategies in the same way.

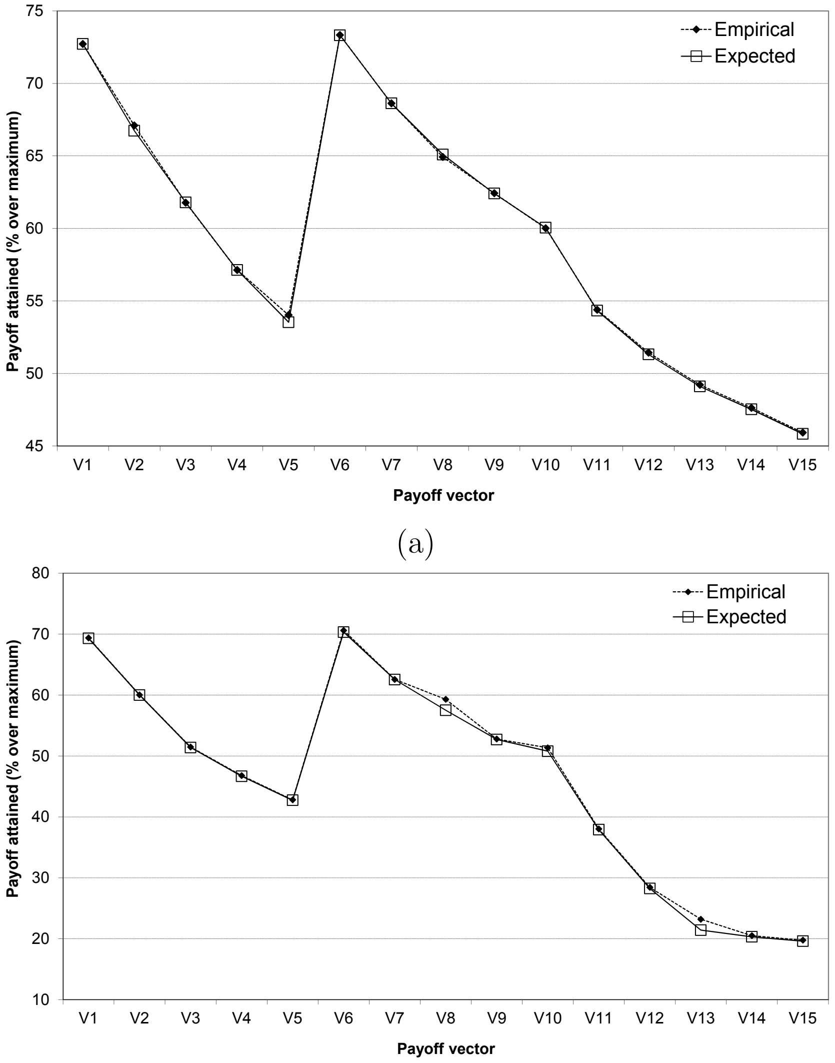
Recall that the number of periods H is not part of the optimization process but must be set by the user. In order to test the influence of such parameter, we tested H varying from 1 to 4 periods and compared the results. Insights on this are provided in Section 4.4.2. On the other hand, the optimal length N_h of every period is part of the set of variables to optimize.

4.4.2 Results

In order to answer the first question posed above, first note that the theoretical expressions have to be evaluated given the numerical values (period lengths and weights) that define the concrete strategy being tested. In our case, the strategies that we will evaluate are those obtained by the optimization process explained in previous sections.

Fig. 4.4(a) shows a comparison of the expected and empirical payoff for an optimal dynamic mixed strategy with 4 periods with unlimited memory (see expression 4.6). The plot confirms an almost perfect matching between the predicted and the actual payoff in all the payoff vectors. The payoff is presented as a percentage of the accumulated payoff over the *maximum*, which is the total payoff attainable if agent S always selects the action with the highest payoff and he is never guessed. This would be the ideal situation that only occurs when there is no adversary. Fig. 4.4(b) shows again a perfect matching in case of limited memory for T, so expression 4.13 has been proven to be correct as well.

We now analyze the performance of both static and dynamic mixed strategies with unlimited memory to answer the second question posed above. The results are shown in Fig. 4.5. This figure proves a very important result.



(a)

(b)

Figure 4.4: Expected and empirical payoff for S of dynamic mixed strategies with 4 periods and unlimited memory (a) and with limited memory of 30 steps (b)

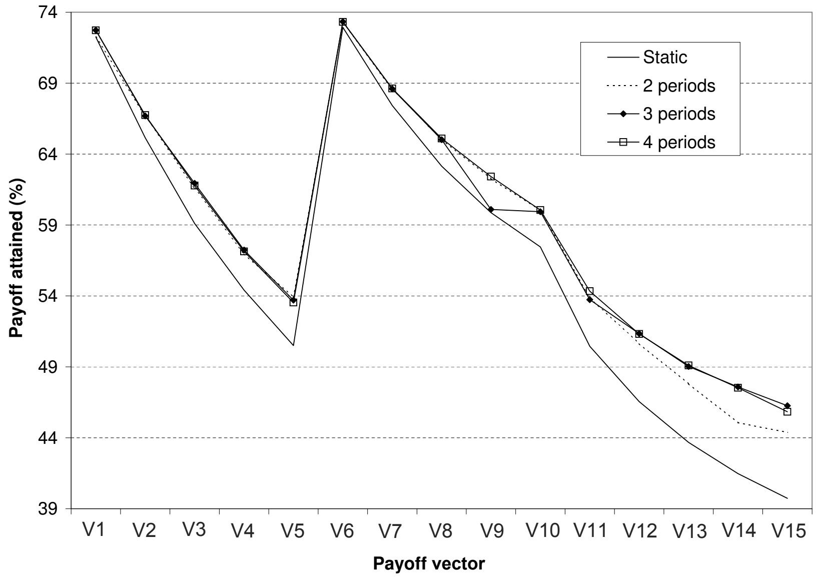


Figure 4.5: Expected payoff for S with unlimited memory for every payoff vector

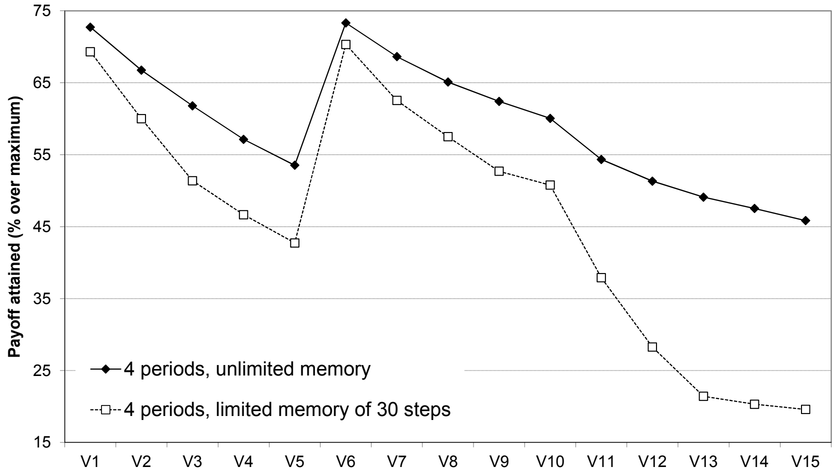


Figure 4.6: Expected payoff for S of the 4-period dynamic strategies originally designed under unlimited memory assumption, now being tested with both unlimited and limited memory for every payoff vector

In all the payoff vectors tested, the three optimal dynamic mixed strategies outperformed the optimal static strategy. In addition, increasing the number of periods was always beneficial in terms of the payoff attained. Recall that these results do not come from a simulation but from a prediction made using the expressions, so they are not influenced by random factors. Notice that the greater gain in performance was achieved in vectors V_{11} to V_{15}, which are those where the highest payoff is much greater than the rest. This is a particularly encouraging result for problems in which it is especially important to do the best action as many times as possible.

In order to answer the third question, the impact of limited memory over the expected payoff is going to be studied. As explained in Section 4.3, limited memory is a violation of the assumptions made for the optimized dynamic mixed strategies. In other words, the optimization process carried out was assuming some facts that are not true anymore, so the results obtained are not optimal now. Fig. 4.6 shows the expected payoff using the optimized dynamic strategies with 4 periods (which were the best-performing ones) both with unlimited and limited memory. Recall that all these strategies being tested now were originally designed assuming an unlimited-memory adversary so our aim now is to measure how the existing strategies are affected if the assumption is violated. According to Fig. 4.6, in all cases it was beneficial for T to have a limited memory as this allowed him to recover faster from deception and consequently the payoff of S was much lower. The case of payoff vectors V_{11} to V_{15} is dramatic. In these vectors, S achieved the greatest gain with respect to static strategies using 4-period dynamic strategies. But now that T has limited memory, however, S only attains 20 % of the maximum payoff in V_{13}, V_{14} and V_{15}, 30 % in V_{12} and less than 40 % in V_{11}.

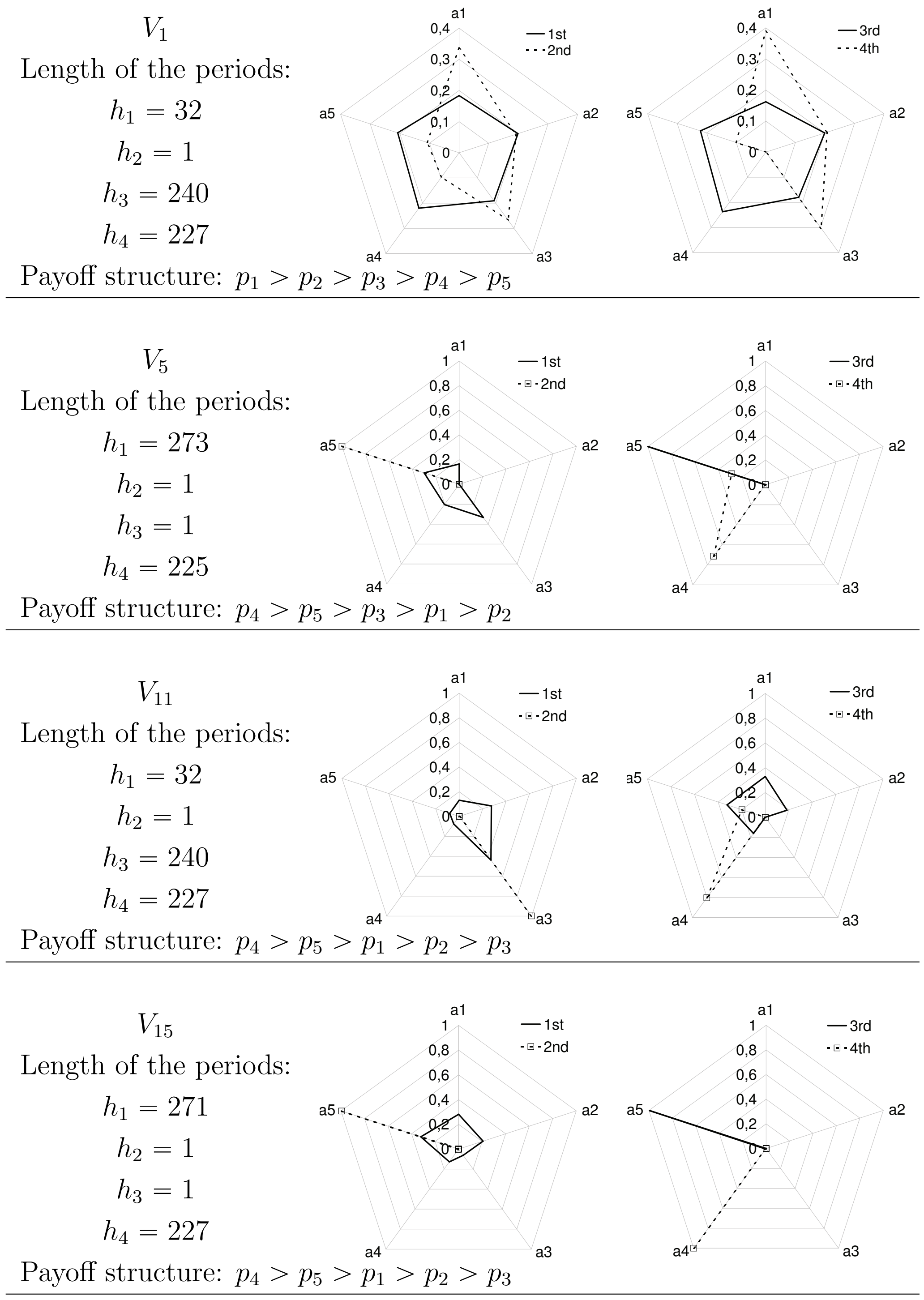
It is important to note that agent S could have designed better strategies if he knew in advance the length of the limited memory, i.e. if he had an accurate model of the adversary. In that case, he could exploit this knowledge by substituting the objective function of the optimization problem by the one with limited memory given in Section 4.3.1. As a result, S would not suffer such payoff loss, but we think that assuming such an accurate knowledge of T is too unrealistic as the information would be very asymmetric and favour

S even more than in the current model.

Strategies obtained with 4 periods. Examining the optimal dynamic strategies obtained is useful to understand how agent S tries to cause deception along time. Figure 4.7 shows four selected dynamic strategies with 4 periods. The mixed strategy used in each period is depicted in a 5-axis star-plot where the axes represent the $m = 5$ available actions of the model $a_1, ..., a_5$. The continuous and dashed polygons represent the probabilities of choosing each action during each of the periods. Since the dynamic strategies depicted have 4 periods, we have plotted the first 2 periods in one plot (on the left side) and the last 2 periods in another (on the right), to avoid excessive overlapping on one single plot that makes it difficult to read. Notice that in the last period, usually the best and second-best actions are the only ones with a probability greater than 0 of being chosen. However these choices are supposed to be safe because the rest of the actions were chosen many times during the first periods. This behaviour is emphasized in vectors in which the best action has associated a very prominent payoff in relation with the other actions, as happens in V_5 and V_{15}. In the last period the strategy just chooses the best action all the time, but since we assume unlimited memory, agent S will not be able to recover from all the previous observations and as a result, it is still safe to deterministically choose the best action during all the 4th period.

Insights into the behaviour with limited memory

In order to understand why limited memory is beneficial for T, we propose to study the probability of making a correct guess, and how it changes along time. This study must be done for one specific action since the evolution of this probability is different from one action to another. Recall that dynamic strategies are aimed at inducing confusion at the beginning to safely choose the best action after the initial periods. For this reason, the most interesting action to track this evolution is the best action (i.e. that with highest payoff) of each vector. Fig. 4.8 shows the evolution of the probability of being guessed correctly ($O_b / \sum_{j=1}^{m} O_j$) when choosing the best action a_b of payoff vectors V_1, V_{11} and V_{15}. Agent T employs a 4-period dynamic mixed strategy. The changes on the guess probability are more abrupt when the memory is



V_1

Length of the periods:

$h_1 = 32$

$h_2 = 1$

$h_3 = 240$

$h_4 = 227$

Payoff structure: $p_1 > p_2 > p_3 > p_4 > p_5$

V_5

Length of the periods:

$h_1 = 273$

$h_2 = 1$

$h_3 = 1$

$h_4 = 225$

Payoff structure: $p_4 > p_5 > p_3 > p_1 > p_2$

V_{11}

Length of the periods:

$h_1 = 32$

$h_2 = 1$

$h_3 = 240$

$h_4 = 227$

Payoff structure: $p_4 > p_5 > p_1 > p_2 > p_3$

V_{15}

Length of the periods:

$h_1 = 271$

$h_2 = 1$

$h_3 = 1$

$h_4 = 227$

Payoff structure: $p_4 > p_5 > p_1 > p_2 > p_3$

Figure 4.7: Depiction of the optimal 4-period strategies obtained for vectors V_1, V_5, V_{11} and V_{15} after applying optimization, assuming unlimited observation memory of T. Each p_i stands for the payoff of action a_i.

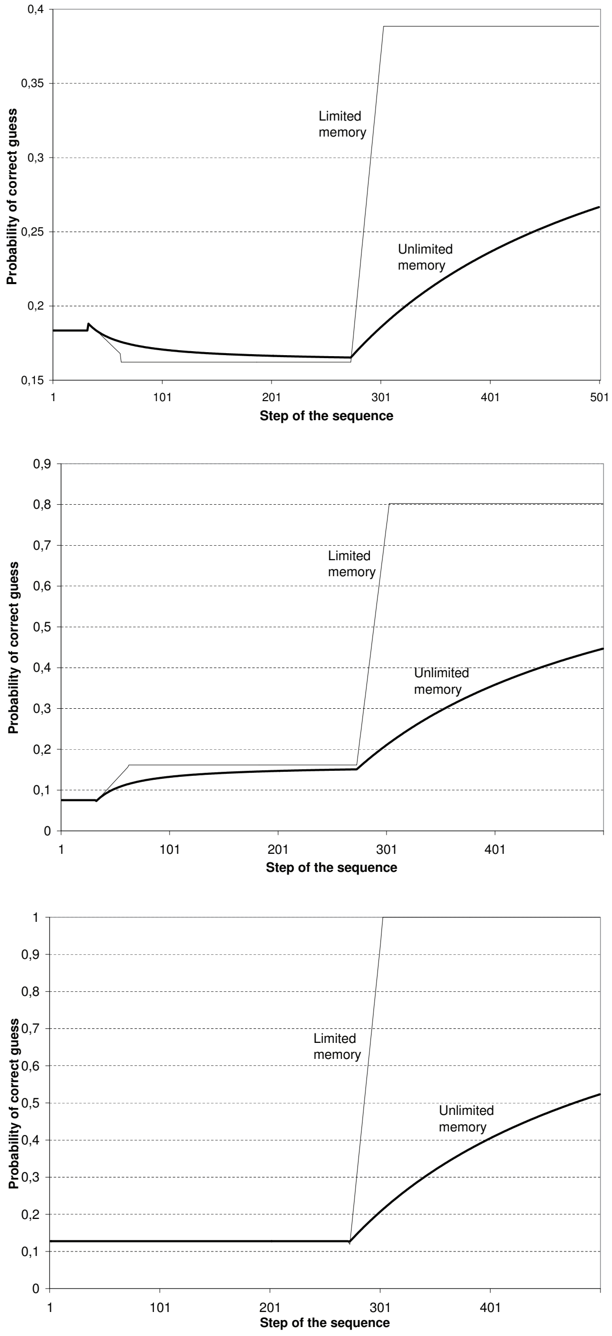


Figure 4.8: Evolution of the probability that T successfully matches the action chosen by S if S chooses the action with highest payoff of each vector. S is using the optimal 4-period dynamic strategy obtained for payoff vectors V_1 (top), V_{11} (middle), and V_{15} (bottom)

limited because the observation vector starts to display the new behaviour very quickly after a change of S. As a consequence, T is able to adapt to the new situation faster and the deception that S tried does not work very well. The cases of vector V_{15} is specially dramatic. As it was very important to choose the best action as many times as possible because it has a remarkably high payoff, S had planned a first period with several different bad choices, and then switching to a second period with a single repeated best choice. Now T quickly notices this change and when S starts choosing always the best action in the second period, T learns this behaviour and starts being successful in his predictions from a very early stage of this second period.

Since limited memory for T seems to be very harmful for S in relation to unlimited memory, it may be interesting as well to find out what the exact impact of memory capacity over the expected payoff is. In the extreme case, with a memory of only one step of capacity, T would always make his election based on the last action observed. Of course, if S were aware of this, it would be very easy for him to exploit this behaviour to his own benefit, but for now we will assume S does not know. We will study the impact in relation to the payoff vector used in order to discover if limited memory affects some vectors more than others. Figure 4.9 shows the results. The lines represent the theoretically expected payoff for a given capacity value after 500 encounters. As can be seen in the plot, we have tested the capacity of the observation vector varying from 1 to 500. A capacity of 500 in a 500-step setting is equivalent to infinite memory.

First of all, the three plots confirm that the shorter the memory, the worse for S or equivalently the better for T. As expected, the payoff vector has a clear influence on the impact of limited memory. For instance in vectors V_1 to V_5 as well as V_6 to V_{10}, the slope of the curves 4 and 5 as well as 9 and 10 is greater than the others, because these vectors are "more difficult" than the rest. By *difficult* we mean that the payoffs of the vector are very different among them, so every time a good action is correctly guessed by T, it leads S to a great loss in payoff. In these cases it is specially important to choose good actions to avoid great losses. In vectors V_{11} to V_{15} one of the payoffs is much greater than the rest as explained in preceding sections. All the curves of that group have big slopes meaning that the impact of the capacity of the

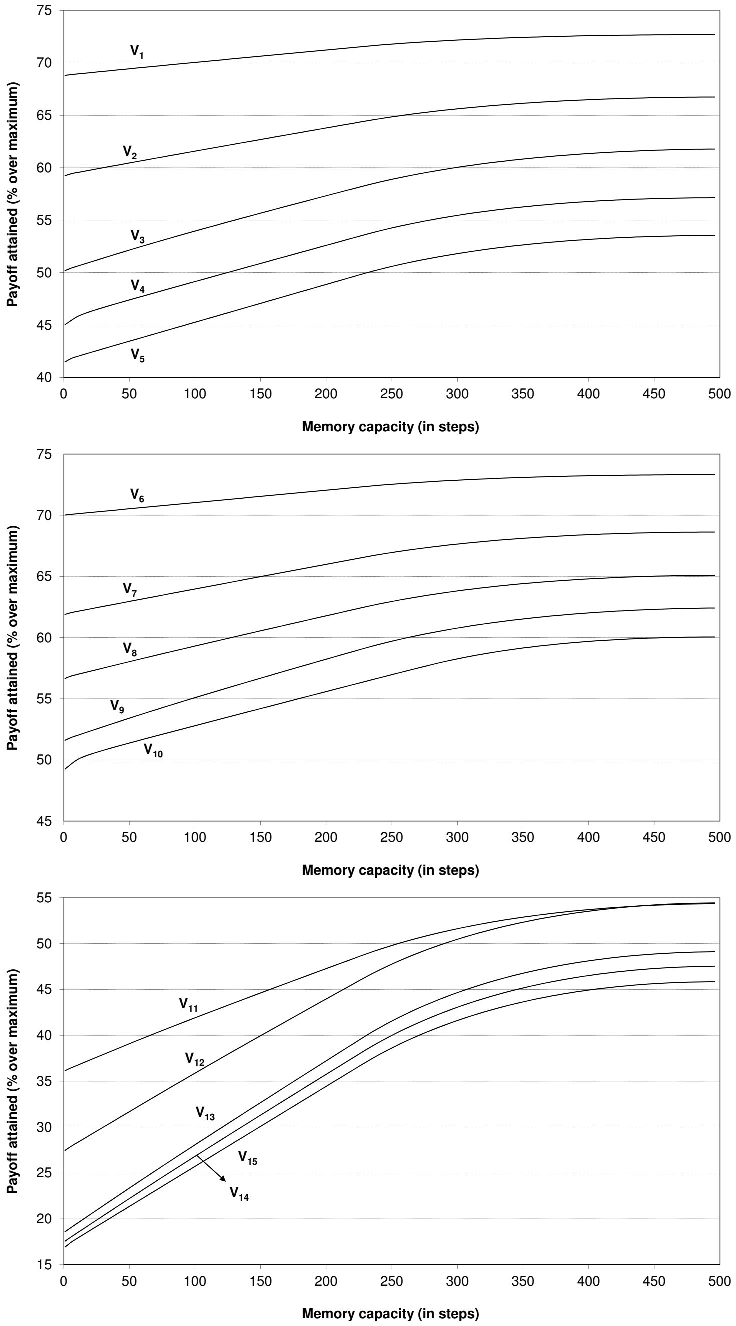


Figure 4.9: Expected payoff for S as a function of the capacity of the observation memory of T for every payoff vector. Agent S is using the optimal 4-period dynamic strategy originally designed under the assumption of unlimited adversarial memory

memory is greater. Secondly, all the curves tend to present big slopes at the beginning but turn horizontal after a certain value of capacity. This value can be thought of as a *threshold*. Above this value, it is almost impossible for T to recover from the past observations and adapt to a new situation (strategy), which means that S has been successful in his deception.

On the influence of the number of periods. We have done all the experiments so far with 4-period dynamic strategies. The value of the number of periods originates a different optimization problem in each case, increasing the number of variables in $m + 1$ as the number of periods increases in one unit. For this reason, this value should not be too high to keep the size of the optimization problem in reasonable limits. Further, it should be in accordance with the number of encounters L that are going to take place, since too short periods have a low impact on the performance of the overall strategy, and the corresponding set of weights cannot modulate properly the overall behaviour due to lack of rounds.

We have sampled H from 2 to 30 by running the optimization process for each H value, for adversaries with memory capacity of 30, 100, 200 and 500 (unlimited memory) steps. The results are shown in Fig. 4.10 and prove that the impact of dynamic strategies is different depending on the adaptive capacity of the adversary, with a shorter memory representing a more adaptive opponent. In case of unlimited memory, the best performance is achieved with 4 periods, and increasing the number of periods does not give S a higher payoff (the corresponding line in the plots becomes horizontal very quickly). This is because unlimited memory allows for greater space for deception and thus it is enough to have less periods. In case the observation memory is limited, then it becomes more important to employ a strategy with a high number of periods, because the deception achieved when S switches from one period to another is quickly attenuated when oldest observations are forgotten. In other words, agent T can quickly adapt and learn the new behaviour. As a result, the effect of switching to a different set of weights becomes useless after a very short time, and therefore another change in the weights used by S is needed soon after the previous one. It can be seen in the plots that in all cases (except for 30-step memory capacity), a threshold

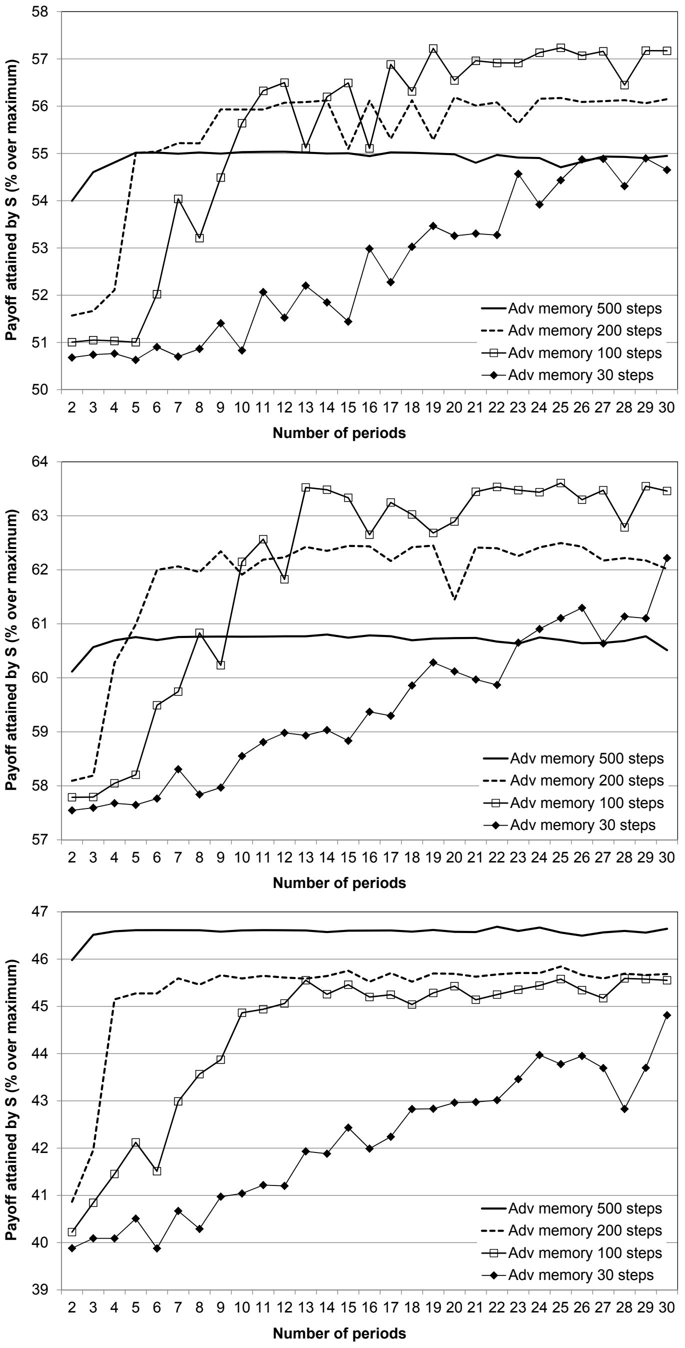


Figure 4.10: Expected payoff for S for payoff vectors V_5 (top), V_{10} (middle) and V_{15} (bottom), as a function of the number of periods of the optimized dynamic strategy he employs. Strategies were found assuming the memory capacity of T was known before the optimization process starts.

on the number of periods exists that gives the maximum performance, and increasing the number of periods beyond the threshold does not clearly improve the payoff. This behaviour mirrors the results of unlimited memory, although the threshold value depends on the payoff vector and the memory capacity of the adversary. If the number of periods is high enough, then it is possible to exploit the limited memory to achieve a greater payoff than with unlimited memory, as shown in payoff vector V_{10}, but this also depends on the payoff vector of the problem.

Study of the competitive ratio against a clairvoyant adversary

An interest kind of analysis consists in comparing the payoff attained by S using our strategies with that attained in the mixed equilibrium situation, even though such equilibrium to mixed strategies cannot be played since T does not know S's payoff. The aim is to compute the *competitive ratio*, which was defined originally for online tasks, i.e. those consisting in satisfying a sequence of requests that arise one at a time, without knowledge of future requests [54, 82]. It can be computed as the performance ratio of the algorithm that does not know in advance which demands will be received, divided by the payoff attained with an optimal, clairvoyant algorithm which knows in advance all the requests and their ordering.

Our imitation problem is not exactly an on-line task because there is nothing new at each repeated encounter that is unknown for any of the agents before that turn. However, it is true that agent T does not have full knowledge of the situation because he does not have access to S's payoffs. Therefore, we can compute the competitive ratio by considering the situation in which agent T knows S's payoffs, and thus he is able to compute and play the mixed strategy prescribed by mixed Nash equilibrium.

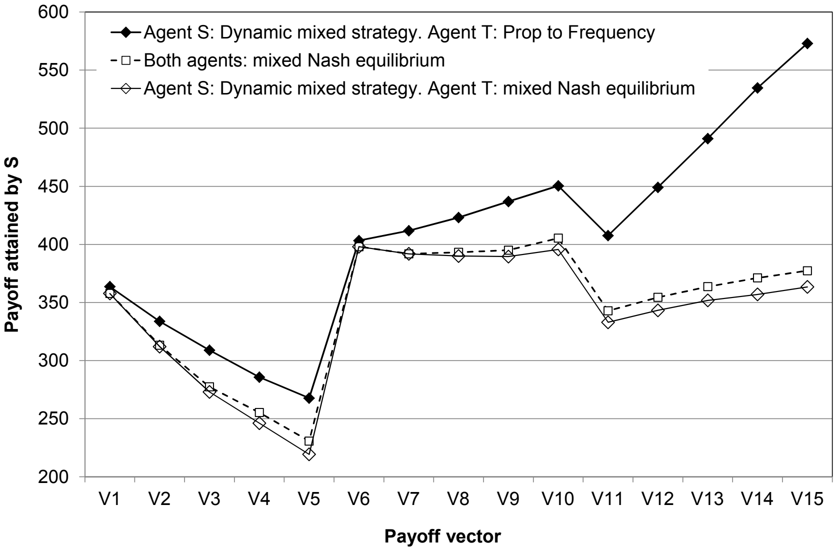
By definition of Nash equilibrium [64], in case T plays the mixed strategy prescribed by the Nash equilibrium, the best S can do is to do the same and play his own mixed equilibrium strategy too. Any other strategy would be worse for S. Therefore, the interesting situation is to assess (a) the payoff attained by S when he uses a dynamic mixed strategy that was designed assuming PF for T and T plays the mixed Nash equilibrium strategy, with respect to (b) the payoff attained by S when both agents play Nash equilibrium (which is the most clairvoyant situation because both agents know the adversary's payoffs and also know that the adversary knows their own payoff, so they both have a motivation to play Nash equilibrium). As stated before, if T plays Nash equilibrium, we know for sure that S's payoff when playing any strategy different than the Nash equilibrium (in particular, when playing a dynamic mixed strategy) is smaller than the payoff when playing S's own Nash equilibrium. The aim is to assess how big this difference can be, in ratio.

Such experiment has been conducted assuming a 4-period dynamic mixed strategy for S that was found by the optimization algorithm under the assumption that T was using PF. The strategy found was then played by S against an adversary using the mixed Nash equilibrium. The one-shot mixed Nash equilibrium strategy for T was computed using the Gambit software tool [1]. The results are depicted in Fig. 4.11. It can be seen in the figure that the competitive ratio, as defined in Fig. 4.11, is very close to 1 in all cases. This means that a dynamic mixed strategy designed under the assumption of and unlimited-memory PF adversary is not seriously damaged if the adversary actually plays the one-shot mixed Nash equilibrium at every encounter, instead of PF as he was assumed to play. Here, *not seriously damaged* refers to S's payoff in relation to the payoff S could have attained if he also plays Nash equilibrium instead of a 4-period dynamic mixed strategy. Of course, any of these payoffs are notably lower than the payoff S could attain if T actually played PF (see the top-most series of Fig. 4.11(a)), but this situation is not the one considered for the competitive ratio.

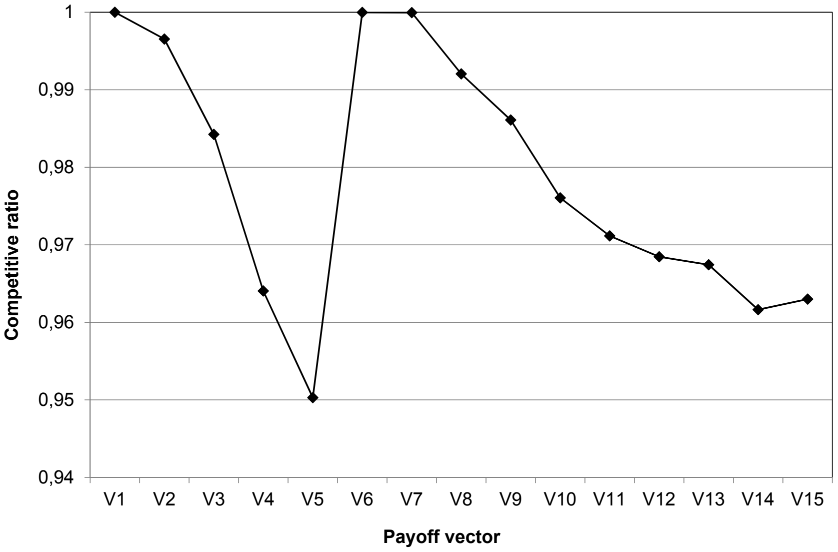
4.5 Conclusions

Static and dynamic mixed strategies for an agent in an adversarial model have been successfully designed using numerical optimization methods. Analytical expressions of the expected payoff for both strategies have been provided and validated also from an empirical point of view. Furthermore, optimal

[1] It is freely available at www.gambit-project.org



(a)



(b)

Figure 4.11: Payoff attained by S when playing against different kind of adversaries (a) and competitive ratio in the case that S plays 4-period dynamic strategies designed originally for an unlimited-memory adversary playing PF, and T is assumed clairvoyant and plays mixed Nash equilibrium (b). Note that (b) is the quotient of the two bottom lines of (a)

dynamic mixed strategies have shown to outperform optimal static mixed strategies in all the scenarios tested, specially when the difference between the payoff of the best action and the payoff of the rest of actions becomes greater.

However, such conclusions are valid only when we accept some assumptions about the behaviour of the adversary, agent T. These assumptions include that (a) the adversary has an unlimited observation memory (so an agent that knows he is being watched can manipulate the observer agent with his own behaviour), and (b) T uses a strategy proportional to the observed frequency to make his predictions, which allows S to previously take this into account when designing his strategies. If these conditions are not met, i.e. our model of the opponent is not accurate, then the strategies previously designed are not optimal any more, i.e. if T has a limited observation memory, then the manipulation that S had planned has no effect, thus leading to important (sometimes dramatic) losses in the payoff attained by S. Again, theoretical expressions of the expected payoff were provided and successfully contrasted with empirical simulations for this new situation. When T simply guesses with the last action observed (1-step memory), the results were very bad for S because in a mixed strategy a player often repeats probabilistically the same action in consecutive turns. Notice that one easy way to avoid this behaviour is to use the so-called *Random among K best actions* strategy, which was one of the very first strategies presented in [67, 92].

Chapter 5

Considering statistical dependence between actions and events

In chapter 3 it was shown how theoretical expressions can predict the expected payoff attained by one of the agents when using certain simple strategies, while in chapter 4, a simplified version of the original model of Fig. 2.1 (in which the external events were removed) was analyzed to find a randomized strategy that changed along the time for one of the agents. The main difference with the model we will develop here is that, in the aforementioned chapters, the outcome of an encounter did not affect the payoff attainable at the next encounter. This condition will be modified now.

The aim of this chapter is twofold. First, we present an extension of the original model in which statistical dependence is introduced between the action taken by agent S and the next event to arise. This is an important issue the agents should take into account before making a decision, because the action selected will have an influence over the next event to arise at the next encounter, and such event determines the maximum payoff attainable when the agents choose their actions. Second, decision strategies for agent S that are not constant along the time, but change at certain time steps in the iterated process, are proposed for this extended model. More specifically, we tackle the strategy design as a constrained non-linear optimization problem whose solution gives both the exact moment at which agent S must switch its strategy, and which strategy it must use. The work departs from the results shown in [91, 95] where such idea was developed on a model with no statistical dependence between the events and the actions.

In connection with the aforementioned objectives, the experiments we will conduct are aimed at answering the following questions:

1. Is there any substantial difference between the theoretical expected payoff and the average payoff attained by empirical simulations in the extended model?

2. Do dynamic mixed strategies outperform a static mixed strategy in terms of expected payoff in the extended model?

3. How are dynamic strategies affected by the number of different periods employed?

The remainder of the chapter is organized as follows. Section 5.1 describes the main characteristics and components of the model used, including the novel mechanism of statistical dependence between an action and the next input. Section 5.2 deals with the need of randomized strategies, both static and dynamic. The analytical expression of the expected payoff attained by S when using such both kinds of strategies is given and explained in detail, and the need of an optimization process for determining the best parameters in this expression is motivated. In Section 5.3 we describe the computational experiments performed and the results obtained. Finally, conclusions and further work are discussed in Section 5.4.

5.1 A model with statistical dependence

Recall that the model described in Fig. 2.1 had several types of events or stimuli. They were issued by the external environment (represented as another agent R). As described by the authors of the original model in [67], these events were *independent* and *randomly generated* following a *uniform* probability distribution. Independence means there is no relation between the current event, the current response and the next event. Uniformly generated events means that each time an event is going to be generated, all of them are equally probable. Notice that considering other probability distributions different than the uniform would not change the methodology of the existing works about this model ([92, 91, 95] and the two preceding chapters). It just would require to repeat the experiments and adjust the results, but the conclusions of such works do not depend on the probability distribution employed because, in the model considered so far, the agents' decisions have no influence on the forthcoming events and therefore, the agents do not need to include any information about the event probabilities in their decision strategies. Up to now, the probability distribution of the events has been considered an external, inalterable constraint.

However, extending the model by introducing statistical dependence between the events of consecutive stages is a different matter. We can think of several kinds of dependence. The first one is dependence between the event of one stage and that of the next stage. The question is, is this information useful for an agent in any way? In other words: before giving a response to the current event, is it relevant for the decision maker to know (in a probabilistic sense) which event will arise next? The answer is no, because every time an event arises, the agent should try his best, disregarding the next event since it will not be affected by the current decision. With this kind of dependence, we would be in the same setting mentioned above, i.e. it would be equivalent to considering no dependence at all from the agents' point of view.

The second type of dependence is that between the action taken in the current stage by one or both agents, and the next event to arise. That is why an arrow labeled a_j (action taken by S) goes from the agents back into the environment (agent R) in Fig. 5.1, which represents the new version of the model. This setting will be analyzed in the remainder of this chapter as it has interesting implications. The most important is that, before making a decision for the current event, an agent should consider that her choice will affect not only the immediate payoff of the current stage but also the maximum payoff attainable in the next stage, since not all the events offer the same range of payoffs. Such a consideration captures the fact that some events may be rare or *critical*, so they provide very high payoffs when the

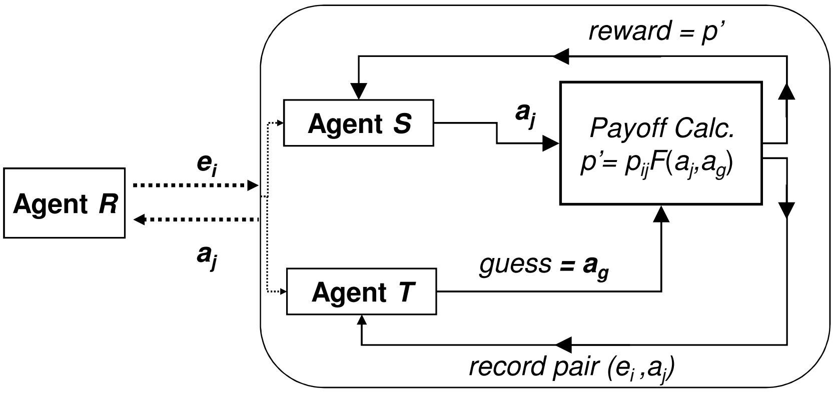


Figure 5.1: An imitation model that incorporates statistical dependence between S's actions and the next event to arise.

response is chosen properly, while some others are less important so they provide very low payoffs regardless the action chosen. With this in mind, now a *good* action is not just one that provides a high payoff for current event (immediate payoff) but also causes events with very high rewards[1] to be more likely to arise at the next stage.

In our model, we introduce dependence only between the action chosen by agent S and the next event to arise. We assume that the information of such stochastic dependence is available only to agent S, in the form of a *conditional probability matrix* C with dimensions m x n shown below.

$$C(m \times n) = \begin{pmatrix} P[X = e_1 | Y = a_1] & \dots & P[X = e_n | Y = a_1] \\ P[X = e_1 | Y = a_2] & \dots & P[X = e_n | Y = a_2] \\ \vdots & \ddots & \vdots \\ P[X = e_1 | Y = a_m] & \dots & P[X = e_n | Y = a_m] \end{pmatrix}$$

The value C_{ij} stands for the conditional probability $P[X = e_j | Y = a_i]$, so $\sum_j P[X = e_j | Y = a_i] = 1$ for every row $i = 1, ..., m$. In this expression, X is the discrete random variable representing the next event arising, and Y is the discrete random variable representing the current action taken by agent S that, as will be explained, is based on a randomized behaviour rule. Finally, let $(\pi_1, ..., \pi_n)$ be the probabilities of each event to arise at the first step of the simulation. We cannot give conditional probabilities in this case as there

[1] Those for which the values of the corresponding row of the payoff matrix are all large.

is no previous action. For our experiments, we will take such probabilities as uniform, $\pi_i = 1/n$ for $i = 1, ..., n$.

5.2 Behaviour of the agents

The strategy T will use in this chapter is called Proportional to Frequency (PF) and has been described in section 2.4.2. In the next subsections, we provide alternatives for S's strategy.

5.2.1 Static mixed strategy for S under statistical dependence

As in section 4.2.1, agent S may use a mixed strategy, which is a probability distribution (also called a set of weights) over the actions. Again, we are interested in computing the best randomization, which yields the largest expected payoff for S when the randomization is employed to play against the PF strategy of agent T. In order to compute S's expected payoff corresponding to a given set of weights, it is necessary to calculate the probability that agent S eventually gets each of the possible payoffs of matrix P. This process is slightly trickier than the one of section 4.2.1. This probability can be computed as the product of the probabilities of several independent events happening simultaneously. Agent S will attain payoff p_{ij} if three conditions hold:

(i) Event e_i must arise. We will refer to this probability as $P[I = e_i]$. In this case, I is the discrete random variable representing the occurrence of each event at *current* stage.

(ii) Assuming that event e_i has arisen, agent S must select action a_j as a response. We will note this probability (also called *weight*) by α_{ij} but can be formally written as $P[Y = a_j | I = e_i]$.

(iii) Finally, S will only get the score p_{ij} if agent T does not successfully predict her response.

The probabilities α_{ij} involved in condition (ii) constitute the mixed strategy we are searching for. The calculation of the probability of conditions (i) and (iii) is described below.

Condition (i): marginal probability of an event. first of all, recall that we had previously defined another random variable X that represents the occurrence of each event at the *next* stage. When the simulation advances one step, then $P[I = e_i]$ adopts the value of $P[X = e_i]$ computed in the preceding step, for all $i = 1, ..., n$. Notice that we are not given the probability distribution of I so the values $P[I = e_i]$ have to be computed using the known conditional probabilities of matrix C and the mixed strategy of S mentioned in (ii).

Actually the only data we have about the marginal probabilities $P[I = e_i]$ of the ocurrence of the external events are those of the *first* step of the game, $P_1[I = e_i] = \pi_i$. Recall that the decisions of agent S influence the marginal probabilites, and such decisions are modeled by the mixed strategy $(\alpha_{ij}, j = 1, ..., m)$ it employs for each event e_i. If we apply the *Theorem of total probability* to the first step, we have

$$\begin{aligned} P_1[Y = a_j] &= \sum_i P[Y = a_j | I = e_i] P_1[I = e_i] \\ &= \sum_i \alpha_{ij}\pi_i, \quad j = 1, ..., m \end{aligned} \tag{5.1}$$

The sub-index of P indicates the step to which it is referred. Once those values are known, they can be substituted in the following expression, which results from applying again the Theorem of total probability since $[Y = a_j], j = 1, ..., m$ constitute another partition of the sample space:

$$\begin{aligned} P_1[X = e_i] &= \sum_j P[X = e_i | Y = a_j] P_1[Y = a_j] \\ &= \sum_j C_{ji} P_1[Y = a_j], \quad i = 1, ..., n \end{aligned} \tag{5.2}$$

Now, the values $P_1[X = e_i], \quad i = 1, ..., n$ can be taken as the values $P_2[I = e_i]$. The probability that an event arises at the next step when *current* step is number 1 is equivalent to the probability that an event arises at current step if current step is number 2. Then, since the values $P_2[I = e_i], \quad i = 1, ..., n$ are known, they can be used to compute the same probabilities at step 2, applying the same expressions indicated above. In general, the following equations can be applied recursively for $k \geqslant 2$:

$$\begin{aligned} P_k[Y = a_j] &= \sum_i P[Y = a_j | I = e_i] P_k[I = e_i] \qquad (5.3)\\ &= \sum_i \alpha_{ij} P_{k-1}[X = e_i], \quad j = 1, ..., m \\ P_k[X = e_i] &= \sum_j P[X = e_i | Y = a_j] P_k[Y = a_j] \\ &= \sum_j C_{ji} P_k[Y = a_j], \quad i = 1, ..., n \qquad (5.4)\\ P_k[I = e_i] &= P_{k-1}[X = e_i] \end{aligned}$$

Condition (iii): probability of not being guessed. Assume agent S is using a mixed strategy so that it uses α_{ij} to select the action a_j with payoff p_{ij}. Assume agent T uses strategy PF, and let us suppose that a certain event e_i has arisen L_i times during the repeated game. Then action a_j will have been selected $L_i \cdot \alpha_{ij}$ times, and this number is what agent T has recorded in O_{ij}. The probability that T selects action a_j as a prediction using PF is then

$$P_{guess_{ij}} = \frac{O_{ij}}{\sum_{j=1}^{m} O_{ij}} = \frac{L_i \cdot \alpha_{ij}}{L_i} = \alpha_{ij} \qquad (5.5)$$

with m being the number of actions available. Actually the value L_i is unknown as it depends on the sequence of decisions made by S, but in the last expression it is simplified and eventually disappears. Therefore the probability of not being guessed correctly is $1 - P_{guess} = 1 - \alpha_{ij}$.

Expected payoff with static mixed strategies. Using the probabilities of the three conditions described above, the expected payoff for agent S after a sequence of L inputs when it uses weights $\alpha = (\alpha_{ij})$ to select her actions has the following expression:

$$EP_{static}(\alpha) = \sum_{k=1}^{L} \sum_{i=1}^{n} P_k[I = e_i] \cdot \sum_{j=1}^{m} \alpha_{ij} \cdot (1 - \alpha_{ij}) \cdot p_{ij} \qquad (5.6)$$

Notice that the probability that each event arises evolves along time because it also depends on the choices made by agent S.

Optimal strategy maximizing the expected payoff. If we want to maximize the expected payoff, we have to maximize expression 5.6 by computing the values of the optimal probabilities α_{ij}. This leads to an optimization problem with m x n variables since agent S uses a different mixed strategy for each event (each strategy having m variables) and there are n different events. It can be expressed as:

$$\max_{\{\alpha_{ij}\}} \left\{ \sum_{k=1}^{L} \sum_{i=1}^{n} \left(P_k[I = e_i] \cdot \sum_{j=1}^{m} \alpha_{ij} \cdot (1 - \alpha_{ij}) \cdot p_{ij} \right) \right\} \tag{5.7}$$

subject to:

$$\sum_{j=1}^{m} \alpha_{ij} = 1 \quad i = 1, ..., n$$
$$\alpha_{ij} \geqslant 0 \quad i = 1, ..., n; \quad j = 1, ..., m \tag{5.8}$$

Recall that agent S, who is interested in solving the problem to get the maximum reward, must know in advance the number of repetitions or steps L that the game will have, and should be aware that T will use PF as well. If the value of L is unknown, the agent will have to estimate it somehow. The optimization problem should include all the terms of expression 5.6 that depend on the values α_{ij}. Since the probabilities P_k also depend on them, they must be part of the target function to be maximized.

5.2.2 Dynamic mixed strategy for S under statistical dependence

We now introduce dynamic mixed strategies for the extended model with statistical dependence, following the same idea of section 4.2.2. Recall that, in a dynamic mixed strategy, the weights change along the time at some concrete time instants. The expression of the expected payoff when using dynamic strategies in the extended model is a bit different from that obtained in the previous chapter, but is based on the same approach. As in section 4.2.2, we make use of the concept of period length, which is the number of events during which agent S uses the same mixed strategy. In order to gain flexibility and achieve a higher reward, we can define a different number of

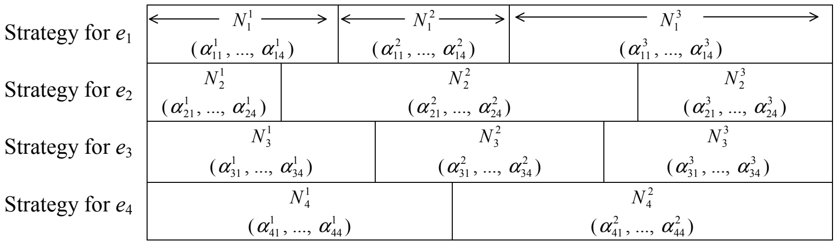

Strategy for e_1
N_1^1
$(\alpha_{11}^1, ..., \alpha_{14}^1)$
N_1^2
$(\alpha_{11}^2, ..., \alpha_{14}^2)$
N_1^3
$(\alpha_{11}^3, ..., \alpha_{14}^3)$
Strategy for e_2
N_2^1
$(\alpha_{21}^1, ..., \alpha_{24}^1)$
N_2^2
$(\alpha_{21}^2, ..., \alpha_{24}^2)$
N_2^3
$(\alpha_{21}^3, ..., \alpha_{24}^3)$
Strategy for e_3
N_3^1
$(\alpha_{31}^1, ..., \alpha_{34}^1)$
N_3^2
$(\alpha_{31}^2, ..., \alpha_{34}^2)$
N_3^3
$(\alpha_{31}^3, ..., \alpha_{34}^3)$
Strategy for e_4
N_4^1
$(\alpha_{41}^1, ..., \alpha_{44}^1)$
N_4^2
$(\alpha_{41}^2, ..., \alpha_{44}^2)$

Figure 5.2: Example of different periods for each event in a model instance with 4 different events. The letters inside each rectangle represent the length of that period and the mixed strategy to be used in it.

periods of different length for each event. We will call N_i^h the length of the h-th period of event e_i.

The next example illustrates this concept. Suppose that we have an input sequence of length L and that a given event e_i is expected to arise $L_i = 100$ times along the sequence. Then, we can define for instance 4 periods of lengths $N_i^1 = 30$, $N_i^2 = 10$, $N_i^3 = 20$ and $N_i^4 = 40$. For a given period, the set of optimal weights is different from that of other periods because the distribution of the payoffs in a row of the payoff matrix may differ a lot from other rows. Fig. 5.2 shows another example of different dynamic mixed strategies for each event, with different number of periods and/or different moments of change. The length of the whole input sequence is $L = 1000$. Suppose there exist $n = 4$ different kinds of inputs in our model and $m = 4$ different actions, so a mixed strategy is a vector of 4 weights. If the inputs are uniformly distributed, then each event is expected to arise about 250 times, so the sum of the lengths of the periods for any event should be 250.

In order to calculate the best randomization under this scenario, we need to obtain the expression of the expected payoff for a dynamic strategy. The basics to obtain the expected payoff are again premises (i), (ii) and (iii) stated in Section 5.2.1, as explained next. Some new issues have to be taken into account to address the dynamic situation with statistical dependence.

Estimation of the number of occurrences of an event. Before computing the probabilities of conditions (i) and (iii), recall that in the dynamic case the number L_i of occurrences of an event is unknown and depends on the sequence of decisions made by S. We are able to compute only the marginal probability $P_k[I = e_i]$ of each event at step k. However, that information, together with the total number of steps L of the game, is enough to estimate L_i, since $L_i = \sum_{k=1}^{L} P_k[I = e_i]$. In general, let L_i^k be the estimated number of times event e_i has arisen after k steps of the game, which can be computed as

$$L_i^k = \sum_{t=1}^{k} P_t[I = e_i] \tag{5.9}$$

Since this is an estimation based on probabilities, it is expected to be a real (not integer) value. However, this number will be used as a summation limit in the next section, so in that case it has to be rounded to the nearest integer.

Condition (i): marginal probability of an event. From now on, we will focus only on one single event e_i. Let $(\alpha_{ij}^h)_{j=1,...,m}$ be the set of weights agent S uses to choose an action as a response to an input of type e_i during the h-th period. Then, within a given period, α_{ij}^h represents the probability that S selects action a_j when event e_i has already arisen, also expressed as $P^h[Y = a_j | I = e_i]$. The novel part is that the values $P^h[Y = a_j | I = e_i]$ are different from one period h_1 to another h_2, but the calculation method is the same described in Eq. 5.3 and 5.4. In this case, we must be careful to substitute the adequate values of $P^h[Y = a_j | I = e_i]$ according to the period h to which step k (of the $P_k[Y = a_j]$ being computed) belongs. In a more formal way, Eq. 5.3 can be rewritten to consider a distinct mixed strategy for each period, as follows:

$$P_k[Y = a_j] = \sum_i P^{H_i(k)}[Y = a_j | I = e_i] P_k[I = e_i]$$

$$= \sum_i \alpha_{ij}^{H_i(k)} P_{k-1}[X = e_i], \quad j = 1, ..., m \tag{5.10}$$

Function H_i maps an absolute step $k : 1 \leqslant k \leqslant L$ to the period h whose set of weights $(\alpha_{ij}^h)_{j=1,...,m}$ must be used to issue a response within the dynamic strategy for event e_i. It is based on the estimation of the number of times that e_i has arisen after $k-1$ steps, as follows:

$$H_i(k) = \begin{cases} 1 \text{ if } & L_i^{k-1} < N_i^1 \\ 2 \text{ if } & N_i^1 \leqslant L_i^{k-1} < N_i^1 + N_i^2 \\ 3 \text{ if } & N_i^1 + N_i^2 \leqslant L_i^{k-1} < N_i^1 + N_i^2 + N_i^3 \\ \ldots \end{cases}$$

The above changes only affect the computation of the estimated number of times each event arises along the simulation, see (5.9).

Condition (iii): probability of not being guessed. This is the only part that remains unchanged from our results in [91, 95] so the explanations of this section can also be found in those works. The difficult part of the expression we need involves the computation of the probability of not being guessed. After the first period of length, say, N_i^1, the observation matrix O has the following values in row i (representing absolute frequencies of the responses given in the past by S to inputs of type e_i):

$$T(n \times m) = \begin{pmatrix} \ldots & \ldots & \ldots & \ldots \\ N_i^1 \cdot \alpha_{i1}^1 & \ldots & \ldots & N_i^1 \cdot \alpha_{im}^1 \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}$$

The probability of not being guessed P_{NG} during the first period is (1-α_{ij}^1), according to the same explanation given in section 5.2.1. This reasoning becomes more complicated when considering row i of the observation matrix at the end of the second period, whose length is N_i^2:

$$T(n \times m) = \begin{pmatrix} \ldots & \ldots & \ldots & \ldots \\ N_i^1 \cdot \alpha_{i1}^1 + N_i^2 \cdot \alpha_{i1}^2 & \ldots & \ldots & N_i^1 \cdot \alpha_{im}^1 + N_i^2 \cdot \alpha_{im}^2 \\ \ldots & \ldots & \ldots & \ldots \end{pmatrix}$$

According to the values of the former matrix after 2 periods, the probability *at the end of the second period* that agent T selects action a_j as a prediction is

$$P_{guess_{ij}} = \frac{O_{ij}}{\sum_{j=1}^{m} O_{ij}} = \frac{N_i^1 \cdot \alpha_{ij}^1 + N_i^2 \cdot \alpha_{ij}^2}{N_i^1 + N_i^2}$$

so the probability of not being guessed *at the end* of the second period is

$$P_{NG_{ij}} = 1 - P_{guess} = \frac{N_i^1(1-\alpha_{ij}^1) + N_i^2 \cdot (1-\alpha_{ij}^2)}{N_i^1 + N_i^2}$$

What happens in the middle, i.e. *during* the second period? The probability of not being guessed changes at every step because the number of times each response has been observed by T varies along time. This variation can be modeled as follows. At a certain step s of the second period (s is measured from the beginning of the period, so $0 \leqslant s \leqslant N_i^2$, with N_i^2 being the length of the second period), the probability that T correctly predicts response j to event i is

$$P_{guess_{ij}} = \frac{O_{ij}}{\sum_{j=1}^{m} O_{ij}} = \frac{N_i^1 \cdot \alpha_{ij}^1 + s \cdot \alpha_{ij}^2}{N_i^1 + s}$$

and the probability of not being guessed is then

$$P_{NG_{ij}} = 1 - P_{guess} = \frac{N_i^1(1-\alpha_{ij}^1) + s \cdot (1-\alpha_{ij}^2)}{N_i^1 + s}$$

As stated before, notice that this probability changes at every step within a period. Now, it is possible to generalize this reasoning to obtain the probability of not being guessed at step s of the h-th period ($0 \leqslant s \leqslant N_i^h$):

$$P_{NG_{ij}} = \frac{\sum_{k=1}^{h-1} N_i^k(1-\alpha_{ij}^k) + s \cdot (1-\alpha_{ij}^h)}{\sum_{k=1}^{h-1} N_i^k + s} \tag{5.11}$$

If we want to express such probability as a function of the number t of times that event e_i has arisen, we can rewrite the above expression as follows:

$$P_{NG_{ij}}(t) = \frac{\sum_{k=1}^{U_i(t)-1} N_i^k(1-\alpha_{ij}^k) + Y_i(t) \cdot (1-\alpha_{ij}^{U_i(t)})}{t} \tag{5.12}$$

$$U_i(t) = \begin{cases} 1 \text{ if } & t < N_i^1 \\ 2 \text{ if } & N_i^1 \leqslant t < N_i^1 + N_i^2 \\ 3 \text{ if } & N_i^1 + N_i^2 \leqslant t < N_i^1 + N_i^2 + N_i^3 \\ \ldots \end{cases}$$

$$Y_i(t) = t - \sum_{k=1}^{U_i(t)-1} N_i^k$$

Function U_i maps the number of occurrences of event e_i (including the present one) to the period whose weights should be used at current stage, according to the dynamic strategy for event e_i. It is similar to function H_i defined in the previous section, but its argument is not a step of the simulation but the number of times event e_i has arisen. It is basically a formalism rather than a true mathematical function. Similarly, function Y_i computes the number of times that event e_i has arisen during the *current* period of the dynamic strategy, which is equivalent to the value of s in expression 5.11. In other words, it is the number of occurrences of e_i, but counted from the beginning of current period.

Expected payoff with dynamic mixed strategies. The expression of the total expected payoff with dynamic strategies is shown below. It is a generalization of Eq. 5.6, using 5.12 as the probability of not being guessed. The marginal probabilities of each event $P_k[I = e_i]$ explained in the previous section do not appear explicitly in this expression, but recall that the estimations L_i depend on such probabilities.

$$EP_{dyn}((\alpha_{ij}^{h}), (N_i^h)) = \sum_{i=1}^{n} \sum_{k=1}^{\lfloor L_i \rfloor} \sum_{j=1}^{m} \alpha_{ij}^{U_i(k)} \cdot P_{NG_{ij}}(k) \cdot p_{ij} \tag{5.13}$$

Optimal strategy maximizing the expected payoff. Recall that the values L_i ultimately depend on the values α_{ij}^h. If we want to maximize the expected payoff according to this expression, the optimization problem must contain all those unknown weights simultaneously. The number of periods H_i of a dynamic strategy is not part of the optimization process and must be set by the user. For simplicity, we will consider that number to be the same for all events, so $H_i = H$ for all $i = 1, ..., n$. The length of the periods for strategy i should equal the number of times that each event e_i is expected to arise, L_i: $\sum_{h=1}^{H_i} N_i^h = L_i, \quad i = 1, ..., n$. These should be additional constraints in the optimization process that will be carried out to determine the optimal values of the weights α_{ij}^h and the periods N_i^h that we are searching.

With this approach, the number of unknown parameters is greater than that of static mixed strategies. Instead of computing only m x n weights, we have to compute H sets of weights per event, and all these variables are related because they mutually influence the probabilities of the events that will arise so the problem cannot be broken down in smaller optimization problems as proposed in [91]. In total, the problem being solved has $(n\cdot m\cdot H)$ + $(n\cdot H)$ unknown variables. In this sum, the first product is the number of weights α_{ij}^h. A dynamic strategy has H periods, and there are m weights in each period. Since we allow more flexibility, S maintains a different strategy for each event so there are n independent dynamic strategies with H x m weights each. The second product represents the lengths of the periods, which are positive integers. Recall that the optimal length N_i^h of every period is being optimized, and there are H periods in each of the n dynamic strategies used by S. The problem is thus a non-linear mixed optimization problem. The above summation yields 120 unknown real variables for a simple instance of our adversarial model with $m = 5$ actions, $n = 5$ events and dynamic strategies with $H = 4$ different periods, which means that the complexity of the problem makes it hard to solve using exact mathematical optimization methods. Formally the optimization problem can be described as

$$\max_{\{\alpha_{ij}^h \cup N_i^h\}} \left\{ \sum_{i=1}^{n} \sum_{k=1}^{\lfloor L_i \rfloor} \sum_{j=1}^{m} \alpha_{ij}^{H_i(k)} \cdot P_{NG_{ij}}(k) \cdot p_{ij} \right\} \tag{5.14}$$

subject to:

$$\sum_{j=1}^{m} \alpha_{ij}^h = 1 \quad i = 1, ..., n; h = 1, ..., H$$

$$\begin{aligned} \alpha_{ij}^h &\geqslant 0 \quad i = 1, ..., n; j = 1, ..., m; h = 1, ..., H \\ \sum_{h=1}^{H} N_i^h &= L_i \quad i = 1, ..., n \end{aligned}$$

$$\text{Recall}: L_i = \sum_{k=1}^{L} P_k[I = e_i] \quad i = 1, ..., n$$

5.3 Experiments and results

The experiments we conducted are aimed at answering the following questions:

1. Is there any substantial difference between the theoretical expected payoff and the average payoff attained by empirical simulations?
2. Do dynamic mixed strategies outperform a static mixed strategy in terms of expected payoff?
3. How are dynamic strategies affected by the number of different periods employed?

In order to answer these questions we follow the next steps.

Model Configuration. The parameter configuration of the model instance that has been used in the empirical evaluation of strategies was the following:

- Number of events and actions: $n = m = 5$
- Length of the input sequences: $L = 500$.
- Matrix of conditional probabilities:

$$C = \begin{pmatrix} 0.2 & 0.5 & 0.15 & 0.1 & 0.05 \\ 0.4 & 0.1 & 0.25 & 0.05 & 0.2 \\ 0.15 & 0.2 & 0.4 & 0.1 & 0.15 \\ 0.1 & 0.1 & 0.2 & 0.5 & 0.1 \\ 0.3 & 0.4 & 0.3 & 0 & 0 \end{pmatrix}$$

Payoff matrices. 15 different matrices were tested. For each matrix, a set of m payoffs is defined, and every row of the matrix has a permutation of the same set. The payoffs of every matrix are summarized in Table 5.1. The rest of the rows of each matrix are different permutations of the set displayed in the table. An extra column is shown containing the maximum total payoff attainable by S after 500 events if it always chooses the action with the largest payoff and is never guessed. This is the ideal situation that would only occur when there is no adversary, so it is taken as a reference.

As can be seen in the table, the matrices are characterized by an increasing difference between the payoff attained when choosing the best action, the second-best action and so on. When this difference is big, it becomes much more important to achieve a good balance between the payoff attained and the confusion caused, since choosing low payoff actions to induce confusion leads to a great loss in the payoff when compared with the greatest-payoff action.

Table 5.1: Set of payoffs associated to each payoff matrix.

Payoff matrix	First row					Max. reward after 500 ev.
M_1	1	0.9	0.95	0.8	0.85	500
M_2	0.8	0.9	0.6	0.7	1	500
M_3	1	0.85	0.7	0.4	0.55	500
M_4	1	0.6	0.8	0.4	0.2	500
M_5	0.25	0.01	0.5	1	0.75	500
M_6	1.1	0.95	0.9	1.05	1	550
M_7	1.2	1	1.1	0.9	0.8	600
M_8	1.3	1	1.15	0.85	0.7	650
M_9	1.2	1.4	1	0.8	0.6	700
M_{10}	1.5	1	0.75	1.25	0.5	750
M_{11}	0.8	0.6	0.4	1.5	1	750
M_{12}	0.8	0.6	0.4	1.75	1	875
M_{13}	0.8	0.6	0.4	2	1	1000
M_{14}	0.8	0.6	0.4	2.25	1	1125
M_{15}	0.8	0.6	0.4	2.5	1	1250

Evaluation of a strategy. When a strategy is evaluated empirically, Algorithm 1 is run 100 independent times and the payoff attained by S is annotated. This value is transformed into a percentage over the maximum payoff attainable in one 500-event execution (see Table 5.1). The average of such percentages is taken as the empirical payoff of the strategy.

Optimization algorithm. An important point is the optimization method employed to solve the problems formalized in Eq. 5.7 and 5.14. A 4-period dynamic mixed strategy in an adversarial model with $n = 5$ events and m

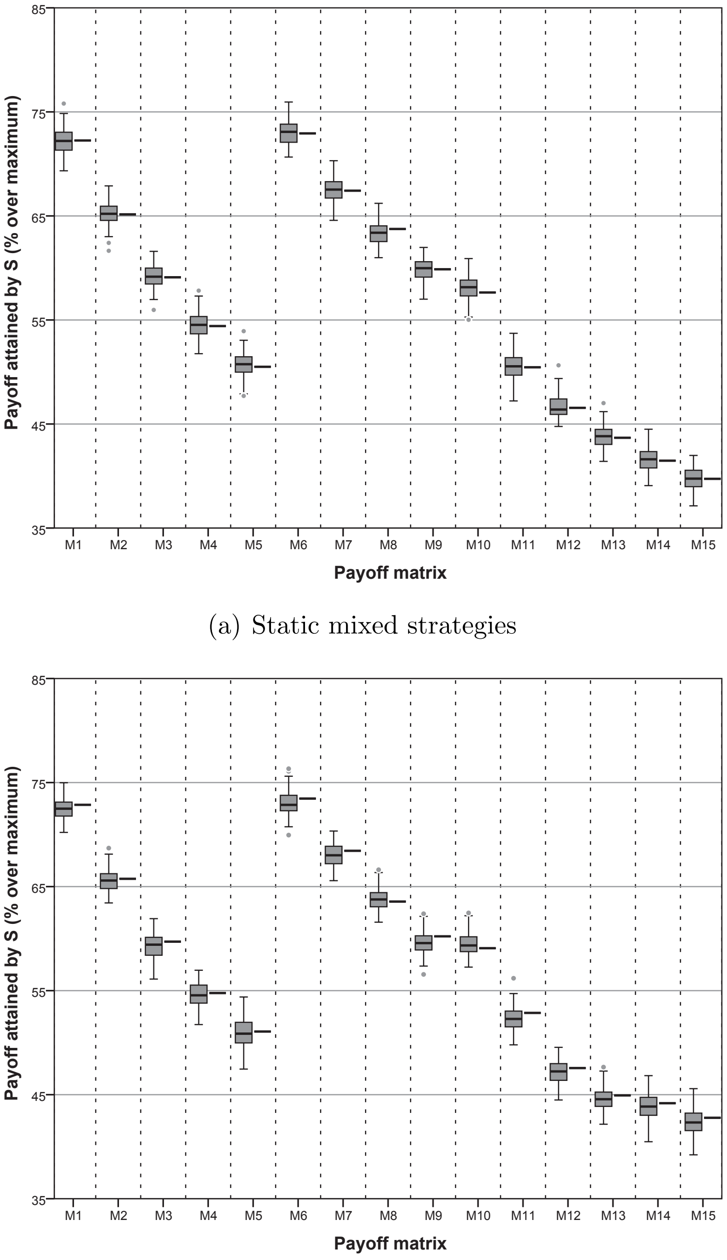
= 5 actions has 120 variables (100 real numbers representing the weights of the strategies at each period, and 20 integer values representing the lengths of the periods for each event) to be optimized. This optimization problem is very hard to solve, so we have made use of a heuristic optimization method called *Differential Evolution* (DE) because it is particularly suited for real optimization. A lot of variants of DE have been proposed since it was first introduced in [83, 71]. We have employed an enhanced auto-adaptive variant called *Self-adaptiveDE* (SADE [72]) which shows specially good performance in high-dimensionality problems. We used a Java implementation that is freely available[2] and has been already used in machine learning studies that ultimately require real optimization [86]. No formal study has been conducted to tune the parameters of the algorithm but some preliminary experiments led to the following values:

- Crossover operator: binomial crossover
- Crossover probability: 0.5 for static strategies and 0.9 for dynamic strategies
- Scaling factor: 0.5 for both static and dynamic strategies
- Population size: 50
- Number of iterations: 50000.

In order to answer the first question asked at the beginning of this section, it is necessary to evaluate empirically and theoretically one (or more) mixed strategy and check that both results match. We could pick any arbitrary strategy for this, but we decided to employ the strategies obtained after applying a basic DE optimization algorithm. The steps were the following:

1. For each payoff matrix, run a fast, basic DE twice (once to get an optimized static strategy and once more to get a 4-period dynamic strategy) with the above parameters. As a result, we get 15 optimized static strategies and 15 optimized dynamic strategies. Since the only goal here is to compare expected payoff with empirical values, it is not

[2] http://sci2s.ugr.es/EAMHCO/src2/advancesDEs.zip



(a) Static mixed strategies

(b) 4-period dynamic mixed strategies

Figure 5.3: Empirical (gray boxes) and expected payoff (lines outside the boxes) for every payoff matrix using the optimized static and dynamic strategies. Values expressed as percentages over the maximum possible. Data of empirical payoff after 100 runs are depicted in gray boxes while expected payoff is represented by lines outside the boxes.

important to do several independent runs of the optimization process.

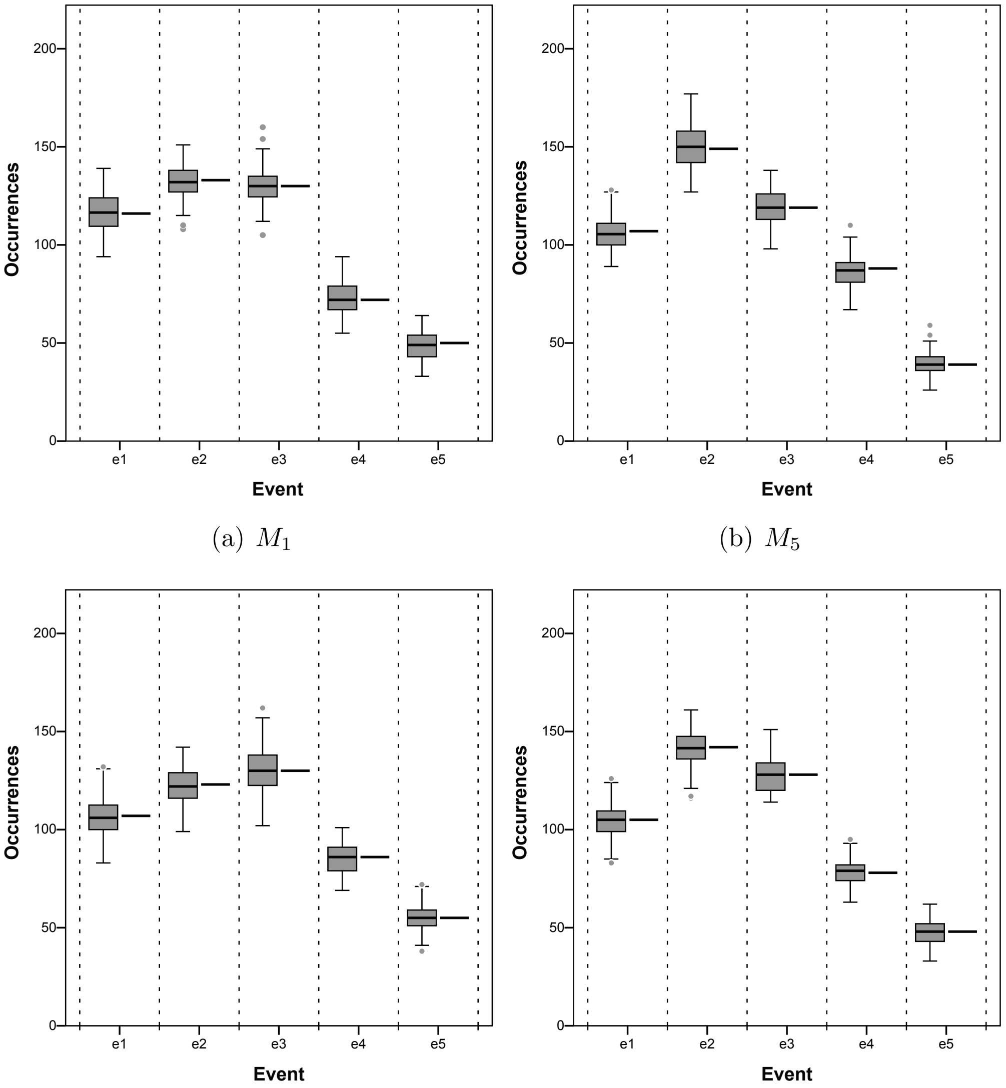
2. Evaluate every strategy theoretically, applying expressions 5.6 or 5.13 as required if the strategy is static or dynamic. As a result, we get 15 expected payoff values corresponding to static strategies and 15 values corresponding to dynamic strategies.

3. Evaluate every strategy empirically by running 100 independent times Algorithm 1 (section 2.4.1). The total payoff is annotated after each run, and the mean of the 100 resulting values is taken as the empirical payoff of the strategy. As a result, we get 15 empirical payoff values corresponding to the static strategies and another 15 corresponding to dynamic strategies.

4. Compare the expected and empirical payoff every strategy to check the differences and the variability. Recall that the expected payoff indicates the average behaviour, and that is why several independent runs are required when empirically evaluating a strategy.

Figs. 5.3(a) and 5.3(b) show a comparison of the expected and empirical payoff of the static and dynamic mixed strategies found in step 1. The comparison was done as indicated in step 4. The plots confirm an almost perfect matching between the expected and the empirical average payoff attained.

In the case of the dynamic strategies, one of the most difficult parts is the correct estimation of the number of occurrences of each event. These values were also annotated in the simulations of the optimized 4-period dynamic strategies, in order to contrast them with the theoretical estimation. The results are displayed in Fig 5.4, which only shows four of the 15 payoff matrices (M_1, M_5, M_{10} and M_{15}) because the results on the others are similar. An almost perfect matching was achieved between the expected and the empirical average number of events of each type.



(a) M_1

(b) M_5

(c) M_{10}

(d) M_{15}

Figure 5.4: Number of occurrences of each event during a 500-step simulation with 4-period dynamic strategies for four of the payoff matrices. Empirical (gray boxes) and expected values (lines outside the boxes) after 100 independent runs with each payoff matrix.

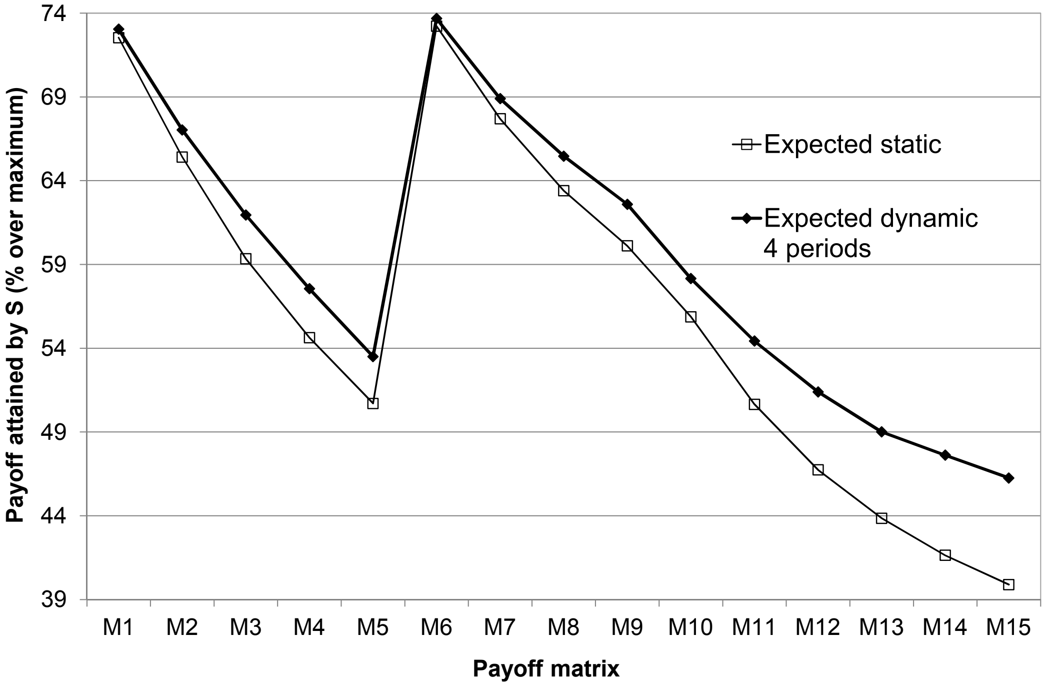


Figure 5.5: Expected payoff of the best static and 4-period dynamic strategies found by SADE in five independent runs. Values expressed as percentages over the maximum possible.

5.3.1 Static vs dynamic strategies: performance comparison

We now analyze the performance of both static and dynamic mixed strategies by comparing their expected payoff to answer the second question asked above. For this experiment, the strategies tested are the best strategies obtained for each payoff matrix after five independent runs of the SADE optimization algorithm. The results are shown in Table 5.2 and Fig. 5.5.

The most important conclusion on this figure is the following. In all the payoff matrices tested, the optimal 4-period dynamic mixed strategy consistently outperformed the optimal static strategy. The differences are statistically significant (p = 6.1E-5) at any significance level according to a paired Wilcoxon signed rank test with the two samples of Table 5.2 (the samples are paired due to the payoff matrices they share). Recall that each value of the table does not come from a simulation but from a prediction

Table 5.2: Payoff attained by static and dynamic strategies found by SADE, as a percentage over the maximum.

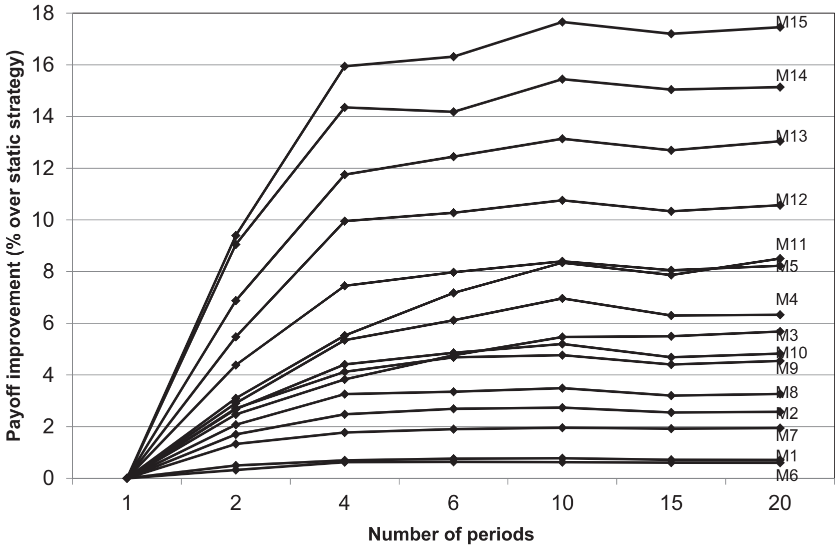
Strategy:	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
Static	72.54	65.41	59.34	54.63	50.70	73.22	67.69	63.40	60.11	55.87	50.65	46.74	43.85	41.64	39.89
Dyn H=4	73.04	67.03	61.95	57.55	53.50	73.68	68.90	65.47	62.59	58.16	54.42	51.39	49.01	47.61	46.25

made using the expressions, so it is not influenced by random factors apart from the payoff matrix generated for each scenario. Notice that the greater gain in performance was achieved in matrices M_{11} to M_{15}, which are those where the highest payoff is much greater than the rest. This is a particularly encouraging result for problems in which it is very important to do the best action as many times as possible because its payoff is much greater than that of the rest of the alternatives.

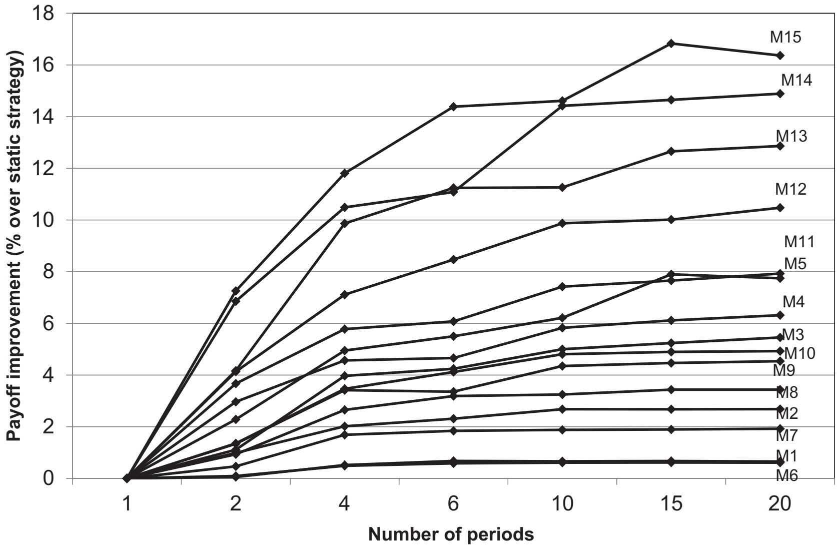
5.3.2 On the influence of the number of periods

In order to provide insights on the impact the number of periods of a dynamic mixed strategy has over the performance, an experimental sampling with different number of periods has been done. Again, the SADE optimization algorithm has been run five times independently to find a dynamic strategy with a fixed number of periods, and the average of the best solutions found in the five runs has been annotated. This has been repeated for $H \in \{1, 2, 4, 6, 10\}$ in two different contexts: one with $L = 500$ steps and one with $L = 1000$ steps, in order to assess how the number of steps affects the conclusions. The improvements for each payoff matrix with respect to the optimal static mixed strategy ($H = 1$) are displayed in Fig. 5.6.

Fig. 5.6(a) shows that increasing the number of periods causes a severe improvement at the beginning, but becomes less pronounced after four periods, reaching its maximum at ten periods. This means that using more than four periods hardly enhances performance. Therefore our choice of $H = 4$ was a good one to show the benefits of the dynamic strategies approach (although



(a) $L = 500$ steps



(b) $L = 1000$ steps

Figure 5.6: Improvement achieved with dynamic mixed strategies for different numbers of periods. Average results over 5 independent runs of the SADE optimization algorithm for each payoff matrix.

some additional gains are also observed for up to ten periods). Further, it is confirmed once again that dynamic mixed strategies provide a higher improvement when applied to more difficult payoff matrices. For instance, M_{15} is the most difficult one and where the improvement is the largest, followed by M_{14} and M_{13}. The same happens for matrices M_5, M_4 and M_3, and also for M_{10}, M_9 and M_8.

In Fig. 5.6(b) the trend is similar, but the improvement is gradual and continues growing until reaching 20 periods (although the gain when moving from 15 to 20 is quite small). Since 1000 steps are being considered here, we can expect that there is more room for improvement using more periods because there is more time available to switch to a new strategy. Intuitively, this should be done when T has learnt S's strategy and can no longer be deceived, except by switching to a new set of weights.

Notice that the optimal value of H also depends on the length of the simulation, since it is necessary to play a given strategy for long enough (before switching to a different one) in order to cause the intended manipulation, reflected in how agent T's observation matrix changes. For this reason, the improvement is small after $H = 4$ and stops at $H = 10$ with 500 steps, but keeps growing gradually until $H = 20$ when 1000 steps are played. The trend of Fig. 5.6(b) also confirms that SADE still works well when increasing the number of unknowns, as additional gains are achieved with $H = 15$ and $H = 20$ provided that the simulation is long enough. Therefore, the little and subsequent no improvement at all after $H = 10$ in Fig. 5.6(a) is due to the number of steps of the simulation (too short for strategies with more than ten periods), and not to a fail of the optimization process.

5.4 Conclusions

An extension to an existing adversarial model has been proposed consisting in introducing statistical dependence between actions and the next event. Static and dynamic mixed strategies for an agent in this extended model have been successfully designed using heuristic optimization methods. Analytical expressions of the expected payoff for both strategies have been provided and validated also from an empirical point of view. The design of good strategies has been tackled as a non linear mixed optimization problem and solved using heuristic techniques. Furthermore, dynamic mixed strategies found by SADE have shown to outperform the best static mixed strategies found by the same algorithm in all the scenarios tested, specially when the difference between the payoff of the best action and the payoff of the rest of actions becomes greater.

Chapter 6

Estimation of Fuzzy Stationary Probabilities of a Markovian patrolling strategy

The preceding chapters have focused on an imitation model. We have presented several strategies for agent S in the original model and in two extensions of it. All the strategies share one feature: they assume that the adversary has no information other than the past observations about S's decisions. Therefore, agent S intentionally makes decisions that are aimed at being observed and recorded by T, so that in the long term, they will cause deception. In this chapter, we switch our view to the other player, i.e. to the watching agent, and propose a way to obtain more information from the observations of the behaviour of a moving agent.

The application that motivates this work is the use of game-theoretic techniques to develop patrolling models for autonomous robots. They are a particular example of a Security Game played by the patrolling agent (robot) and the potential intruder. As in the imitation game described so far, the patroller computes the optimal randomized (Markovian) patrolling strategy. Since the area being patrolled is assumed big enough to make it impossible to protect all the locations all the time, randomness is necessary to prevent the exploitation of some uncovered[1] locations that would arise in a deterministic

[1] Not covered with enough frequency to avoid a successful attack. We will provide more

movement pattern. The intruder first learns the probabilities that guide the randomized movement, which happens to be a Markov chain, and then chooses a location to attack. Note that the only information to learn the patroller's probabilities consists of past observations about the patroller's behaviour. This has a clear similarity with the observations of agent T and the deceptive actions of agent S in our imitation model.

In the remainder of this chapter, we present a mathematical technique that allows the observer (in this case, the potential intruder) based on fuzzy numbers and fuzzy Markov chains to extract more useful information from the past observations about the patrolling strategy. The method is valid for any Markov chain. As a simple example, consider that an external observer, after conducting surveillance of a robot moving in an area, comes up with vague information expressed in the following terms: *The robot goes very often to location 6 when it is in location 1; it almost never goes from there to location 2; it does not come very often to location 3; it tends to go to state 8 when it is located in location 5 almost surely*, etc. The mathematical method we present allows to capture the uncertainty associated with these statements, and to carry computations with them. Finally, we also describe an implementation of the method as an R package, and show how it can be applied to the patrolling problem.

6.1 Markov chains

Markov chains are well-established probabilistic models of a wide variety of real systems that evolve along the time. Countless examples of applications of Markov chains to successfully capture the probabilistic nature of real problems include areas as diverse as biology [3], medicine and particularly disease expansion models [38, 77, 42, 39, 63], economy [45] -with emphasis on market modelling [87], social science and many other engineering problems such as wind speed modeling [75]. They are also the basis for Hidden Markov models, which constitute themselves another active research field with important applications in such problems as speech recognition [46] and

details later when the model is fully explained.

automatic music generation [78], just to cite a few. A broader survey can be found in [28]. For our purposes, a Markov chain successfully represents a randomized patrolling strategy of an autonomous agent following a randomized movement to defend an area.

Assume we observe the state of a system at certain time points (i.e., time is discretized for our observations). An observation X_t obtained at time t represents one state the system can be in. In fact, the collection of observations $X_0, X_1, \ldots, X_n$ represent an indexed sequence of random variables, each of which can take values within the system *state space* $S = \{1, \ldots, r\}$, which we will assume finite. Such indexed sequence $\{X_t, t = 0, 1, 2, \ldots\}$ is known as a stochastic process [60] in discrete time and discrete state space S.

A stochastic process $\{X_t, t = 0, 1, 2, \ldots\}$ with discrete state space and discrete time is a Markov chain if, and only if, it has the Markov property, which can be stated in simple terms as the fact that the state of the system at the next time point depends only on the state at the current time, and it is independent of the succession of states reached in the past. Formally, this independence from the past is expressed as

$$\begin{aligned} & P[X_{t+1} = x_{t+1} | X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0] = \\ = \; & P[X_{t+1} = x_{t+1} | X_t = x_t] \qquad (6.1) \end{aligned}$$

When the probabilities $P[X_{t+1} = x_{t+1} | X_t = x_t]$ do not depend on the time points $t, t+1$ but only on the time difference h between them, the chain is said to be homogeneous (in the preceding example, $h = 1$). In that case, it is usual to collect the transition probabilities in a (stochastic) transition matrix $P^{(h)} = (p_{ij}^{(h)})$ with dimensions $r \times r$ representing the transition probabilities in h steps, $p_{ij}^{(h)} = P[X_{t+h} = j | X_t = i]$. In general, $P^{(h)} = P^h$.

One of the features that characterize certain kinds of Markov chains is the stationary distribution, which represents the probability of the chain being at each state after an infinite number of time steps. It can also be thought of as the percentage of time the chain spends at each of its states. The computation of such probabilities requires an accurate knowledge of the transition matrix of the chain. If it is not available, it has to be estimated in some way, possibly from a set of observations of the chain, with the uncertainty that any estimation procedure involves.

6.1.1 Related work

Several methods have been proposed to estimate transition probabilities, see [31] and more recently [100]. Other works have addressed the problem of computing stationary probabilities with uncertain data from a mathematical programming point of view [16]. The approach adopted in the remainder of this work is based on using fuzzy sets [103, 104] to cope with uncertainty, thus considering fuzzy transition probabilities that constitute a fuzzy Markov chain.

Quite a lot of research has been conducted on fuzzy Markov chains. Zadeh himself envisaged their potential importance in fuzzy Markov algorithms in [106]. Other recent successful applications of fuzzy Markov chains are [50] which deals with processor power, [85] for speech recognition, and [35, 10] for multitemporal image classification. Some works consider a fuzzy state space [47]. On the contrary, in our proposal we consider crisp states but fuzzy probabilities. In [51, 12] and in most of the works mentioned previously on fuzzy Markov chains, the uncertain transition matrix is modeled as a fuzzy relation between the states. Hence there is no restriction on the values it can take, as far as they are membership grades in the closed interval [0, 1] that in fact represent a perception of a classical Markov chain [50].

Another, more restrictive approach presented in a series of more general works on fuzzy probability theory [22, 23, 24, 25, 26] consists in defining fuzzy numbers for the transition matrix, subject to the crisp constraint that they must add to 1, no matter how uncertain they are. Operations with the fuzzy transition matrix are carried using a restricted fuzzy matrix multiplication. Little work has been done in this line regarding Markov chains, and the author's ideas still remain as a theoretically feasible procedure whose practical applicability has not been investigated. It may be suitable when the input is a sequence of crisp observations of the chain. For those reasons we have developed a new software package called `FuzzyStatProb` for the R environment [73]. Moreover, to the best of our knowledge, currently there are no implementations available -either free or proprietary- of any of the fuzzy Markov chain approaches, so this aims to be the first one and pave the way for further programming efforts in this field.

The method can be summarized as follows: the Fuzzy Probabilities of the fuzzy transition matrix will be first decomposed in their α-cuts, which will be used to solve the problem in separate pieces, and finally the results will be aggregated to re-construct the fuzzy stationary probabilities we are searching for. The starting point of the problem is a finite sequence of observations of the system state at each step. This is the input required by the `FuzzyStatProb` package we have developed. The output is provided as a list of fuzzy numbers, one for each state of the input Markov chain, using the `FuzzyNumbers` package described in [36] that provides plotting functionality and is currently under active development.

The chapter is structured as follows. In Section 6.2, the background and the mathematical method are formally developed. Section 6.3 describes implementation details and the signature and options of the public function of the package. Section 6.4 shows use examples and provides insights on the effect of a greater number of observations over the fuzzy output of the method. Finally, Section 6.5 is devoted to conclusions and further work.

6.2 A method to compute fuzzy stationary probabilities

An homogeneous Markov chain is said to be irreducible if all states form a single group so that every state is accessible from every other state, be it in one or more than one step. A finite, irreducible Markov chain with transition matrix P has a *stationary distribution* π such that $\pi = \pi P$ [60]. The `FuzzyStatProb` package is aimed at computing a vector of fuzzy stationary probabilities $\tilde{\pi} = (\tilde{\pi}_1, \ldots, \tilde{\pi}_r)$ departing from an uncertain (fuzzy) transition matrix $\tilde{P}$. The remainder of this section briefly summarizes the ideas on fuzzy Markov chains presented in [23], as they lie at the very core of our package.

6.2.1 Fuzzy numbers

Let us consider an uncertain transition matrix $\tilde{P} = (\tilde{p}_{ij})$ in which one or more elements are uncertain. We capture the uncertainty regarding some of the elements by substituing them by fuzzy numbers, which are a special case of fuzzy set [103, 62]. Roughly speaking, a fuzzy set $\tilde{A}$ over a universe U is a set whose characteristic function is not binary, $f : U \rightarrow \{0, 1\}$ as in a classic set, but takes values in the unit interval, $\mu : U \rightarrow [0, 1]$, so that $\mu_{\tilde{A}}(x)$ is the degree to which element $x \in U$ belongs to the fuzzy set $\tilde{A}$. A fuzzy number is a fuzzy set over $\mathbb{R}$ representing a number with uncertainty [32]. For instance the uncertain quantity "about 2" may be represented as a fuzzy number $\tilde{A}$ whose membership function reaches its maximum value 1 at $x = 2$, but also assigns non-zero membership degrees to other values close to 2, since a value of, say, 1.8 is also ('belongs also to') "about 2", not with degree 1 but a bit less, say, 0.6. Then $\mu_{\tilde{A}}(2) = 1$ and $\mu_{\tilde{A}}(1.8) = 0.6$. We now explain this concept formally as it is the mathematical structure used in our package to cope with uncertainty in the transition and stationary probabilities.

Definition 1. *The α-cut of a fuzzy set $\tilde{A}$ for any $\alpha \in [0, 1]$ is defined as the set of elements that belong to $\tilde{A}$ with degree α or greater:*

$$\tilde{A}(\alpha) = \{x \in U : \mu_{\tilde{A}}(x) \geqslant \alpha\}$$

Definition 2. *A fuzzy number $\tilde{A}$ is a fuzzy set on $\mathbb{R}$ such that*

(i) $\tilde{A}(\alpha)$ *are nonempty convex sets* $\forall \alpha \in [0, 1]$

(ii) $\tilde{A}(\alpha)$ *are compact sets* $\forall \alpha \in [0, 1]$

(iii) $\tilde{A}(\alpha)$ *satisfy:*

(a) $\tilde{A}(\alpha) \subseteq \tilde{A}(\beta)$ *for* $\alpha > \beta$

(b) $\tilde{A}(\alpha) = \bigcap_{\beta<\alpha} \tilde{A}(\beta)$ *for* $\alpha \in (0, 1]$

Since α-cuts of a fuzzy number are closed intervals of $\mathbb{R}$, they can be written as $\tilde{A}(\alpha) = [\tilde{A}_L(\alpha), \tilde{A}_R(\alpha)]$.

The first two conditions are basically equivalent to the fact that the membership function of a fuzzy number must be continuous and monotone, which are two features of the output fuzzy numbers computed by our method as will be shown later.

6.2.2 Fuzzy transition probabilities from observations

As stated in Section 6.1, our data consists of a finite sequence of observations drawn from the system at consecutive time points. Each observation is the state of the system at that time point, represented as an integer belonging to the state space of the chain. In such sequence, an estimation of the transition probability p_{ij} can be computed as the number of times N_{ij} that state i was followed by j in our data, divided by the number of transitions N_i observed from i to another state (including i itself). However, this would be just a point estimate and does not capture the uncertainty associated to it. Interval estimation would be the next step as it also takes into account the sample size and variance of the data, but it calculates an interval only for a significance level that is fixed beforehand.

Our aim is to model each uncertain transition probability as a fuzzy number that properly captures all the uncertainty of the data, regardless of external parameters such as the confidence level. For this purpose, the fuzzy number is obtained by the superposition of confidence intervals for the true transition probabilities at different confidence levels, one on top of another, as done in [23]. Although the significance level and the membership degree are numerically the same in this case, it should be noted that they are completely different concepts for which the letter α is commonly used. Such intervals are in fact simultaneous confidence intervals for multinomial proportions, since the problem of estimating the transition probabilities from a given state can be reduced to that of estimating the parameters (proportions) of a multinomial distribution. The method applied was introduced by Sison and Glaz [80, 56] although many others had been proposed before.

6.2.3 Fuzzy Markov chains and restricted matrix multiplication

As we have said, fuzzy numbers are used in those entries of the transition matrix on which there is uncertainty. The rest of the elements could be crisp since a crisp number is a special case of fuzzy number. However, the uncertainty is on the probabilities but not on the fact that every row must add to 1. Now assume a Markov chain with r states and fuzzy transition matrix $\tilde{P}$, then for a given $\alpha \in [0,1]$, $\tilde{P}(\alpha) = (\tilde{p}_{ij}(\alpha))$ represents a matrix of intervals. Such matrix can also be thought of as the set of all $r \times r$ matrices M such that their elements fall inside their corresponding interval, $m_{ij} \in \tilde{p}_{ij}(\alpha)$, and every row adds to 1, $\sum_{j=1}^{r} m_{ij} = 1, i = 1, \ldots, r$. The *domain* of row i for a given α value is defined as the set of r-dimensional vectors that simultaneously fulfill those two constraints for row i, i.e., the set of probability distributions that row i could take in our uncertain transition matrix when we take its α-cuts for that α. If we call $\Delta_r = \{(x_1, \ldots, x_r) : x_i \geqslant 0 \text{ and } \sum_{i=1}^{r} x_i = 1\}$, then

$$\mathrm{Dom}_i(\alpha) = \left(\bigtimes_{j=1}^{r} \tilde{p}_{ij}(\alpha) \right) \bigcap \Delta_r \tag{6.2}$$

The domain of the matrix for a fixed α, $\mathrm{Dom}(\alpha)$ is defined as the Cartesian product of the domain of every row and represents the space of matrices we admit when the uncertainty level is α. $\mathrm{Dom}(\alpha)$ is closed and bounded (compact) because it is the Cartesian product of compact sets. Therefore, any continuous function applied to its elements will have a compact image. In particular, as explained in [23, Chapter 6], the pow to exponent n of the $r \times r$ transition matrix (which is done as successive matrix multiplications) can be thought of as a collection of r^2 independent functions $f_{ij}^{(n)} : \mathbb{R}^{r^2} \to \mathbb{R}$, each of which calculates the value of one entry of the resulting matrix by operating with the entries of P: $p_{ij}^{(n)} = f_{ij}^{(n)}(p_{11}, \ldots, p_{1r}, p_{21}, \ldots, p_{2r}, \ldots, p_{r1}, \ldots, p_{rr})$. Such functions are continuous and thus, when applied on a compact set like $\mathrm{Dom}(\alpha)$ for a concrete α, the image of all of them is another compact set (a closed interval) of $\mathbb{R}$. Such interval can be viewed as an α-cut of the fuzzy number $\tilde{p}_{ij}^{(n)}$, i.e., $\tilde{p}_{ij}^{(n)}(\alpha) = f_{ij}^{(n)}(\mathrm{Dom}(\alpha))$. By the representation theorem [62], it is possible to reconstruct the fuzzy number $\tilde{p}_{ij}^{(n)}$ as the union of its α-cuts which, in theory, can be computed using the restricted fuzzy multiplication described before with different values of α. For our problem, we will use the superior of the α values as the union operator.

In particular, the stationary distribution π matches any of the (identical) rows of a matrix Π such that $\Pi = \lim_{n\to\infty} P^n$, which means that theoretically, π can be computed using matrix multiplication. We are thus in the same case explained above which guarantees that the resulting images are compact sets over $\mathbb{R}$, or more precisely the α-cuts of the fuzzy stationary probabilities we aim to calculate.

6.2.4 User-specified fuzzy transition probabilities

The package also supports user-specified fuzzy transition probabilities. Instead of departing from a sequence of observations of the state of the chain at consecutive time instants, the user provides a fuzzy transition matrix composed of fuzzy numbers that constitute the transition probabilities. As explained before, despite being fuzzy, these numbers must form a probability distribution for each row. According to previous works [40], this can be formalized by imposing the constraint that $\tilde{p}_{i1} \oplus \tilde{p}_{i2} \oplus \ldots \oplus \tilde{p}_{in} \supseteq 1_{\chi}$ for each row $i = 1, \ldots, n$. Here, $\tilde{A} \supseteq \tilde{B} \leftrightarrow \mu_{\tilde{A}}(x) \geqslant \mu_{\tilde{B}}(x) \forall x \in \mathbb{R}$, the symbol $\oplus$ stands for the fuzzy sum, and 1_{χ} is the real number 1 represented as a fuzzy number (singleton). Note this is a necessary condition to ensure that $\text{Dom}(\alpha)$ is not empty, as proved in [97] and, therefore, our code first checks this condition and stops if it is not fulfilled.

One of the main applications of fuzzy probabilities is related to linguistic probabilities [40], as we explain next. In case no data of the chain are available, we may have to elicit the transition probabilities from a human expert or group of experts. However, it can be difficult for them to express their expertise using numeric probabilities which, in addition, must fulfill the requirement of having a sum of exactly 1. Therefore, natural language can help express probabilities in a more natural way. The transition probability becomes a linguistic variable whose possible values are linguistic terms [105], such as *Very unlikely, Most likely, It may, Extremely likely* and so on. Each linguistic term has an underlying fuzzy number that captures the vagueness inherent to natural language. The input information would consist of subjective statements like *The robot goes very often to location 6 when it is in location 1, it almost never goes from there to location 2, when it is placed in location 5 it seems to go to state 8 almost surely*, etc.

It is not relevant for our implementation whether the fuzzy transition probabilities have an associated linguistic term or not, but each fuzzy number of the input transition matrix must be referred by a name when calling the function (see the next section).

6.2.5 Computation of fuzzy stationary probabilities

The approach outlined in Section 6.2.3 did not make direct use of the membership functions of the fuzzy transition probabilities, but worked only with their α-cuts to establish Dom(α). Thus it is not necessary to fully reconstruct such fuzzy numbers; it suffices to compute their α-cuts for the desired levels of α. Each of those levels provides us with an interval transition matrix that defines a domain, i.e., a space in which we admit our uncertain transition matrix lies.

Now, in order to find the α-cuts of the fuzzy stationary probabilities for a certain α, we just have to find, for each state of the chain, a pair of matrices within Dom(α) which respectively minimize and maximize the stationary probability of that state. This is repeated independently for every state. In other words, assuming a Markov chain with r different states, and focusing on a significance level α, we have to solve r independent minimization problems and r independent maximization problems in order to compute r different α-cuts of the fuzzy stationary probabilities we are looking for. Note that the search space is the same for all the problems, Dom(α), but the objective function is not: for each state s_j whose stationary probability $\tilde{\pi}_j$ is being computed, it is the expression which yields that stationary probability as a function of the entries of the transition matrix. The function is first minimized to find the lower bound of the α-cut $\tilde{\pi}_j(\alpha)$, and then maximized to find the upper bound. It seems clear that a general expression cannot be calculated for any r-state chain to apply conventional optimization techniques, since such function (in which all the coefficients p_{ij}

will be symbolic as well) would be too complicated for chains with more than 5 or 6 states. Furthermore, R symbolic capabilities are very poor and make it unsuitable. For both reasons, heuristic search (optimization) algorithms will be employed, as suggested in [23].

The α-cuts can be written as follows. Let $g_j : \mathbb{R}^{r^2} \rightarrow \mathbb{R}$ be a function that represents the calculation of the j-th component of the stationary distribution of an r-state Markov chain whose transition matrix is expressed as an r^2-dimensional vector $(p_{11}, \ldots, p_{rr})$. As explained in the previous section, such distribution can be obtained by means of matrix multiplication that converges to a matrix Π with identical rows that are indeed the stationary distribution. Using the same notation and assuming we are only interested in the first row of Π, we can set $g_j = \lim_{n\rightarrow\infty} f_{1j}^{(n)}$, i.e., g_j is the function that computes the element Π_{1j}. Such function only applies sums and products during the calculation process, so it is continuous and hence $g_j(\text{Dom}(\alpha))$ is a compact set on $\mathbb{R}$, i.e., an α-cut. Then

$$\begin{aligned} \tilde{\pi}_j(\alpha) &= [\pi_{jL}(\alpha), \pi_{jR}(\alpha)], \quad j = 1, \ldots, r && (6.3)\\ \pi_{jL}(\alpha) &= \min\{w_j | w_j = g_j(p_{11}, \ldots, p_{rr}), (p_{11}, \ldots, p_{rr}) \in \text{Dom}(\alpha)\} && (6.4)\\ \pi_{jR}(\alpha) &= \max\{w_j | w_j = g_j(p_{11}, \ldots, p_{rr}), (p_{11}, \ldots, p_{rr}) \in \text{Dom}(\alpha)\} && (6.5)\end{aligned}$$

Eq. 6.4 and 6.5 state that, in order to compute the lower and upper bounds of an α-cut of $\tilde{\pi}_j$ (the j-th fuzzy component of the fuzzy stationary distribution vector $\tilde{\boldsymbol{\pi}}$), we must find the minimum and the maximum of the j-th component of all the crisp stationary distributions corresponding to feasible crisp matrices, understanding feasibility as belonging to the space $\text{Dom}(\alpha)$ for that α. In practice, the condition $w_j = g_j(p_{11}, \ldots, p_{rr})$ is implemented as $\mathbf{w} = \mathbf{w}P$.

The final step is to calculate an analytical expression for the membership function of the fuzzy stationary probabilities. If the above α-cuts were exact, i.e., if we could guarantee that the solutions for the optimization problems are global optima, then the adequate way to proceed would be to interpolate the lower and upper bounds for every sampled α to separately obtain expressions for the left and the right sides of the membership function. The number of points depends on how much precision we need for the membership function. If the membership function has to be reconstructed very accurately, we should probably sample α values in [0, 1] in steps of, say, 0.01, which yields 100 points in each side. A larger step size of about 0.05 (20 points) is probably enough for most real problems and is less time consuming.

However, since we have no guarantee that the α-cuts are exact because they have been obtained by heuristic optimization procedures, regression may also be a good choice, and the loss of precision is globally quite small. The form of the regression function depends on what the points look like. In order to allow for the maximum flexibility, it is the user who can choose between spline interpolation or some kind of regression to be applied. More details are provided in Section 6.3.

Numerical example. The first part of the example does not use data from observations but serves to illustrate the mathematical procedure. Let $\tilde{A} = (a, b, c)$, where $a, b, c \in \mathbb{R}, a \leqslant b \leqslant c$, be a Triangular Fuzzy Number (TFN), which can be viewed as a particular case of fuzzy number with the following membership function:

$$\mu_{\tilde{A}}(x) = \begin{cases} (x-a)/(b-a) & a \leqslant x < b \\ (c-x)/(c-b) & b < x \leqslant c \\ 0 & \text{otherwise} \end{cases}$$

Now consider the fuzzy transition matrix depicted on the left, formed by TFNs which, in this particular case, are symmetric (although they do not have to). Note some of them carry more uncertainty than others. Focusing on a concrete value for α, for instance $\alpha = 0.5$, this matrix yields the interval matrix $\tilde{P}(0.5) = (\tilde{p}_{ij}(0.5))$ on the right, constituted by the 0.5-cuts of the fuzzy transition probabilities of $\tilde{P}$.

$$\begin{aligned} \tilde{P} &= \begin{pmatrix} (0.6,\ 0.7,\ 0.8) & (0.2,\ 0.4,\ 0.6) \\ (0.3,\ 0.4,\ 0.5) & (0.55,\ 0.6,\ 0.65) \end{pmatrix} \\ \tilde{P}(0.5) &= \begin{pmatrix} [0.65,\ 0.75] & [0.3,\ 0.5] \\ [0.35, 0.45] & [0.575,\ 0.625] \end{pmatrix} \end{aligned}$$

$$\begin{aligned} \mathrm{Dom}_1(0.5) &= \{(x,y) \in \mathbb{R}^2 : x + y = 1, x \in [0.65, 0.75], y \in [0.3, 0.5]\} \\ \mathrm{Dom}_2(0.5) &= \{(x,y) \in \mathbb{R}^2 : x + y = 1, x \in [0.35, 0.45], y \in [0.575, 0.625]\} \\ \mathrm{Dom}(0.5) &= \{(p_{11}, p_{12}, p_{21}, p_{22}) \in \mathbb{R}^4 : (p_{11}, p_{12}) \in \mathrm{Dom}_1(0.5), \\ &\quad (p_{21}, p_{22}) \in \mathrm{Dom}_2(0.5)\} \end{aligned}$$

Let g_1, g_2 be the functions that, for the given $\alpha = 0.5$, compute the elements Π_{11} and Π_{12} of the (crisp) stationary matrix Π corresponding to each of the 2×2 feasible matrices $\{P = (p_{11}, \ldots, p_{22}) \in \text{Dom}(0.5)\}$. Let us rename the crisp stationary distribution (Π_{11}, Π_{12}) of a feasible matrix as $\mathbf{w} = (w_1, w_2)$. We are interested in finding four feasible matrices $P_1, P_2, P_3, P_4 \in \text{Dom}(0.5)$ which, respectively, minimize and maximize w_1, and minimize and maximize w_2. The minimum and maximum w_1 act as the lower and upper bounds of the 0.5-cut $\tilde{\pi}_1(0.5)$, and the same applies to w_2 with respect to the 0.5-cut $\tilde{\pi}_2(0.5)$:

$$\begin{aligned}
\pi_{1L}(0.5) &= \min\{w_1 | w_1 = g_1(P), P \in \text{Dom}(0.5)\} = \min\{w_1 | \mathbf{w} = \mathbf{w}P, \\
&\quad P \in \text{Dom}(0.5)\} \\
\pi_{1R}(0.5) &= \max\{w_1 | w_1 = g_1(P), P \in \text{Dom}(0.5)\} = \min\{w_1 | \mathbf{w} = \mathbf{w}P, \\
&\quad P \in \text{Dom}(0.5)\} \\
\pi_{2L}(0.5) &= \min\{w_2 | w_2 = g_2(P), P \in \text{Dom}(0.5)\} = \min\{w_2 | \mathbf{w} = \mathbf{w}P, \\
&\quad P \in \text{Dom}(0.5)\} \\
\pi_{2R}(0.5) &= \max\{w_2 | w_2 = g_2(P), P \in \text{Dom}(0.5)\} = \min\{w_2 | \mathbf{w} = \mathbf{w}P, \\
&\quad P \in \text{Dom}(0.5)\}
\end{aligned}$$

This procedure should be repeated by sampling α in [0, 1] with a step size that depends on the precision desired to reconstruct the fuzzy stationary vector from its α-cuts.

Now, suppose we do not depart from a fuzzy transition matrix (provided by a human expert, for instance) as in the example above, but from a collection of observations of the state of the chain at several consecutive time instants, e.g. $\{1, 3, 2, 3, 4, 4, 2, 3...\}$. Then, in order to estimate the α-cuts of the fuzzy transition probabilities that are needed to establish the domains for each α, we resort to simultaneous confidence intervals at level α of multinomial proportions, using the number of times that the chain transitioned from one state to any other as the input. Once the CIs for the transition probabilities have been computed to compose the interval matrices $\tilde{P}(\alpha)$ for each α, we can define the domains and go on with the rest of the process as it remains unchanged. In this case, matrix $\tilde{P}$ is never constructed explicitly because we are only interested in the α-cuts of the fuzzy entries, as they constitute the constraints of the optimization problems.

6.3 The FuzzyStatProb package

We now describe the R implementation of the method in the `FuzzyStatProb` package. The signature of the only public function is

```
fuzzyStationaryProb(data, options, step = 0.05, ...)
```

where:

- `data`: This argument can be: (a) An array of either strings or natural numbers representing the observed states of the chain at consecutive time points. The function first coerces the elements to a factor. (b) A 2D square matrix of strings representing fuzzy transition probabilities directly given by the user. Each string should be contained in `names (fuzzynumbers)` and refers to the corresponding `FuzzyNumber` object in the `fuzzynumbers` vector (see below). When the transition probability from state i to j is 0 (in the crisp sense), then entry (i, j) must be NA. The matrix should have `colnames` and `rownames` set.

- `options`: a tagged list containing the following parameters:

  - `verbose`: boolean, set to `TRUE` if progress information should be printed during the process. It is set to `FALSE` if this option is not specified.

  - `states`: an array of strings indicating the states for which the stationary distribution should be computed. The values should match those specified in the `data` argument. If this option is not specified, the fuzzy stationary probabilities are computed for every state of the chain.

  - `acutsonly`: boolean, set to `TRUE` if no regression should be done after computing the α-cuts. This option is set to `FALSE` if not specified.

  - `regression`: a string with the type of the regression to be applied at the end of the algorithm for fitting the membership functions of the fuzzy stationary probabilities. Possible values are 'linear', 'quadratic', 'cubic', 'gaussian', 'spline' and 'piecewise' (piecewise linear). In all cases (including the gaussian), a different curve is fitted for each side of the fuzzy number. The `gaussian` option fits curves of the form $\mu(x) = \exp\left(-\frac{1}{2}\left|\frac{x-c}{s}\right|^m\right)$. The `spline` option performs interpolation by a monotone cubic spline according to the Hyman method (check `?splinefun`) while `piecewise` computes a piecewise linear membership function by connecting consecutive points of the α-cuts with straight lines, using the built-in subclass `PiecewiseLinearFuzzyNumber` of the `FuzzyNumbers` package. If this option is not specified, quadratic regression is carried out by default. If `acutsonly` is set to true, this option is ignored.

  - `ncores`: positive integer representing the maximum number of cores that can be used when running in parallel. If set to more than 1, then each processor takes care of all the computations involving one of the values of α that have to be sampled, via `parLapply` function of the `parallel` package. Defaults to 1 (sequential) if not specified. If `ncores` is greater than the actual number of cores in the computer, all available cores are used.
  - `fuzzynumbers`: a tagged list with all the different `FuzzyNumber` objects that appear in `data` when `data` is a matrix of labels; ignored otherwise. Every element of the list must have a name, referenced in at least one entry of `data`.

- `step`: step size for sampling α when computing the α-cuts. The smallest α is always present and equals 0.001, and the rest of values are calculated as $\alpha = k\cdot \mathit{step}$ for $k \geqslant 1$. The greatest sampled value that is always present as well is $\alpha = 0.999$. It is set to 0.05 when this option is not specified.

- `dots` Further arguments to be passed to `DEoptim.control` to customize the algorithm that finds the lower and upper bounds of the α-cuts by solving a minimization and a maximization problem.

The value returned by the function is a tagged list that belongs to a new S3 class called `FuzzyStatObj`. This class has only two specific methods, `print` and `summary`, and both print exactly the same: a brief summary of the processing that has been done, including the number of states, number of observations, regression type, step size and time elapsed, and information about the names of the two tags that should be queried to retrieve the results, namely `$fuzzyStatProb` and `$acuts`. The former is a tagged list of `FuzzyNumber` objects (see the `FuzzyNumbers` package cited before) whose length equals `length(options$states)` and whose names match those in `options$nstates`. The latter is another tagged list with the same length and names, consisting of data frame objects that contain the α-cuts of every output stationary probability, represented as pairs $(\tilde{\pi}_{jL}(\alpha), \alpha)$ and $(\tilde{\pi}_{jR}(\alpha), \alpha)$. The object at each position represents the fuzzy stationary probabilities and α-cuts of the element of the `options$states` list that is in the same position. Note that in case the user specifies within `options$state` a state that is not found among the elements of the `data` argument, then the corresponding position of `$fuzzyStatProb` and `$acuts` will be `NA`. When setting `acutsonly = TRUE`, the object returned does not have a `$fuzzyStatProb` tag but only `$acuts`. Section 6.4 contains a fragment of code and the R output showing how our function should be called.

6.3.1 Implementation issues

This section describes how each step of the mathematical process explained before has been implemented in R and the external packages used. A general scheme of the package is depicted in Figure 6.1, which has two possible starting points depending on whether the user provides a sequence of observations or a fuzzy transition matrix.

Confidence intervals. The first step is to build the fuzzy numbers of the fuzzy transition matrix, as the superposition of confidence intervals. The procedure samples several α from 0 to 1; the step size of the sampling is specified by the user in the `step` argument of the function. For each α and for each state s_i of the chain, the method builds as many simultaneous confidence intervals as transitions to other states s_j have been observed from

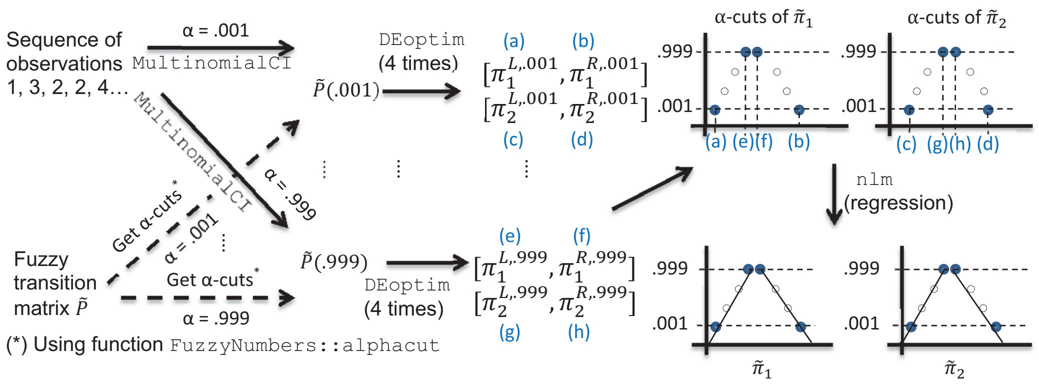

Sequence of observations 1, 3, 2, 2, 4…
α = .001
MultinomialCI
MultinomialCI
α = .999
Get α-cuts*
α = .001
Fuzzy transition matrix $\tilde{P}$
Get α-cuts*
α = .999
(*) Using function FuzzyNumbers::alphacut
DEoptim (4 times)
$\tilde{P}(.001)$
$\tilde{P}(.999)$
DEoptim (4 times)
(a) (b)
$[\pi_1^{L,.001}, \pi_1^{R,.001}]$
$[\pi_2^{L,.001}, \pi_2^{R,.001}]$
(c) (d)
(e) (f)
$[\pi_1^{L,.999}, \pi_1^{R,.999}]$
$[\pi_2^{L,.999}, \pi_2^{R,.999}]$
(g) (h)
α-cuts of $\tilde{\pi}_1$
α-cuts of $\tilde{\pi}_2$
.999
.001
(a) (e)(f) (b)
(c) (g)(h) (d)
nlm (regression)
$\tilde{\pi}_1$
$\tilde{\pi}_2$

Figure 6.1: Diagram of the function workflow and the R packages used in each phase.

s_i. The procedure makes use of the `MultinomialCI` package created by the first author [93] which implements the Sison and Glaz method [80] to calculate simultaneous confidence intervals for multinomial proportions. It is a direct R translation of the SAS code published in [56]. The `FuzzyStatProb` loads this package during the execution.

If the user has provided directly the fuzzy transition probabilities, this step is omitted and the α-cuts are taken directly from the fuzzy numbers passed to the function.

Computation of α-cuts of stationary probabilities. The second (and main) step of the method is to solve the optimization problems of Eq. 6.4 and 6.5, using some heuristic optimization technique. As explained in Section 6.2.5, the confidence intervals which are the α-cuts of the fuzzy entries of the transition matrix are used in this step as box-constraints for the optimization that searches for the minimum and maximum value of each stationary probability within Dom(α). The `DEoptim` package [61] was employed for this purpose. It is an implementation of the Differential Evolution (DE) algorithm [83, 71] that has exhibited excellent performance in a wide variety of hard continuous optimization problems. The package provides built-in capabilities to specify box-constraints for every variable but cannot handle equality constraints (in this case, the probabilities of every row of the matrix being evaluated must add to 1), so they have to be controlled directly in the objective function.

Computation of membership functions via regression. The third and last step is to reconstruct the membership function of the fuzzy stationary probabilities whose α-cuts were computed in the preceding step. Focusing on regression, the user can choose among linear, quadratic, cubic and gaussian regression. When fitting a quadratic or cubic membership function, care should be put to avoid non-monotonous functions, not allowed in this context for fuzzy numbers. To be precise:

- Since the membership function has to be continuous, it should intersect with the horizontal axis in at least one point between 0 and the central (core) point of the fuzzy number for the left-side function, and between the central point and 1 for the right-side function. If the function cuts the X axis in more than one point, only the greatest of those between 0 and the central point is considered, as well as the smallest of those between the central point and 1.

- The function only needs to be monotonic in those intervals so local minima cannot exist within them.

- The left and right functions (before normalizing to [0, 1]) must satisfy $f(\text{core}) = g(\text{core}) = 1$, so the degrees of freedom decrease in one because it is possible to write one of the curve parameters as a function of the others.

It is clear that the problem can be viewed as a constrained minimization of the squared error function with respect to the cloud of points representing the α-cuts. First, the `nls` function for Non-linear least squares regression is called. The diminished degrees of freedom are expressed in the formula passed to `nls`, in which one of the parameters of the curve is expressed as a function of the others. This function yields a solution satisfying $f(\text{core})$ $= 1$, but the rest of the conditions are not guaranteed by `nls` so they have to be checked in the obtained solution. If they are fulfilled, the solution is accepted and returned.

If they are not fulfilled, and taking into account that regression accuracy at this stage is not as fundamental as the satisfaction of all the constraints, again Differential Evolution is used to minimize the total quadratic regression error function for gaussian, quadratic and cubic regression. Constraints were implemented in the objective function of DE. For quadratic regression, the vertex must not fall between the horizontal cut-points and the core, and such cut-points cannot be negative nor greater than 1. For cubic regression, a turning point must not exist between the cut-points and the core. Linear and gaussian regression are themselves monotonic so the only constraint to be considered is the intersection with the horizontal axis. By definition, the gaussian function does not intersect the horizontal axis at any point but for practical purposes, it can be rounded to 0 when the membership degree is smaller than, say, 0.01.

Computation of membership functions via interpolation. This option always yields good and fast results, although the expression of the solution is a bit more complicated. Interpolation with cubic spline functions can be applied to the α-cuts to obtain a membership functions with a soft shape, yet continuous. Monotonicity is guaranteed by the use of the *hyman* option of the `splinefun` function of the `stats` package; see the associated *vignette* for details. An alternative interpolation method supported is linear piecewise interpolation, provided by the subclass `PiecewiseLinearFuzzyNumber` of the `FuzzyNumbers` package. The membership function is built by connecting a monotone[2] sequence of points $(x_i, \mu(x_i))$ with straight lines.

Parallelism. The most computationally demanding task is the calculation of the α-cuts of fuzzy stationary probabilities. It requires running DE twice per each α value, and it is expected that any user samples at least ten α values, which yields 20 full executions of the optimization process for each state whose stationary probability has to be calculated. In our implementation, when α has been set, the α-cuts are computed for all the chain states indicated by the user, which means that lengthy computations have to be

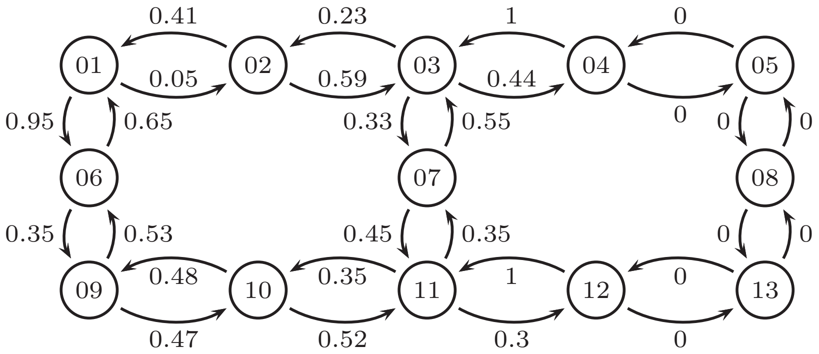
[2] $x_i < x_{i+1} \Leftrightarrow \mu(x_i) \leqslant \mu(x_{i+1})$ for the left side function and $x_i < x_{i+1} \Leftrightarrow \mu(x_i) \geqslant \mu(x_{i+1})$ for the right side

done before moving to another α value. For this reason, and since the computations with each α are independent of the rest, we have introduced parallelism at this level by using the built-in `parLapply` function of the `parallel` package over a vector containing all the α values to be sampled. Each physical core thus takes care of all the computations for a given α, i.e., runs a minimization and a maximization for every state of the chain. With this approach, possibly all the cores available will be engaged in heavy computations since the number of sampled α's will most likely be greater than the number of processors. Such coarse grain parallelization is enough, although the `DEoptim` function provides a built-in option to run in parallel as well. Parallelism is disabled initially in our package (`ncores = 1` by defult) to prevent a sudden slowdown of the computer when getting all the available processors engaged in our computations, but can be enabled by setting `ncores` to a higher number, which is strongly recommended to keep computation times within affordable limits.

6.4 Application example

Now we will show how the package can be used in a patrolling example. We will generate a realization of a 10-state Markov chain which controls the randomized movement of an autonomous robot that patrols an area, modeled as a two-dimensional grid (Figure 6.2). Each cell of the grid is a state of the chain, and the robot can only move to an adjacent cell at a turn (a discrete time instant). The movement is based on a probability distribution over the adjacent cells. Such distribution depends only on the current cell, and not on the trajectory followed by the robot in the past to reach that cell, so clearly it is a Markovian movement. It is time-homogeneous as well, since the probabilities used by the robot do not change along the time. Such models are very common in the field of autonomous robotic patrolling [6, 11]. There are two main reasons that justify the randomness of this kind of strategies:

- The area is large enough to prevent full coverage with a deterministic patrolling scheme, since some locations would remain uncovered in the sense that the time between two consecutive visits is larger than the



(a)

	01	02	03	04	06	07	09	10	11	12
01	0	0.05	0	0	0.95	0	0	0	0	0
02	0.41	0	0.59	0	0	0	0	0	0	0
03	0	0.23	0	0.44	0	0.33	0	0	0	0
04	0	0	1	0	0	0	0	0	0	0
06	0.65	0	0	0	0	0	0.35	0	0	0
07	0	0	0.55	0	0	0	0	0	0.45	0
09	0	0	0	0	0.53	0	0	0.47	0	0
10	0	0	0	0	0	0	0.48	0	0.52	0
11	0	0	0	0	0	0.35	0	0.35	0	0.3
12	0	0	0	0	0	0	0	0	1	0

(b)

Figure 6.2: (a) Optimal Markovian patrolling strategy according to game-theoretic techniques for a map with 13 cells. Reproduced from [11]. Non-reachable states 5, 8 and 13 were not considered for the Markov chain. (b) The 10x10 transition matrix.

time needed by the intruder to successfully penetrate that location.

- The robotic movement should be unpredictable when facing a full knowledge opponent, i.e., one that has perfectly learnt the robot's patrolling scheme. This situation arises when the opponent has been observing the robot's movement for a long time before choosing the cell to attack. A randomized movement leads to maximizing the robot payoff even in this case, since the only knowledge the intruder may acquire is the exact probability distribution employed by the robot, not the actual trajectory he will follow at each walk. Note that considering the strongest adversary is the common assumption in game theory as it is the worst case one may face.

Although the problem is usually solved with game-theoretic techniques[3], it represents indeed a good example of a Markov chain with very large state space (actually, proportional to the extension of the area under consideration). The concept of stationary distribution becomes relevant since it gives an idea of the locations in which the robot spends less time or, in other words, the places that have less coverage and are thus the most promising to be attacked[4]. If the assumption of perfect adversarial knowledge is eliminated, i.e., if we consider a more realistic situation in which the adversary only has a sequence of observations of the robot's movement, then what the opponent does is actually an estimation of stationary probabilities based on the observations from an unknown Markov chain that guides the robot's movement. A fuzzy estimation captures more information about the observations than a point estimate.

6.4.1 Departing from a sequence of observations

The Markov chain of Figure 6.2 depicts the game-theoretic solution [11] against a full knowledge opponent in a map with 13 cells, three of which

[3] Most often the equilibrium strategy in security games is computed using Mixed Integer Linear Programming.

[4] This deserves further discussion since a poorly covered location may not be worth attacking if the benefits of a successful attack are low for the attacker, but intuitively coverage is modeled by the stationary distribution.

are not reachable and thus discarded for our Markov chain. Assume the adversary observes a realization of 200 time instants of this chain. The R code that generates such sequence of observations and then computes the fuzzy stationary probabilities is shown below. `robotStates = c("01", "02", ..., "12")` is a 10-component string vector with the names of the states, and `transRobot` is the transition matrix of Figure 6.2. Both variables are defined in an `.Rda` file attached to the package. The `states` argument is not specified in the call to `fuzzyStationaryProb`, which means the fuzzy stationary probabilities of all the states should be computed.

```
R> library(‘‘markovchain")
R> mcPatrol <- new("markovchain", states = robotStates, byrow = TRUE,
+    transitionMatrix = transRobot, name = "Patrolling")
R> set.seed(666);
R> simulatedData <- rmarkovchain(n = 200, object = mcPatrol,
+    t0 = sample(robotStates, 1))
R> mcfit = markovchainFit(simulatedData)
R> vsteady = steadyStates(mcfit$estimate)
R> quadratic = fuzzyStationaryProb(simulatedData, list(verbose = TRUE,
+    regression = "quadratic", ncores = 4), step = 0.1);
```

```
Parallel computation of a-cuts in progress...finished successfully
(elapsed:  194.04 s.)
Applying quadratic regression to obtain the membership functions of
fuzzy stationary probabilities...
Fitting memb.func. for state: 01  02  03  04  06  07  09  10  11  12
```

Now we demonstrate different types of regression to fit the membership function of the fuzzy stationary probabilities.

```
R> linear = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+    regression="linear", ncores = 4), step=0.1)
R> cubic = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+    regression = "cubic", ncores = 4), step = 0.1)
R> gaussian = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+    regression = "gaussian", ncores = 4), step = 0.1)
R> splines = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+    regression = "spline", ncores = 4), step = 0.1)
R> pwlinear = fuzzyStationaryProb(simulatedData, list(verbose = FALSE,
+    regression = "piecewise", ncores = 4), step = 0.1)
```

Finally we are going to depict in a figure the α-cuts of the fuzzy stationary probabilities computed by our program, and the functions fitted to them. The code relies on the plot function for `FuzzyNumber` objects that were created for each state as a result of the regression. In order to compare to the output given by the `steadyStates` function of the `markovchain` package, dashed lines have been drawn at the stationary probabilities calculated by that function, using function `abline`.

```
R> m <- matrix(c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ,11, 11), nrow = 4,
+    ncol = 3, byrow = TRUE)
R> layout(mat = m, heights = c(0.25, 0.25, 0.25, 0.25))
R> for (i in robotStates){
+    par(mar = c(4, 4, 2, 1))
+    plot(linear$fuzzyStatProb[[i]], col = "blue", main =
+    paste("State ", i), cex.lab = 1.1, lwd = 2);
+    plot(quadratic$fuzzyStatProb[[i]], col = "red", cex.lab = 1.1,
+    lwd = 2, add = TRUE);
+    plot(cubic$fuzzyStatProb[[i]], col = "springgreen4", cex.lab =
+    1.1, lwd = 2, add = TRUE);
+    plot(gaussian$fuzzyStatProb[[i]], col = "black", cex.lab = 1.1,
+    lwd = 2, add = TRUE);
+    plot(splines$fuzzyStatProb[[i]], col = "orange", cex.lab = 1.1,
+    lwd = 1, add = TRUE);
+    abline(v = vsteady[1,i], lty = "dashed");
+    points(linear$acuts[[i]]);
+    }
R> plot(1, type = "n", axes = FALSE, xlab = "", ylab = "")
R> plot_colors <- c("blue", "red", "springgreen4", "black", "orange")
R> legend(x = "top", inset = 0,
+    legend = c("Linear", "Quadratic", "Cubic", "gaussian", "Spline"),
+    col = plot_colors, lwd = 2, cex = 1, bty = "n", horiz = FALSE)
R> summary(quadratic)
```

```
 . Fuzzy stationary probabilities of a Markov chain with 10 states
 . Probabilities have been computed for states: 01
 . Number of input observations: 200
 . Parameters:
        Step size: 0.1
        Execution was done in parallel ( 4 cores used )
        Regression curve for output membership functions: quadratic.
        To retrieve the results use $fuzzyStatProb and $acuts with
        this object
 . Computation of alpha-cuts took 194.04 seconds
 . Membership functions regression took 34.71 seconds
```

Let us retrieve the FuzzyNumber object corresponding to the stationary probability of state 01, and the α-cuts from which such number was built. The α-cuts are returned as a data.frame object in which the first column represents the probability and the second, the membership degree α. Note there are always two rows (rows 1 and 10, 2 and 11, etc) with the same y (membership) value, one for the lower bound and the other for the upper bound of that α-cut. This way, the limits of the α-cuts can be plotted as if they were 2D points.

```
R> quadratic$fuzzyStatProb[["01"]]

Fuzzy number with:
   support=[0, 0.294939],
      core=[0.15054, 0.15054].

R> quadratic$acuts[["01"]]

            x     y
1  0.0001480569 0.001
2  0.0336437600 0.100
3  0.0510534166 0.200
4  0.0725788506 0.300
5  0.0936139309 0.400
6  0.1111090020 0.500
7  0.1300688992 0.600
```

```
8  0.1387885412 0.700
9  0.1505397380 0.999
10 0.3822915287 0.001
11 0.3046761228 0.100
12 0.2549764457 0.200
13 0.2434278879 0.300
14 0.2104871378 0.400
15 0.1949950486 0.500
16 0.1763232720 0.600
17 0.1625293319 0.700
18 0.1505397380 0.999
```

An important remark should be made here. There are no output α-cuts for $\alpha = 0.8$ and $\alpha = 0.9$. This is due to the fact that the intervals of the input interval matrices were very small, and thus the minimization and maximization problems for each of these values have all the same solution. It is because the multinomial confidence intervals are very small when the required confidence level is also very low. This happens for $\alpha = 0.8, 0.9$ and 0.999. In practice, the corresponding output α-cuts are points rather than intervals, and actually it is the same point for those three values of α. By the representation theorem [62], the membership degree of all the elements within such very small output α-cuts is actually the greatest of them, i.e., 0.999, so the others can be discarded and the 0.999 is the only one retained. In our implementation, before carrying the regression we always discard those α-cuts such that the distance from the lower or upper bounds of the α-cut to the fuzzy number vertex (which is the point estimate of the stationary probability) is less than 0.005.

The graphical output of the above code is the plot of Figure 6.3. A number of details can be observed. The gaussian curve usually provides the best fit, and this is actually the reason for which it was included as an option in our package. However, it has the most complicated expression. Cubic fitting also fits quite well. Regarding the shape of the figures, it is clear that the fuzzy numbers are in general not symmetric around the point estimate as the vertex is sometimes very close to 0 or 1 so the points cannot spread much in that side.

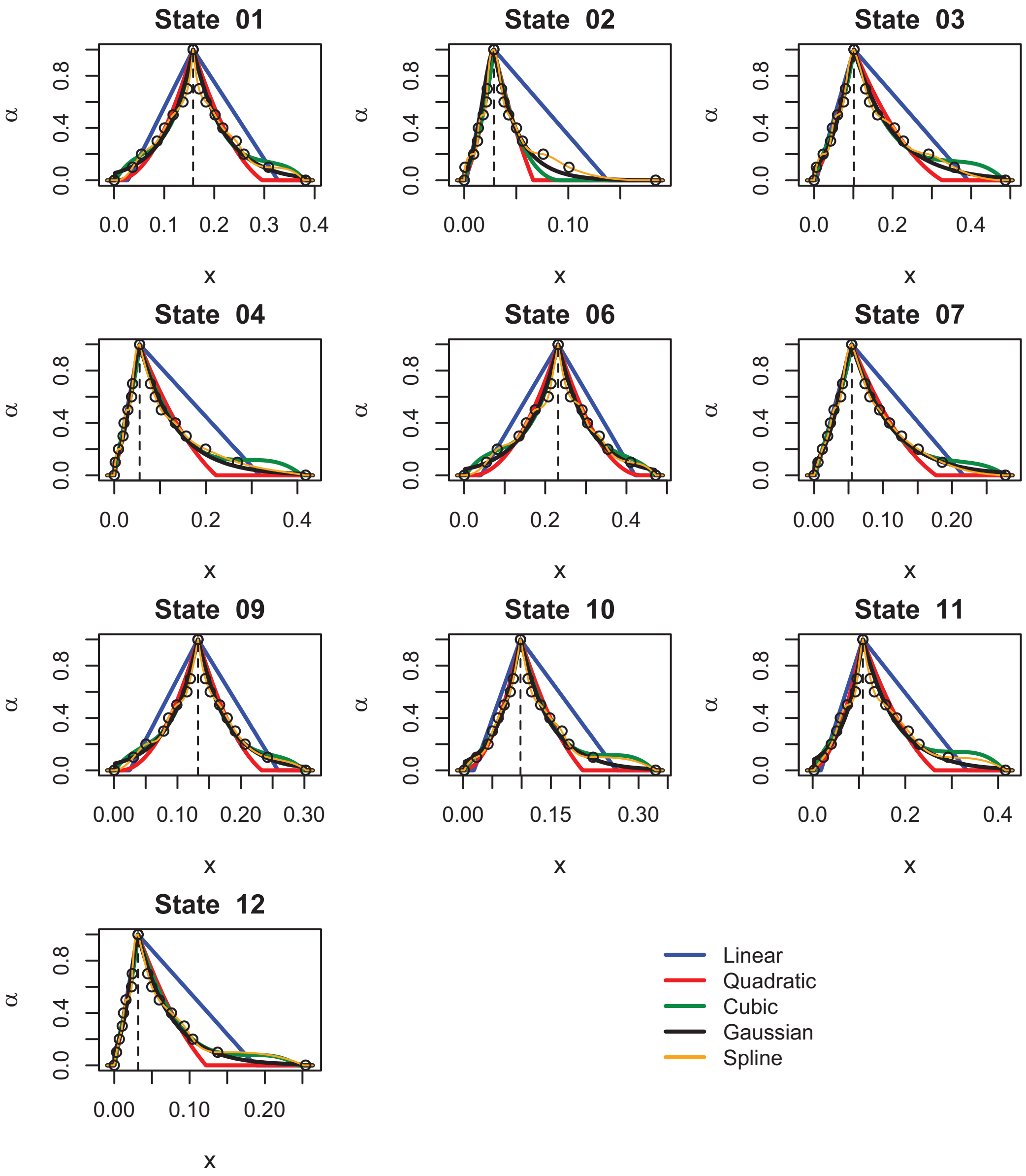


Figure 6.3: Fuzzy stationary distribution of the Markov chain of Figure 6.2, computed using 200 observations (several fitting curves available are shown). Dashed lines correspond to stationary probabilities returned by the function `steadyStates` of `markovchain`.

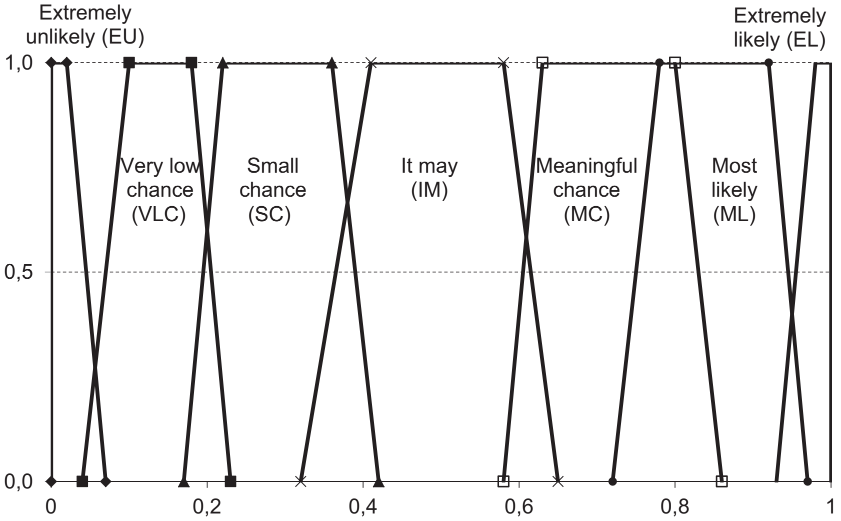
Moreover, fuzzy stationary probabilities provide more information than a point estimate based on the transition proportions from the sequence of observations. A comparison between stationary probabilities of states 6 and 8 serves to illustrate how beneficial it is to have fuzzy estimations. According to point estimates (which are the central point of the output fuzzy numbers), the stationary probability of being at state 6 is smaller than that of state 8. However, the α-cuts of fuzzy stationary probability for state 6 are wider and the fuzzy number is more right-skewed than state 8, indicating there is more uncertainty on this value and suggesting that it could be actually greater. In fact, the true stationary probabilities are $\pi_6 = 0.1946, \pi_8 = 0.1154$, and therefore $\pi_6 > \pi_8$. Note, additionally, that the output of the function `steadyStates` from `markovchain` consists of punctual estimates of stationary probabilities. The package is able to compute bootstrap confidence intervals for the transition probabilities but not for the stationary probabilities. Therefore fuzzy transition probabilities are more informative. As could be expected, the point estimates of stationary probabilities given by `steadyStates` match the 1-cuts of our fuzzy stationary probabilities.

6.4.2 Departing from user-specified fuzzy transition probabilities

Now we depart from a matrix of fuzzy numbers that constitute the fuzzy transition probabilities and are given directly by the user. In this case we assume they are linguistic probabilities as each of them is represented by a (meaningful) linguistic term: EU ("extremely unlikely"), VLC ("very low chance"), SC ("small chance"), IM ("it may"), MC ("meaningful chance"), ML ("most likely"), EL ("extremely likely"). However, we could have chosen the names to be meaningless labels like "L_1", "L_2", ..., "L_l" where l stands for the number of different fuzzy numbers employed in the matrix. The linguistic transition matrix created for our problem (Figure 6.4(a)) is similar to that in Figure 6.2(a). We have used Trapezoidal Fuzzy Numbers (TrFNs) in accordance to previous studies about the probability ranges that human people associate with each linguistic expression [17] (Figure 6.4(b)). Every row of the fuzzy matrix fulfills the condition of being a well-formed probability distribution in the sense stated in Section 6.2.4.

	1	2	3	4	5	6	7	8	9	10
1	-	VLC	-	-	ML	-	-	-	-	-
2	IM	-	IM	-	-	-	-	-	-	-
3	-	SC	-	SC	-	IM	-	-	-	-
4	-	-	EL	-	-	-	-	-	-	-
5	MC	-	-	-	-	-	SC	-	-	-
6	-	-	IM	-	-	-	-	-	IM	-
7	-	-	-	-	IM	-	-	IM	-	-
8	-	-	-	-	-	-	IM	-	IM	-
9	-	-	-	-	-	SC	-	IM	-	SC
10	-	-	-	-	-	-	-	-	EL	-

(a)



(b)

Figure 6.4: (a) User-specified linguistic transition matrix. (b) Associated TrFNs.

The code to compute the fuzzy stationary probabilities from this matrix is the following. Variable `linguisticTransitions` is a matrix of labels (strings) defined in an `.Rdata` file of the package. Its entries should match the names of `allnumbers`, which is a list of `FuzzyNumbers`.

```
R> EU = TrapezoidalFuzzyNumber(0, 0, 0.02, 0.07);
```

```
R> VLC = TrapezoidalFuzzyNumber(0.04, 0.1, 0.18, 0.23);
R> SC = TrapezoidalFuzzyNumber(0.17, 0.22, 0.36, 0.42);
R> IM = TrapezoidalFuzzyNumber(0.32, 0.41, 0.58, 0.65);
R> MC = TrapezoidalFuzzyNumber(0.58, 0.63, 0.8, 0.86);
R> ML = TrapezoidalFuzzyNumber(0.72, 0.78, 0.92, 0.97);
R> EL = TrapezoidalFuzzyNumber(0.93, 0.98, 1, 1);
R> allnumbers = c(EU, VLC, SC, IM, MC, ML, EL);
R> names(allnumbers) = c("EU", "VLC", "SC", "IM", "MC", "ML", "EL");
R> rownames(linguisticTransitions) = robotStates;
R> colnames(linguisticTransitions) = robotStates;
R> linear = fuzzyStationaryProb(linguisticTransitions, list(verbose =
+    TRUE, regression = "linear", ncores = 4, fuzzynumbers =
+    allnumbers), step=0.1)
```

The code that plots the results (Figure 6.5) is similar to the previous section so we will not reproduce it again. Notice the perfectly trapezoidal shape of the output fuzzy sets defined by its α-cuts, mirroring the shape of the input probabilities. In this case, package `markovchain` cannot deal with uncertain transition probabilities directly, in absence of data.

6.4.3 On the reduction of uncertainty with more observations

It is clear that the longer we observe a chain, the more certain we are about its behaviour. The causes of this intuitive idea can be mathematically explained as follows. When the number of observations increases, the simultaneous confidence intervals for multinomial proportions are smaller, representing a less uncertain value or, equivalently, a more accurate estimation. Since such intervals are indeed the α-cuts of our fuzzy transition matrix at the input, this means that the membership functions of our fuzzy transition probabilities are narrow and sharper in shape (they tend to be more singleton-like). Equivalently, as the intervals act as bound constraints for our optimization problems to compute the minimum and maximum stationary probabilities that are possible for each of the chain states, the feasible region of those optimization problems gets smaller and therefore, the minimum and the

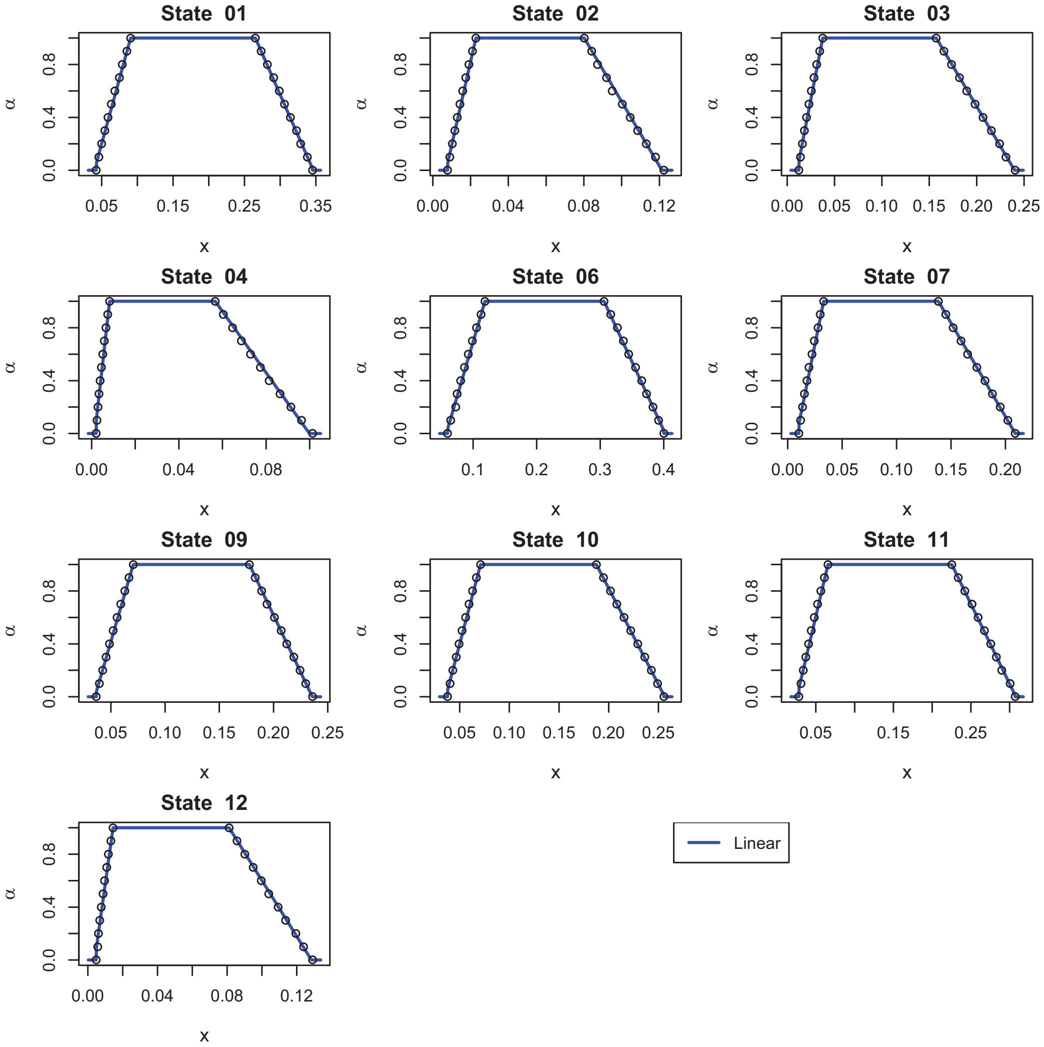


Figure 6.5: Fuzzy stationary distribution of the Markov chain of Figure 6.2, computed from user-specified fuzzy transition probabilities.

maximum found are closer to each other. This leads to smaller output α-cuts, which yields narrow, sharper fuzzy stationary probabilities as they also more singleton-like, mirroring the same phenomenon we have described for the input fuzzy transition probabilities.

In order to check this empirically, the following experiment has been designed. We have drawn a sequence of 1000 observations of the same Markov chain. From them, we take only the first 100 observations and use them to estimate the stationary probability of one of the states. After that, we do the same with the first 300 observations, then 500 and then the whole sequence of 1000 observations. The aim is to compare the uncertainty of the output fuzzy stationary probabilities obtained with every sequence. We have focused only on state 1, because the resulting fuzzy number in Figure 6.3 is quite wide so there is space for improvement. We have used a step size of 0.1 and piecewise linear fitting for the output membership function. The R code written (run just after the code shown above) for this experiment is shown below.

```
R> mcPatrol <- new("markovchain", states = robotStates, byrow = TRUE,
 transitionMatrix = transRobot, name = "Patrolling")
R> set.seed(666);
R> data4 = rmarkovchain(n = 1000, object = mcPatrol, t0 = sample(
+    robotStates, 1))
R> data1 = data4[1:100];
R> data2 = data4[1:300];
R> data3 = data4[1:500];
R> pw1 = fuzzyStationaryProb(data1, list(states = "01", regression =
+    "piecewise", ncores = 4), step = 0.1)
R> pw2 = fuzzyStationaryProb(data2, list(states = "01", regression =
+    "piecewise", ncores = 4), step = 0.1)
R> pw3 = fuzzyStationaryProb(data3, list(states = "01", regression =
+    "piecewise", ncores = 4), step = 0.1)
R> pw4 = fuzzyStationaryProb(data4, list(states = "01", regression =
+    "piecewise", ncores = 4), step = 0.1)
R> plot(pw1$fuzzyStatProb[["01"]], cex.lab = 1.1, lwd = 2, main =
+    "State 01")
R> plot(pw2$fuzzyStatProb[["01"]], add = TRUE, cex.lab = 1.1, lwd = 2,
+    col = "springgreen4")
```

```
R> plot(pw3$fuzzyStatProb[["01"]], add = TRUE, cex.lab = 1.1, lwd = 2,
+    col = "red")
R> plot(pw4$fuzzyStatProb[["01"]], add = TRUE, cex.lab = 1.1, lwd = 2,
+    col = "blue")
R> plot_colors <- c("black", "springgreen4", "red", "blue")
R> legend(x = "topright", inset = 0.1, xjust = 1,
+    legend = c("100 observations", "300 observations",
+    "500 observations", "1000 observations"), col = plot_colors,
+    lwd = 2, cex = 1, horiz = FALSE)
R> abline(v = 0.14045093);
```

Now we will retrieve the support (0-cut) of the `FuzzyNumber` objects representing the fuzzy stationary probability of state 01 departing from different number of observations in order to check whether increasing the observations decreases the uncertainty (hence the amplitude of the fuzzy number).

```
R> support1 = alphacut(pw1$fuzzyStatProb[["01"]], 0);
R> support2 = alphacut(pw2$fuzzyStatProb[["01"]], 0);
R> support3 = alphacut(pw3$fuzzyStatProb[["01"]], 0);
R> support4 = alphacut(pw4$fuzzyStatProb[["01"]], 0);
R> cat("Support amplitude of stat.prob. of state 1 with 100 obs: ",
+    abs(support1[1] - support1[2]), "\n",
+    "Support amplitude of stat.prob. of state 1 with 300 obs: ",
+    abs(support2[1] - support2[2]), "\n",
+    "Support amplitude of stat.prob. of state 1 with 500 obs: ",
+    abs(support3[1] - support3[2]), "\n",
+    "Support amplitude of stat.prob. of state 1 with 1000 obs: ",
+    abs(support4[1] - support4[2]), "\n")
```

```
Support amplitude of stat.prob. of state 1 with 100 obs:  0.3874938
Support amplitude of stat.prob. of state 1 with 300 obs:  0.325542
Support amplitude of stat.prob. of state 1 with 500 obs:  0.2910723
Support amplitude of stat.prob. of state 1 with 1000 obs:  0.2541191
```

The results are plotted in Figure 6.6, which confirms our hypothesis. The black fuzzy number is the widest, although (by chance) the center is already a fairly good approximation of the true π_1. The green fuzzy number is a bit

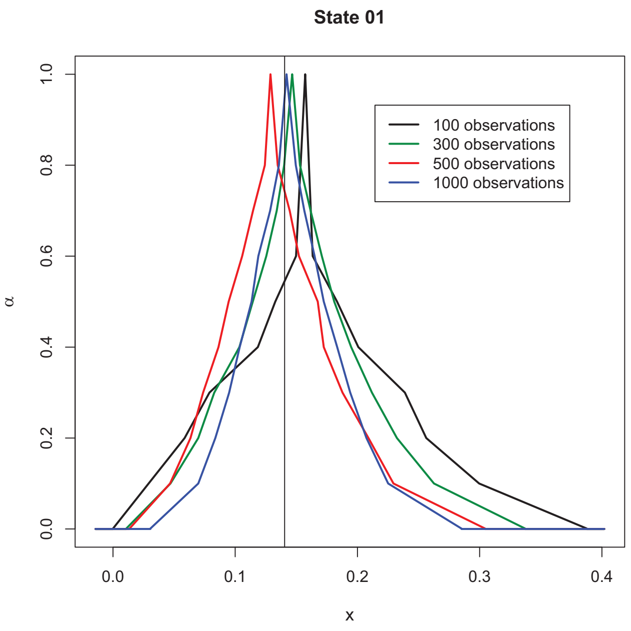


Figure 6.6: Fuzzy stationary probability of state 1 (piecewise linear fitting) obtained after 100, 300, 500 and 1000 observations. The vertical line is the true $\pi_1 = 0.14045093$.

narrower. The red and blue fuzzy numbers exhibit a sharper shape, indicating they carry less uncertainty. The blue one, computed after a sequence of 1000 observations, is the narrowest, hence carries the least uncertainty, and is centered around a good punctual approximation. The phenomenon can be checked in the amplitudes of the base (support) of the membership functions, as shown at the end of the above fragment of code. The explanation for this is that the more observations we have, the narrower the multinomial confidence intervals for the transition probabilities, or equivalently, the narrower the α-cuts that constitute the fuzzy number of the stationary probability.

As a concluding remark concerning the uncertainty represented by a fuzzy number, it should be pointed out that, in some cases, fuzzy numbers are asymmetric as they tend to indicate where the true value could be located. This can be observed in some of the plots of Figure 6.3, specially in states 2 and 4. In both cases, the α-cuts have very large upper bounds at the base, indicating that values greater than the center are more likely than those smaller than it, i.e., the *possibility* that values located in the right side are the true stationary probability is greater. This is the kind of information that crisp numbers cannot provide.

6.5 Conclusions

In this chapter we have proposed a method to obtain more information from the observations of the randomized (Markovian) behaviour of a moving agent. We have adopted the role of a potential intruder who first conducts surveillance about the defender in a conflict situation and then decides the best location to attack. We have approached this problem as the estimation of stationary probabilities of an unknown Markov chain from a sequence of observations of the patroller's behaviour. The transition probabilities are first estimated using fuzzy numbers, and then we carry all the computations with them to obtain fuzzy stationary probabilities. Our work serves as a proof of concept of the method proposed in [23] and, at the same time, provides the first freely available, ready-to-use implementation of a fuzzy Markov chain estimation procedure in a widely extended programming environment, namely the R language.

The R package developed for this purpose, `FuzzyStatProb`, has a number of advantages. The main of them is the ease of use: it just requires to have a sequence of observations represented as integers. In addition, it provides great flexibility as it allows the user to specify the kind of regression to be used for the output membership functions, and it can run in parallel, thus exploiting the computational facilities of modern multi-core architectures of conventional computers without any additional software requirement for parallelism. The results rely on the `FuzzyNumbers` package that has been recently published, is under active development and will most likely be adopted by the fuzzy community working on the R implementation of other fuzzy tools and methods.

The implementation has demonstrated the usefulness of the method, and the advantages of having fuzzy estimations for stationary probabilities. Fuzzy numbers are able to better capture uncertainty on the output, and this is indicated by the asymmetric, wider α-cuts. Defuzzification can better approximate the true crisp stationary probabilities. Furthermore, the method paves the way to the computation of linguistic stationary probabilities, which are more intuitive than crisp values and may be in turn the only type of output suitable when the input information about the chain is given in a linguistic manner, which is inherently uncertain.

Chapter 7

A patrolling model for a UAV protecting an area against terrestrial intruders

This chapter, which constitutes the last contribution of the thesis, presents an application of adversarial decision making to the defense of an environment. Here we move from imitation games to the security games framework. We describe a novel game-theoretic model of an Unmanned Aerial Vehicle (UAV) patrolling over an industrial area to protect it against potential intruders. The equilibrium solution prescribed by Game Theory provides the optimal patrolling route in terms of the benefits it may achieve. Along the chapter we discuss several computational issues that arise when attempting to compute the equilibrium and outline some possible remedies.

The contribution of this chapter consists of (i) a mathematical model of the situation, including realistic physical limitations of the UAV and the intruders, (ii) a mathematical program to compute the optimal patrolling strategy, and (iii) an approximate approach to solve large instances of the problem, since the mathematical program is unfeasible when scaling the problem to a real situation. The computational experiments concerning the third contribution are yet to be done, but we provide enough details to fully implement the proposal.

7.1 Problem statement

Suppose an environment, possibly an urban-like area, with a number of locations containing valuable items that need to be defended against terrestrial intrusions (e.g., prevented from being robbed, from suffering terrorist attacks, and so on). The accessibility of the locations depends on the topology of the urban area, defined by streets and buildings. Some of the locations are more valuable for the intruder than others, hence the adversarial preferences should also be taken into account when designing a solution. It is assumed that the intruder enters the environment from an access point, moves at a certain speed throughout the streets to reach the target point, waits there for some time which depends on the concrete target, and then penetrates the target. If this is accomplished without being detected, the UAV will have failed in protecting the target.

We have a UAV (in the future, the research might be extended to a team of UAVs) flying over the environment and equipped with a video camera that is able to send what it records in real time to a ground station, where either a human operator or an automatic image analysis system can detect the presence of the intruder in the video (Fig. 7.1). The UAV uses a battery with limited duration. For this reason, we define a ground base in the environment, where the UAV must periodically return to change the battery. This means the patrolling routes performed by the UAV must depart from the base and return to it before the battery goes flat.

This problem statement admits a number of user-specified parameters, such as the preferences over the targets, the time required to penetrate each target, the speeds of the UAV and the adversary, the UAV perception radius, and so on. The challenge consists, firstly, in designing a theoretical framework that captures this situation as realistically as possible, and secondly, in solving the model proposed, as the formal solution is far from easy once the problem has been mathematically formalized. The patrolling model proposed is developed in Section 7.3.

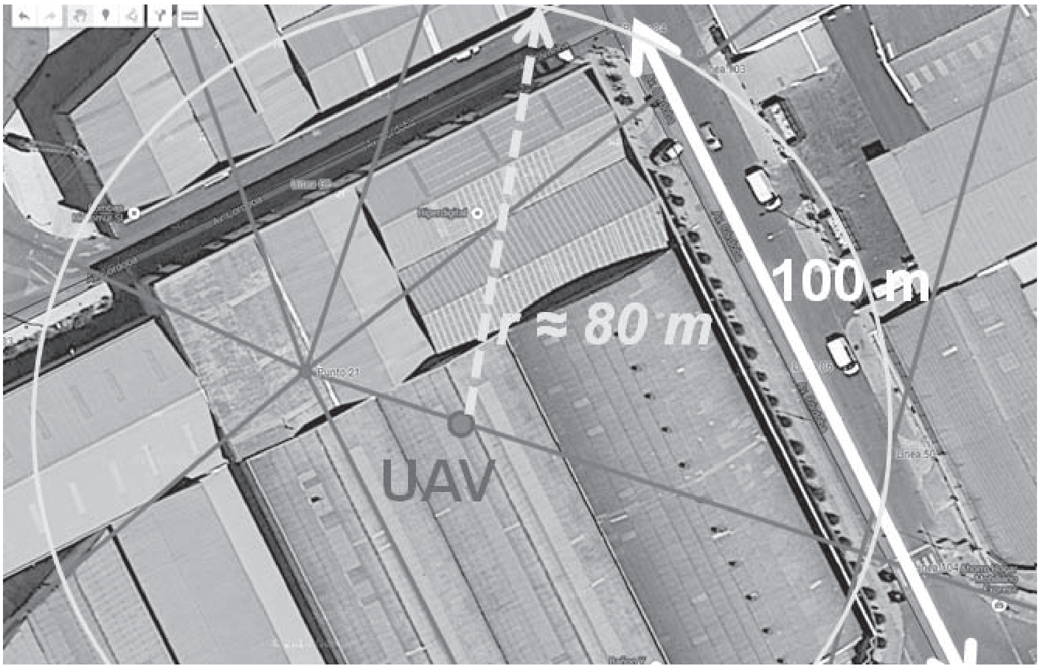


Figure 7.1: View of an industrial area from a UAV flying at an altitude of about 70 meters.

7.2 State of the art

Since we will approach our problem from a Security Game perspective, please refer to section 2.2 for a general introduction to Security Games and patrolling models. Here we review in more detail those works that are more closely related to our proposal, in order to emphasize the similarities and differences.

The series of papers which bear more resemblance to our approach are those by Basilico, Gatti and Amigoni; see [13] and references therein. They introduced the BGA model in [11] for a single-agent setting. Such model considers arbitrary topologies, abstracted as a 2D grid. Space is discretized in cells and time is discretized in turns. Both the patroller and intruder move in the same grid at a constant speed of one cell per turn, and in each turn the patroller can sense the cell where it is located. Their proposal explicitly takes into account patroller's and adversary's payoffs separately, and the problem is modeled in game-theoretic terms as a leader-follower game whose solution is the Stackelberg equilibrium. Our proposal is directly inspired by posterior refinements of such model presented in [14], where the intruder's actions are paths from access areas to the target locations. Note, however, that this model does not admit different movement constraints for the players as they both move on the ground. In addition, no physical considerations are made, such as what happens while the agents are moving between locations, the possibility of different speeds, detection radius, and so on.

Other important work to which our model is closely related is [88]. The authors propose a two player model of area transit, in which a player must cross a graph from one side to the other while the adversary tries to intercept him by doing closed routes (called closed walks here) that start and depart in a base node. This is applied to a transit simulation in the Gulf of Arden, where pirate ships (patroller) try to intercept vessels that need to traverse the area. The situation is modeled as a zero-sum normal-form game. Differently from other works, the patroller does not just indefinitely patrol along the graph but in closed walks that return to the base, capturing patroller's limited autonomy (battery, fuel, etc) as we will do in our proposal. Such approach yields a huge action set for the patroller which makes the computation of Nash equilibrium infeasible with classic techniques, as also happens in our proposal; hence the authors used oracle techniques for this task. In [19], the authors try to provide insights on a basic issue, namely strategy representation in patrolling models. An interesting remark found there is that when a player starts traversing an arc between two targets, it is engaged in that movement until reaching the destination, unless intermediate decision nodes are added to the graph. This can be also observed in the UAV's graph of our model.

Finally, one key feature of our model, namely the continous-time nature of the attacks, has been studied in [34]. The authors develop a game-theoretical model to protect a mobile ferry moving in a straight line repeatedly, escorted by patrol boats which are able to detect an intruder within a fixed detection radius. The model has been deployed and is currently being employed in the Staten Island Ferry in New York. The way to solve continuous-time games like these can be summarized as finding the key time instants where the behaviour of the game and expected payoff changes. Attacking between two of these points makes no difference in terms of the attacker's payoff, therefore the interest is in determining those points and computing the outcome in each time interval. We adopt a similar approach in the present work.

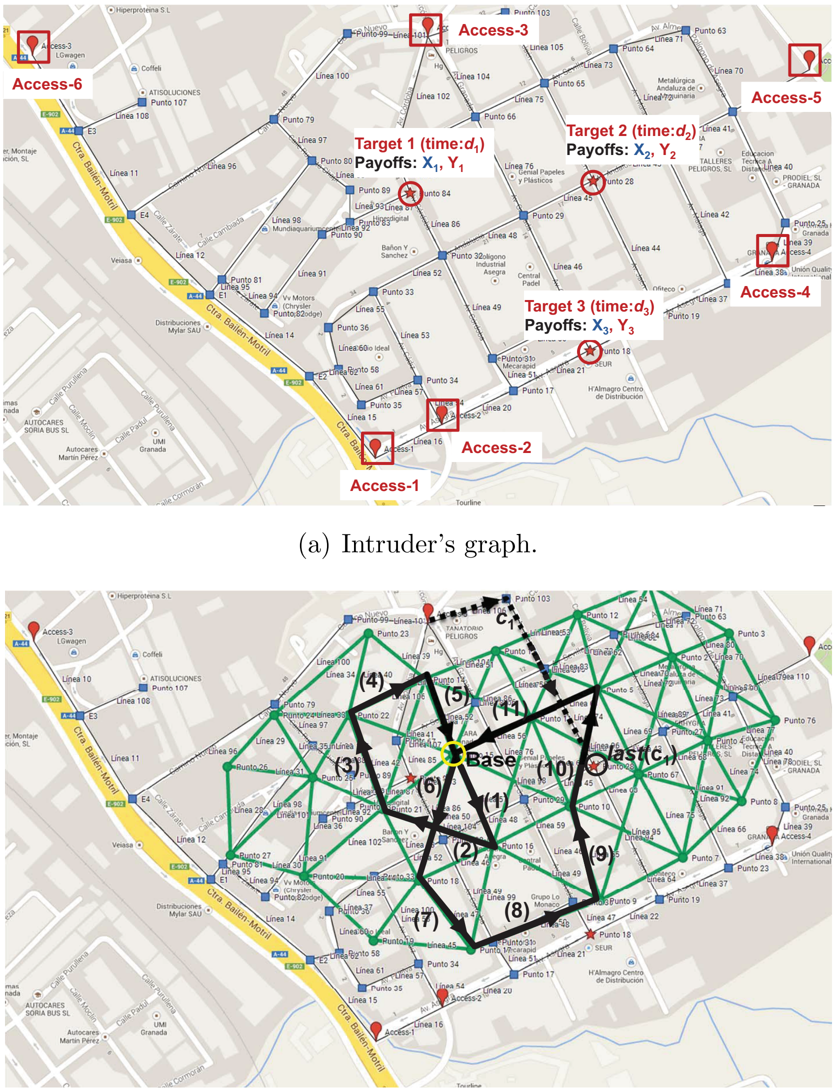
7.3 Methodology

7.3.1 Game-theoretic Formulation

The problem stated in Section 7.1 can be formalized as follows. The environment can be viewed by the intruder (referred to with subscript or superscript e in the equations) as a weighted graph $G_e = (V_e, E_e)$ where the arcs represent the streets and the nodes represent: either target locations, crossroads between streets, or access points to the environment from outside (Fig 7.2(a)). Weights on the arcs measure the spatial distance between nodes. Due to how we will bind this graph to the UAV's movement, it is mandatory that the graph mirrors the spatial disposition of the environment, i.e. the nodes are defined by their Cartesian coordinates, and the weights are computed as the Euclidean distance between them.

That being said, the crucial aspect of the model is the following: the environment to be covered is so large that no deterministic route can protect all the targets all the time, in a way the UAV can guarantee that the time between two consecutive visits to every target is smaller than their respective penetration times. For that reason, what we are searching for is a randomization over routes that at leasts guarantees a strictly positive capture probability for any action of the intruder.

Regarding the UAV (referred to with subscript or superscript u in the equations), it moves in the air independently of the intruder's graph (Fig. 7.2(b)), hence the movement has no constraints except for the need to return to the base node before the battery goes flat. As we will explain, the UAV flies high enough so that windows of the buildings cannot be filmed by the camera, hence privacy is not an issue. We need to define flying routes for the UAV to guarantee the maximum level of protection for the environment, assuming that the UAV will be flying continuously over the area. Due to the absence of flying constraints, we have the ability of defining any graph for the movement, even with curve-like trajectories, as long as the movement



(a) Intruder's graph.

(b) Manually designed UAV's graph (green), and one sample intrusion path c_1. The solid line represents the first two closed walks (1) - (5) and (6) - (11) of a UAV's route. The numbering indicates the order in which the UAV traverses the arcs. Each closed walk is supposed shorter than the UAV's battery.

Figure 7.2: UAV's and intruder's sample graphs on an industrial area.

can be characterized in terms of nodes and arcs because the UAV will be randomizing between routes defined by nodes and arcs, even when no intruder is present. Although the problem of defining the appropriate graph for the UAV in the air poses interesting challenges by itself, at this stage of research we assume the UAV's fly graph $G_u = (V_u, E_u)$ has been previously fixed and is independent of the intruder's graph. Hence, we assume the UAV moves at constant speed in straight lines between two flight nodes defined in the air, and only changes the direction in the nodes [1].

The UAV is equipped with a video camera which records video images of the ground that are sent in real time to a human operator that visualizes them in a ground base. We model this as a circular detection area centered on the UAV with a configurable detection radius ϵ which depends on the UAV altitude and camera specifications (mainly resolution and UAV speed, so it is possible for the human operator to identify a moving intruder in the video). One of the nodes of the UAV's graph is labeled as the *base* node, meaning that the patrolling routes must depart and arrive to that node since it is the place where the UAV recharges the battery[2], which takes a time t_{ch}.

The problem will be approached in an adversarial manner, which means that we explicitly take into account players' preferences over the target nodes (we assume the crossroad nodes report no payoff as they cannot be attacked). The higher the value of a location for the adversary, the higher the adversarial payoff we assign to that node in the intruder's graph. Crossroads are assumed to give no reward for the players. When a location is successfully penetrated, the UAV gets punished for it, hence it receives a negative reward. It is reasonable to think that the higher the intruder's payoff for penetrating a location, the higher the punishment for the UAV because the goods existing in that location are very valuable. However, both quantities do not have to be necessarily equal-but-opposed. For instance, if the UAV belongs to a security company providing security services simultaneously to different

[1] A Wookong UAV [2] can follow a user-defined route of GPS points doing (almost) straight lines at (approximately) constant speed.

[2] The time to recharge battery is a model parameter, although we could consider two different UAVs that alternatively depart from the base as soon as the other one returns, leading to a negligible charging time.

customers having targets in the same area, then the punishment when a location is penetrated depends on the contract signed between the security company and the target's owner (i.e. a customer), which ultimately depends on the customer's budget dedicated to security in relation to the budget employed by the rest of customers in the area. This does not necessarily match the actual value for a potential intruder of the items stored in that location.

According to this, there are three real numbers associated to each target $i = 1, ..., |T|$:

- X_i is the reward (punishment) obtained by the UAV in case target i is successfully penetrated.
- Y_i is the reward obtained by the intruder in case target i is successfully penetrated.
- d_i is the time in seconds the intruder needs to remain in location i for being able to penetrate it.

In addition, X_0 and Y_0 are the payoffs attained by the UAV and the intruder, respectively, when the intruder is detected by the UAV. In this case, Y_0 can be seen as a punishment. It is expected that $X_0 > X_i$ (capturing the intruder is always better for the UAV) and $Y_0 < Y_i$ (being captured is always worse for the intruder than penetrating any location).

With the aforementioned elements, we can formalize the UAV patrolling problem as a two-player, non-constant sum strategic- form game where:

- The players are the UAV and the intruder.
- The game is played only once.
- The possible outcomes of the game are either (a) *intruder-capture*, when the intruder is detected by the UAV at any point, or (b) *penetration-i* when the intruder successfully robs target i and escapes through an access point without being detected.
- The players' utility functions are as follows: when the outcome is *intruder-capture*, the UAV receives X_0 and the intruder receives Y_0.

When it is *penetration-i*, the UAV receives X_i and the intruder receives Y_i.

- The set of actions R available to the UAV are all the possible successions of closed walks taken by the patroller within a definite time horizon of H working hours (e.g. six hours from 9 am to 5 pm), where H is a model parameter. Each of these closed walks must start and return to the base node of the UAV, and must have a duration smaller than the UAV's battery M. The total time elapsed by the patroller in the succession of closed walks must be greater or equal to H, which does not need to be a multiple of M. The patroller's speed is assumed constant, v_u.

- The set of actions $C \times [0, H]$ available to the intruder consists of all the pairs (c_l, t_k), where c_l is a path connecting an access node in the intruder's graph with a target node i, and t_k is the time instant in which the path c_l is initiated. In our case, in order to restrict the huge space of player actions, we will assume the intruder always follows the shortest path between the access point and the target selected, at constant speed v_e and without stopping at any intermediate vertex. For the attack to be completed, the intruder must traverse the path selected, remain in the target i for d_i seconds, and traverse the path back to the same access location, all without being detected and within the H hours time interval.

Note that in our model, differently from [11], intruder's actions do not take into account the UAV's position at the moment of starting the path to the target. One reason for this is that the attack must be completed within H hours, and during that period the UAV might never reach the position required to start the attack. It could also be argued that the attacker does not have the possibility to wait in an access point without being detected (not by the UAV but possibly by any other protection system) until the UAV reaches the desired position.

Utility Functions

As explained before, we suppose no deterministic route covers all the targets with the guarantee that the time between two consecutive visits to one target i is smaller than its penetration time d_i. Moreover, we assume the intruder has observed the UAV's behaviour and has learned it perfectly before launching an attack, hence the UAV must not behave in a deterministic but randomized way. We further elaborate on this assumption in the next section. This means we are searching for a probability distribution over the patrolling routes so that, even though the intruder knows this distribution perfectly, there is no attack path that guarantees a successful penetration. But certainly some paths and attack time instants have a smaller probability of being detected than some others.

Equations 7.1 and 7.2 show the utility functions for the UAV and the intruder respectively. These expressions give the *expected payoff* for each player when the UAV uses a probability distribution $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_{|R|})$ where R is the set of feasible routes being considered.[3]. In game-theoretic terms, the expected payoff for a player as a function of all players' lotteries is defined as the sum of that player's payoffs for every possible strategy profile $(r_i, (c_l, t_k)) \in R \times (C \times [0, H])$ of the game, where each term of the summation is weighted by the probability that the profile eventually arises. In our case, only the UAV randomizes over his routes as explained in section 7.3.2, hence α is referred to the $|R|$-dimensional probability distribution used by the UAV.

$$\begin{aligned} U_u(\alpha, c_l, t_k) &= X_{targ(c_l)}\big(1 - \sum_{i=1}^{R} \alpha_i \varphi(r_i, c_l, t_k)\big) \\ &+ X_0 \sum_{i=1}^{R} \alpha_i \varphi(r_i, c_l, t_k) \end{aligned} \tag{7.1}$$

$$\begin{aligned} U_e(\alpha, c_l, t_k) &= Y_{targ(c_l)}\big(1 - \sum_{i=1}^{R} \alpha_i \varphi(r_i, c_l, t_k)\big) \\ &+ Y_0 \sum_{i=1}^{R} \alpha_i \varphi(r_i, c_l, t_k) \end{aligned} \tag{7.2}$$

[3] Although $|R|$ is a very large number, we will later explain how to restrict the number of routes to a subset of the total feasible routes space.

where $targ(c_l)$ is the target location of path c_l, and $\varphi(r_i, c_l, t_k) = 1$ when route r_i can detect an adversary attempting path c_l in time t_k, and 0 otherwise. Note the detection is deterministic and thus does not depend on α. The intersection will be calculated using a mathematical abstraction of the players' movement as explained in section 7.3.2. The term $\sum_{i=1}^{R} \alpha_i \varphi(r_i, c_l, t_k)$ is the probability of the intruder being caught when it chooses path c_l at time instant t_k.

Solution Concept: Stackelberg Equilibrium

In the scenario described so far, it is clear that the UAV starts patrolling before the intruder decides to attack. Therefore, the UAV has an *strategic advantage* for arriving first to the place where the game is to be played, and thus is able to impose his (randomized) strategy. The intruder has to adapt to the strategy imposed by the leader by issuing a best response to it. This setting is known as a leader-follower game, since one of the players (the *leader*) explicitly commits to a strategy (or to a randomization among the actions) which he reveals to the opponent (the *follower*), who has a strong reason to trust the leader's commitment. In this case, the intruder trusts the leader's randomized strategy because it is what the intruder has observed and learned before attacking. This assumption is customary in security games.

The solution concept for this kind of game is the Stackelberg equilibrium, in which the leader chooses the probability distribution over the actions that maximizes his own expected payoff, taking into account what the follower's response will be in the (most commonly assumed) case the follower is a perfectly intelligent player (known as a *best-responser*). Such best response is always a single strategy, and not a randomization over the follower's strategies [30].

It should be pointed out that a lot of research is being carried for dealing with non-best-responsers, known as bounded-rationality, since it is possible to change the leader's strategy to achieve higher expected payoff when the leader faces a weaker (not fully rational) adversary. Such assumption is generally closer to the way humans perceive and act. Although this might be addressed in future refinements of our patrolling model, at this stage of research we focus on the ideal (best responser) case.

In order to compute the Stackelberg equilibrium, we proceed with a mixed-integer linear programming (MILP) approach as in [43]. Following the formulation introduced in such work, we can identify the intruder's paths c_l, along with the attack time intervals that could eventually be the intruder's best response, as the potential targets to be covered. It should be noticed that, while some time instants can be proven to never be optimal for a path c_l disregarding the patroller's randomization over the routes, some others do depend on the probability distribution and could be optimal under an adequate α. This generates a number of n_l candidate intervals for each intruder's path c_l, each of which makes a strategy $(c_l^*, [t_j^*])$ the optimal with respect to any other $(c_l^*, [t_{j'}]), j' = 1, ..., n_l, j' \neq k$ for the same path, and also with respect to any other $(c_i, [t_j]), c_i \in C, c_i \neq c_l, j = 1, ..., n_i$ for the rest of the paths. In this formulation, the $[t_j] = [t_j^{(1)}, t_j^{(2)}]$ should be understood as time intervals, since the exact instant is irrelevant for both players (given its corresponding attack path c_l), because all attack instants within an interval lead to the same payoffs. This is caused by detection function $\varphi(r_i, c_l, t_j)$ which remains constant for all the t_j within a given attack interval associated to path c_l. We will have a binary variable $a_l^j \in \{0, 1\}$ for each pair $(c_l, [t_j]), l = 1, ..., |C|; j = 1, ..., n_l$.

In the MILP formulation, we search for the probability distribution over the UAV's routes that maximizes the UAV's expected payoff, subject to the constraint that the intruder's payoff is also maximized for one of the candidate attack intervals of one attack path under such distribution (hence a best-responser should play that strategy). In formal terms, for each $c_l \in C$, find the n_l time instants t_k that can be optimal under the proper α. After they all have been found, solve the following MILP program:

$$\max_{d,k,\{\alpha_i\},\{a_l^j\}} \quad d \tag{7.3}$$

$$\text{s.t:} \tag{7.4}$$

$$\begin{aligned} & d - U_u(\alpha, c_l, [t_j]) \leqslant M(1 - a_l^j) \\ & k - U_e(\alpha, c_l, [t_j]) \leqslant M(1 - a_l^j) \\ & 0 \leqslant \alpha_i \leqslant 1 \quad , \quad i = 1, ..., |R| \end{aligned}$$

$$\sum_{i=1}^{|R|} \alpha_i = 1$$
$$a_l^j \in \{0,1\}$$
$$\sum_{l=1}^{|C|} \sum_{j=1}^{n_l} a_l^j = 1$$
$$\text{In all cases: } l = 1, ..., |C|; j = 1, ..., n_l \tag{7.5}$$

7.3.2 Mathematical Representation of Trajectories to Compute Capture Probabilities

Checking if a route detects a path

In order to compute the capture probabilities, we resort to the physical model of the UAV and the separate graphs both players move along, as well as the UAV's detection radius ϵ. Suppose we want to check whether a patroller's route r_i with $|r_i|$ arcs can detect the intruder when he does a path c_l. We are going to compute the exact time instants in which the intruder can start the path without being detected.

Focusing on the UAV, the position within each segment of a route in the UAV's graph is given by a position function $f_k(t) : \mathbb{R}^+ \to \mathbb{R}^2$ where k stands for the segment going from node (x_k, y_k) to (x_{k+1}, y_{k+1}).

$$f_k(t) = [x_k \pm v_u{\cdot}\, t{\cdot}\, cos\beta_k, y_k \pm v_u{\cdot}\, t{\cdot}\, sin\beta_k]^T, t \in [0, \tau_k^u] \tag{7.6}$$

In the above function, β_k stands for the (acute) angle formed by the segment and the horizontal axis, and $\tau_k^u = \frac{v_u}{\sqrt{(x_k - x_{k+1})^2 + (y_k - y_{k+1})^2}}$ is the time needed by the UAV to traverse the segment. The sign $\pm$ will be positive for the first component whenever $x_{k+1} > x_k$, and for the second component when $y_{k+1} > y_k$, and negative otherwise. The intruder's trajectory is represented by a similar function g on the intruder's graph in an analogous way; let g_j be its analytical expression when the intruder goes from (x_j, y_j) to (x_{j+1}, y_{j+1}). Therefore, checking whether the UAV will detect the intruder is equivalent to finding a time instant in which the Euclidean distance between both players' positions is equal or smaller than the detection radius ϵ.

Both the patroller's and defender's movement across one arc of their respective graphs can be represented as lines in a 3D space, defined by their parametric equations, in which the time acts as the free parameter. Detection thus depends on the relative positions of both lines, which is equivalent to the relative moment in which each player starts its segment. Applying the geometrical and analytical procedures described in Appendix 7.5, it is possible to compute the time sub-intervals of $[0, H]$ in which an intruder should not start the path c_l because it will be detected for sure by a patroller doing r_i. We will refer to them as the unsafe intervals for the intruder according to the patrolling route r_i. If the intruder starts c_l at any time outside such intervals, it will not be detected that route (although might be detected by other routes).

Computing the detection probability of a path by a set of routes

When considering several routes and a probability distribution α over them, an intruder starting c_l at a time instant that falls within one of the unsafe intervals of one of the routes, say r_i, will not be detected unless route r_i is actually chosen by the patroller, which may happen with probability α_i. Therefore, the probability that the intruder is detected when starting a path c_l in a time instant that is unsafe according to route r_i is α_i. When a time interval, or a part of it, is unsafe according to more than one route, say $r_{i_1}, r_{i_2}, r_{i_3}$ where $1 \leqslant i_1, i_2, i_3 \leqslant |R|$, then the probability of detecting an intruder starting his path within that time interval is the sum of those routes' probabilities, $\alpha_{i_1} + \alpha_{i_2} + \alpha_{i_3}$. Figure 7.3 represents this aggregation in the case of three routes.

Note in some cases, some intervals can never be optimal, no matter the value of α. As shown in the figure, given an attack path c_l, if the capturing probability of one interval is α_3 and the probability of another interval for the same path is $\alpha_2 + \alpha_3$ (left-most interval), then the attacker's expected payoff if he chooses c_l within the interval with probability $\alpha_2 + \alpha_3$ cannot be greater than the former, because α_3 is contained on the latter and both are referred to the same path, with the same payoff when reaching the target of that path. However, if there is an interval with detection probability α_1, another with

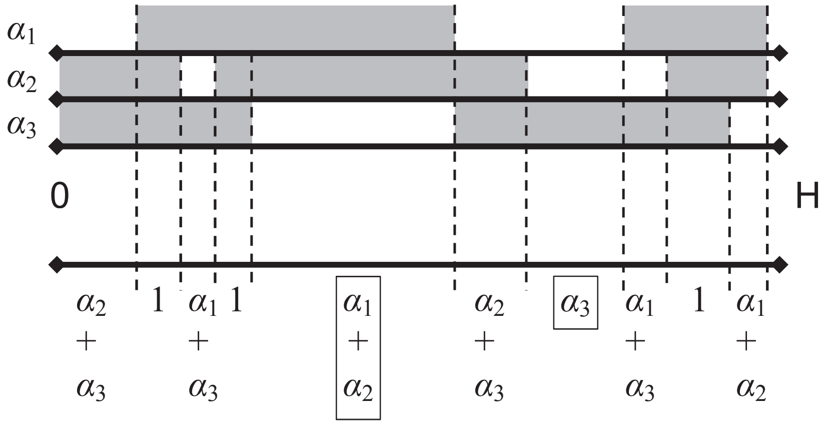


Figure 7.3: Capture probabilities for each interval in which the intruder may start his path c_l, for a setting with three routes. Probabilities that can be minimal under certain values of α have been highlighted.

$\alpha_2 + \alpha_3$, and there is no interval with probability α_2 alone or α_3 alone, then both intervals could be optimal (i.e. maximize the attacker's expected payoff) under certain values of $\alpha_1, \alpha_2, \alpha_3$. For this reason, we would need to consider two separate linear programs: one that maximizes the patroller's expected payoff while ensuring the interval with α_1 is the best for the intruder, and the other ensuring the interval with $\alpha_2 + \alpha_3$ is the intruder's best response. Finally, consider the case of two different paths $c_l, c_{l'}$ whose targets provide the attacker a payoff of $Y_{targ(c_l)}$ and $Y_{targ(c_{l'})}$ respectively, such that $Y_{targ(c_l)} < Y_{targ(c_l')}$, with detection probabilities α_1 for c_l and $\alpha_1 + \alpha_2$ for c_l' in some attack intervals. Then, both attack intervals could be optimal, each under the proper probability values, because c_l' provides a larger payoff despite having a higher probability of being captured in the corresponding interval, thus the attacker's expected payoff when attacking c_l' could be higher than attacking c_l for some values of α_1, α_2. In other words, when $Y_{targ(c_l')} > Y_{targ(c_l)}$, then $Y_{targ(c_{l'})}(1 - \alpha_1 - \alpha_2)$ could be greater than $Y_{targ(c_l)}(1 - \alpha_1)$ for some values of α_1, α_2. Again, the two paths with their corresponding intervals would constitute two separate problems.

7.4 Towards an approximate solving strategy

Implementing a simulator of the model described so far in a real scenario is not trivial. First of all, it is necessary to obtain the lengths of the arcs and the angles with the reference axes (OX and OY). The easiest way to accomplish this is to use Google Maps to draw the graphs of the UAV and the intruder, and use the exact coordinates of the vertices. After this has been done, and assuming we restrict the intruder's paths to the shortest routes between an access point and a target location, we must implement a shortest path algorithm such as Floyd-Warshall [20] to generate the intruder's set of actions. Moreover, the implementation of Algorithm 2, described in the Annex of this chapter, is difficult due to the specific data structures required to maintain the attack intervals associated to each route separately, check the overlapping and merge them. The C++ code that implements this part has been developed and is available under demand.

Once the model has been programmed, we should be able to solve it according to Eq. 7.3 and obtain the optimal mixed strategy for the UAV, i.e. the probability distribution over all the existing patrolling routes that maximizes the UAV's expected payoff. Recall that such strategy is the solution to the game as it constitutes the Stackelberg equilibrium. Unfortunately, this MILP problem is unsolvable in practice due to the huge number of feasible patrolling routes that exist even with a small time horizon, for example $H = 2$ hours. As pointed out in [43], commercial solvers run out of memory if we attempt to solve it directly, a common issue when solving the large scale problem instances that arise in the real world. Some proposals have been published to overcome this drawback, mainly based on transforming the original problem or exploiting some specific structure of the strategies so that they can be represented in a compact way that allows a fast resolution [34]. This is not the case here, since the complete route does not admit a compact representation.

In spite of this, the technique known as *column generation*, commonly used in very large linear programs, can be applied to Security Games in an efficient manner as shown in previous works [43]. The method is combined with a branch-and-price algorithm developed for general Security Games. In simple terms, column generation is aimed at considering only a subset of the defender's actions (decision variables) involved in the optimization problem. They grow this set iteratively by adding the variable which improves most the optimal value of the objective function, if possible. The branching part works on the attacker's actions (represented as binary decision variables) and establishes lower bounds on the quality of the solution. These methods are quite difficult to implement in an efficient manner, and column generation is usually cut off before it finishes by itself, so it turns out to be an approximate algorithm even though it always returns the optimal solution in theory. Unfortunately, branch-and-price is not an out-of-the-box approach as stated in [43], and the proposal presented in that work for general security games requires specific adaptation and an important programming effort to be applied to each concrete security problem.

In this work, we propose a different scheme to compute a well-performing mixed patrolling strategy for the UAV, namely the use of approximate optimization methods (metaheuristics) to find a good subset of patrolling routes involved in the optimization. The randomization employed by the UAV will then be restricted to this subset of routes. Note that the solution given by a metaheuristic differs from the solution computed by the MILP of Eq. 7.3. As we will explain, at the end of the execution of the metaheuristic, the output consists of three things: (a) a good (not necessarily optimal) subset of routes over which the UAV should randomize, (b) the UAV's expected payoff when it uses the best possible randomization over the aforementioned subset of routes, and (c) the probability distribution that provides the expected payoff described in (b).

Since we are restricting to a subset of routes, (b) is not the optimal payoff that the UAV could attain if all the patrolling routes were considered, but just a reasonably good one. On the other hand, this solution can be obtained within reasonable time and with an affordable amount of RAM memory. Items (b) and (c) are computed solving the MILP with the subset of routes of (a). In fact, during the execution of the metaheuristic, many subsets of routes are explored and assessed according to the UAV's expected payoff by applying Eq. 7.3 to that subset; we will explain this in detail later. Of course, if we had no limitations on the RAM memory available, we could solve the MILP of Eq. 7.3 considering all the possible patrolling routes, not just a limited subset, and the MILP would return the optimal payoff and the optimal probability distribution.

7.4.1 Components of a population-based metaheuristic

Metaheuristics [15, 37] allow to optimize any metric that can be computed from a feasible solution, not necessarily a mathematical function with an analytic expression but also other metrics computed through simulations or any computational procedure. Moreover, many of the difficulties posed by some mathematical functions such as being non-differentiable are overcome by metaheuristics. However, they have proved to work well from a empirical point of view, but have no theoretical guarantee of convergence to the function global optimum. Indeed, they find good solutions (local optima) within a reasonable time, where other techniques often fail to find any solution.

We now describe how to solve the problem of finding a good subset (of a prefixed size) of patrolling routes for the UAV using a metaheuristic. Due to the characteristics of the problem, we believe the class of metahueristics called *population-based* will perform better than departing from a single solution. We now describe how the basic components of a population-based metaheuristic can be adapted to this problem, namely the representation of a solution (individual), how to generate an initial set of feasible solutions, how to mutate a solution, how to "merge" two solutions to obtain two new solutions different from the original ones, and how to evaluate a solution (fitness function).

The number of routes $|R|$ to be considered in the set has to be fixed before the algorithm starts and does not change during the execution. Some experimentation is required to empirically determine a good value for this. If $|R|$ is too large, the MILP to be solved to evaluate a subset of routes (fitness function) will be more difficult and slower to solve. If it is too small, the subset is too restrictive so it will impossible to compute a good randomization to attain a reasonably large expected payoff for the UAV.

The rest of the components are the following:

- Computational representation of a solution: an individual is a set of patrolling routes $r_1, ..., r_{|R|}$ where each route r_i is composed by a succession of closed walks, each with a duration smaller than the battery, and with a route total time greater or equal to the time horizon H being considered (e.g., 2 hours, a working day, one night, etc).

- Fitness function: a feasible solution (subset of routes) will be evaluated solving the MILP of Eq. 7.3 restricted to the patrolling routes prescribed by that solution. The fitness value associated to that solution is the patroller's expected payoff under the Stackelberg equilibrium computed by the MILP.

- Crossover mechanism: two alternatives will be analyzed separately. First, binary crossover between sub-subsets of individuals, which means that K routes of the first individual, $r_1, ..., r_K$ are interchanged with the first K routes of the second individual. This way, the new individuals generated are always feasible. Secondly, it is also possible to interchange parts of each route instead of complete subsets of routes. To ensure feasibility, the routes can be crossed when they are at the base location, where the UAV charges the battery.

- Mutation mechanism: it is the mechanism by which a feasible solution is modified to obtain another feasible solution. The mutation operator in this case consists in randomly generating a new closed walk within a route, which also ensures feasibility of the new solution.

- Initial set of solutions: although many problems employ a totally random initialization, in this case it is necessary to generate the initial set of individuals in an intelligent manner to ensure that the routes are complementary, in the sense that they cover different attack intervals and attack paths. Otherwise, it is almost sure that there will be at least one attack path and one time interval that guarantees a successful penetration for the intruder.

  A way to achieve this is to use the following idea. First, N routes are randomly generated, ensuring that they are composed of feasible closed walks. Then, we compute the time intervals that are covered by 0, 1, ... and N routes, and sort them from the earliest to the latest. After this, we repeat iteratively the next actions: (a) from those intervals, select the one that is closest in time to the current moment and that is still possible to reinforce. Let c_l be its associated attack path; (b) find a patroller arc that could detect any attack arc of the path c_l; (c) wait the time necessary so that the coverage matches the interval we want to reinforce (doing some random waiting); and (d) travel to the origin vertex of that patroller arc and traverse the arc. We have to check that the routes generated this way are still feasible in terms of battery.

At the end of the process, the algorithm returns the best solution found, which is a good subset of patrolling routes that maximizes the fitness (i.e. the patroller's expected payoff under the Stackelberg equilibrium when we restrict to this subset of routes), along with this fitness value. Despite not being optimal, if the metaheuristic works well, we can expect this value to be large and not very far from the really optimal one. The module relative to the metaheuristic is not trivial but easier to implement than the simulator itself. The computational experiments concerning this part are part of the future work.

7.5 Conclusions

A novel patrolling model for a UAV defending an area against terrestrial intruders has been presented. We have provided the game-theoretical formulation as a Stackelberg game, following the framework of Security Games, and we have explained how it can be solved. The solution to the model is a randomized patrolling strategy consisting of an optimal probability distribution among the patrol routes, that constitutes the Stackelberg equilibrium to the game. As customary in Security Games, a Mixed-Integer Linear Program is used for this purpose. When scaling up the model, we have proposed a new approach based on a metaheuristic compute a good (not necessarily optimal) patrolling strategy for the UAV, since the MILP becomes unsolvable in practice.

The model poses good properties that had not been addressed up to now:

- It takes into account intruder's and patroller's payoffs (not necessarily zero-sum).

- The intruder can attack at any time (not discretized).
- It considers totally independent movement graphs for patroller and intruder.
- It takes into account the intruder's paths to target locations, and the intruder may be detected at any point of the path.
- It considers the patroller's physical limitations, such a (non-discretized) limited perception radius and limited battery.
- The problem can still be formulated as a MILP.

Some aspects that can be improved in future versions of the model are the following:

- We are not considering all possible attack paths (as this would be a huge space as well).
- None of the players is allowed to hold at any point of their paths.
- We are not considering uncertainty in the payoffs, perception radius, and speeds.

The experiments are expected to demonstrate the usefulness of the metaheuristic in this problem, which may pave the way to a broad adoption of metaheuristics to tackle large scale Security Games. This has not been done yet in the Game Theory community. Moreover, if the theoretical model shows good performance, it may be adapted to physical agents and put into practice, which will require more practical experimentation.

Algorithm 2 DetectionIntervals(r_i, C)

```
INPUT: a patrolling route $r_i =< i_1, ..., i_{|r_i|} >$, where the $i_j \in E_u$ are the arcs, and a set of attack paths $C$
        Relative offsets $\Delta_1^{k,j}, \Delta_2^{k,j}, k = 1, ..., |E_u|, j = 1, ..., |E_e|$ previously calculated.
OUTPUT: $|C|$ sets of intervals $L(c_1), ..., L(c_{|C|})$ meaning that $r_i$ will detect path $c_l$ if the path starts in a time
        instant contained by an interval of $L(c_l)$.
$t_s^u \leftarrow \sum_{a=1}^{s-1} \tau_a^u$ for every $s = 1, ..., |r_i|$. Whenever the arc represents the UAV's recharge, then $\tau_a^u = t_{ch}$.
[Recall $l_q \in E_e$ (resp. $k_s \in E_u$) is the arc of $c_l$ (resp. $r_i$) traversed in $q$-th place in $c_l$ ($s$-th place in $r_i$)]
for each $c_l =< l_1, ..., l_{|c_l|} >\in C$ do
  for $q = 1, ..., |c_l|$ do
    $t_q^e \leftarrow \sum_{a=1}^{q-1} \tau_a^e$
    $[\psi_1^{s,q}, \psi_2^{s,q}] \leftarrow [\Delta_1^{k_s,l_q} + t_s^u - t_q^e, \Delta_2^{k_s,l_q} + t_s^u - t_q^e]$ for every $s = 1, ..., |r_i|$, cutting negatives to 0
    $L(c_l) \leftarrow L(c_l) \cup \{[\psi_1^{s,q}, \psi_2^{s,q}] : s = 1, ..., |r_i|\}$
  end for
  Merge those intervals of $L(c_l)$ that overlap or subsume each other
end for
return $< L(c_1), ..., L(c_{|C|}) >$
```

Appendix A - Computing detection time intervals

Functions f_k (Equation 7.6) and g_j (defined analogously) can be viewed as the following lines:

$$f_k \equiv \begin{cases} x_k^u(t) = x_k \pm v_u \cdot cos\beta_k \cdot t \\ y_k^u(t) = y_k \pm v_u \cdot sin\beta_k \cdot t \\ z_k^u(t) = t \end{cases}$$

$$g_j \equiv \begin{cases} x_j^e(t) = x_j \pm v_e \cdot cos\beta_j \cdot t \\ y_j^e(t) = y_j \pm v_e \cdot sin\beta_j \cdot t \\ z_j^e(t) = \Delta + t \end{cases}$$

First we study the time windows in which each of the intruder's arcs would be detected by each of the patroller's arcs, as a function of the time offset Δ between the instant the patroller starts his arc and the instant the intruder does the same. This parameter can be positive or negative as it is measured from the moment at which the patroller starts his arc, which does not necessarily have to be at the beginning of the H hours working time (e.g. if the segment is not the first of the route). As the patroller's route comprises all the working horizon of H hours, the times when he traverses each arc are fixed, as opposed to the intruder's path which can be started at any time as long as there is enough time to reach the target, rob it and return to the access point. For this reason, Δ applies to the intruder only.

Given an intruder arc and a patroller arc, we can compute the values of Δ for which the intruder is not detected, meaning that if the intruder starts his arc an amount of time earlier (or later) than the patroller starts his own arc, then there is no point where the distance between both agents is smaller or equal to ϵ, hence the intruder avoids detection. Formally, define

$$d_{k,j}(t) = (x_k^u(t) - x_j^e(t))^2 + (y_k^u(t) + y_j^e(t))^2 + (z_k^u(t) + z_j^e(t))^2$$

to be the square of the distance between the patroller and the intruder when they traverse the k-th and j-th arcs respectively, $k = 1, ..., |E_u|; j = 1, ..., |E_e|$.

This is a squared function in t. We want to compute the minimum of it in the interval of interest, $[0, \min\{\tau_k^u, \tau_j^e\}]$. As this is a closed interval, the minimum is reached either at $t = 0$, $t = \min\{\tau_k^u, \tau_j^e\}$ or at the vertex of the function if it falls within the interval and the parabola is convex. In the three cases the minimum is a function of Δ^2 so we can impose that it is strictly greater than ϵ^2. Solving for Δ in the resulting inequation, we get $\Delta < \Delta_1^{k,j}$ and $\Delta > \Delta_2^{k,j}$, where $\Delta_1^{k,j} < 0$ and $\Delta_2^{k,j} = \min\{-\Delta_1^{k,j}, \tau_k^u\} \geqslant 0$. In other words, the intruder should start arc j at least $|\Delta_1^{k,j}|$ seconds before the patroller starts his k-th arc, or $\Delta_2^{k,j}$ seconds later, but not in between as it would be detected.

After computing all the $\Delta_1^{k,j}, \Delta_2^{k,j}$ for $k = 1, ..., |E_u|, j = 1, ..., |E_e|$, these relative offsets must be converted into absolute by adding them to the time the patroller starts each segment when doing his route. For instance, if a segment k of the patroller's graph is traversed in the third and seventh places of one particular route r_i, then we must compute the time instants t_3 and t_7 in which the patroller starts those arcs, as $t_s^u = \sum_{a=1}^{s-1} \tau_a^u$, and then compute the new absolute intervals $[\delta_1^{3,j}, \delta_2^{3,j}]$ and $[\delta_1^{7,j}, \delta_2^{7,j}]$ for every $j = 1, ..., |E_e|$, using the formula $\delta_{1|2}^{s,j} = t_s^u + \Delta_{1|2}^{k_s,j}$ (cutting negatives to 0) where $k_s \in \{1, ..., |E_u|\}$ is the name of the patroller's arc traversed in s-th place in the route r_i, and $s = 1, ..., |r_i|$. Note $t_1^u = 0$. This computation is not necessary for all the patrolling arcs that are not part of the route r_i being studied. The meaning of these intervals is that the intruder should not start arc $j \in E_e$ within the time $[\delta_1^{3,j}, \delta_2^{3,j}]$ nor $[\delta_1^{7,j}, \delta_2^{7,j}]$ because he would be detected by the patroller. Both intervals denote absolute time instants and thus, are contained in $[0, H]$.

The last step is to particularize the absolute δ's of route r_i computed for any intruder's arc, to the intruder's path c_l being studied. Given an intruder's arc $j \in E_e$ which appears in q-th place in the intruder's path c_l, and given the absolute δ's of intruder's arc j with respect to all the patroller's arc of route r_i, we can transform the absolute offsets that restrict the absolute moment of starting each arc appearing in second, third, ... place of the intruder's path, into absolute offsets restricting the moment in which the intruder starts the path c_l. Applying the same idea of the previous paragraph, we can compute the times at which the intruder starts the q-th arc of the path c_l as $t_q^e = \sum_{a=1}^{q-1} \tau_a^e$. With this, let $\psi_{1|2}^{s,q} = \delta_{1|2}^{s,l_q} - t_q^e$ (cutting negatives to 0) where

$l_q \in \{1, ..., |E_e|\}$ is the name of the intruder's arc traversed in q-th place in the path c_l, with $q = 1, ..., |c_l|$. Note that again $t_1^e = 0$. The ψ's impose a restriction on the time instants where the intruder can start his path; thus he should attack at any time falling outside of all the intervals $[\psi_1^{s,q}, \psi_2^{s,q}]$. As long as this is met, the exact time instant is indifferent to both the intruder and the patroller because all yield the same expected payoff. When considering several routes r_i simultaneously, if such interval does not exist because all the working day $[0, H]$ is covered by some route for the path c_l being considered, the intruder should attack during the interval of minimal probability, as explained in Section 7.3.2.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

This dissertation has focused on the study and development of adversarial decision making models and their applications to practical problems. The objectives were the following:

1. To compile the relevant references in the field of adversarial models and applications.

2. To develop new models and extensions.

3. To propose new strategies for the models, explicitly including a temporal component.

4. To assess the strategies from a theoretical and empirical point of view.

5. To propose examples of application.

With regards to objective 1, the Background chapter reviews important references in a variety of fields, with emphasis on those more closely related to the models we have proposed.

With regards to objective 2, it has been accomplished following two different approaches:

- On the one hand, we have continued the research on the imitation model [67], for which we have developed two extensions. In the first one, we have shown the importance of having an appropriate model of the adversary and the negative impact over one of the agents' payoff that a wrong assumption about the adversary can have. This extension explicitly takes into account the temporal component of a strategy in order to change the randomization that guides the behaviour of one of the agents. While this leads to an increase in the total payoff when the adversary abides to the conception we have about him, it can also lead to severe losses when it does not. In the second one, we have introduced statistical dependence between the current decision of one agent and the circumstances (modeled as the set of payoffs attainable) of the next decision to be made. We show how the same kind of time-varying behaviour is also successful in this new scenario.

- On the other hand, we have developed an adversarial model for a real situation in the context of Security Games: a UAV defending an area against intruders. The proposal takes into account realistic limitations such as the UAV's battery and speed, the existence of separate movement graphs for the UAV and the terrestrial intruders, the perception radius of the UAV (which depends on the camera it is carrying), and so on. We have shown the difficulties of computing the game-theoretic Stackelberg equilibrium of our model due to the computational intractability of the resulting optimization problem, and we have suggested heuristic techniques to tackle it.

With regards to objective 3, we have investigated new strategies that not only are randomized but change the randomization along the time, which has proven beneficial for causing deception and achieving greater payoff. This confirms that there is room for strategical manipulation when the only information accessible to the adversary consists of observations of the past actions. Therefore, when an agent is aware of a watching adversary, it is beneficial to modify the way of making decisions in order to cause deception and attain greater payoff in the long term, as achieved by the time-varying strategies.

In all our publications, we have analyzed the performance of the strategies both from a theoretical and empirical point of view in order to validate the theoretical expressions of the expected payoff, as prescribed by objective 4. We have shown that simulations are not necessary to assess two randomized strategies if the expression of the expected payoff can be obtained in an appropriate way using probability concepts. Comparing the expected payoff is a much more reliable way to determine which strategy performs better.

Finally, in connection with objective 5, we have proposed two applications: firstly, the aforementioned UAV patrolling model, and secondly, a novel mathematical technique to improve the information obtained from observations of a patrolling agent following a randomized Markovian strategy. These observations constitute the perception that the intruder has about the patroller when the former watches the latter with the aim of learning his randomization, as customary in Security Games. We have shown that, by using concepts from Fuzzy Logic and Fuzzy Numbers, the intruder gets a better approximation of the true probabilities that guide the patroller's movement, since a fuzzy number is more informative as it is able to incorporate the uncertainty about a quantity. An open-source software implementation of the method has been released.

8.2 Future Work

Both the imitation game and the patrolling models still deserve further investigation. In the imitation model, some techniques not addressed here may yield good results. For instance, the use of more sophisticated learning mechanisms (for instance, reinforcement learning) by agent T for learning S's strategy. The imitation game could be tackled as a classification problem, for which Machine Learning algorithms could be used to determine which among the possible actions will be chosen next. Finally, real-life applications of the model should also be investigated, with emphasis on computer games. A better prediction of the user's actions may lead to more intelligent adversaries and eventually, to a more enjoyable gaming experience.

With respect to the UAV patrolling model, many questions remain open. The mathematical difficulty of the optimization problem to be solved in order to compute the Stackelberg equilibrium opens the door to the application of approximate optimization techniques, such as bio-inspired metaheuristics, which have not been applied to security games up to now. The magnitude of the actions space for both players makes it necessary to consider a subset of strategies only because the complete problem cannot be solved directly due to physical memory constraints. This issue arises very often in the field of Security Games when scaling up the models to real-sized problems. Experimentation about which metaheuristic performs better and how to better adapt the problem to a specific metaheuristic should be done. Given the increasing number of domains that are being solved as security games in real life nowadays, if the use of metaheuristics proves successful in the UAV problem, it can lead to a broad adoption in other models that suffer from the same scalability problems.

Moreover, uncertainty about the physical limitations of the agents, such as the actual speed of the UAV and the speed assumed for potential intruders, or the exact duration of the battery, should be included in the model. None of these parameters can be known precisely so it seems natural to model them as fuzzy numbers, and resort to existing fuzzy linear programming approaches to solve the optimization problem and compute the Stackelberg equilibrium. This has never been done before in Security Games.

Bibliography

[1] Horizon 2020 Work Programme 2014-2015 (LEIT - Information and Communication Technologies). *European Commission Decision C(2014)4995 of 22 July 2014.*

[2] *Ground Station Wireless Data-link User Manual v.2.8.* DJI Innovations, 2013.

[3] M. Abundo and L. Caramellino. Some Remarks on a Markov Chain Modelling Cooperative Biological Systems. *Open Systems & Information Dynamics*, 3:325–343, 1995.

[4] N. Agmon, C.-L. Fok, Y. Emaliah, P. Stone, C. Julien, and S. Vishwanath. On coordination in practical multi-robot patrol. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 650–656, 2012.

[5] N. Agmon, G. A. Kaminka, and S. Kraus. Multi-robot adversarial patrolling: Facing a full-knowledge opponent. *Journal of Artificial Intelligence Research*, 42(1):887–916, 2011.

[6] N. Agmon, S. Kraus, and G. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *Proc. of the IEEE Conf. on Robotics and Automation*, pages 2339–2345, 2008.

[7] G. Akhras. *Canadian Military Journal*, pages 25 – 31, 2008.

[8] T. Alpcan. *Network Security: A Decision and Game-Theoretic Approach.* Cambridge University Press, 2010.

[9] S. Alpern and S. Gal. *The Theory of Search Games and Rendezvous*. Springer US, New York, 2003.

[10] A. O. Alves, G. L. A. Mota, G. A. O. P. Costa, and R. Q. Feitosa. Estimation of Transition Possibilities for Fuzzy Markov Chains Applied to the Analysis of Multitemporal Image Sequences. In *Proc. of the 4th Int. Conf. on Geographic Object-Based Image Analysis (GEOBIA)*, pages 367–371, 2012.

[11] F. Amigoni, N. Basilico, and N. Gatti. Finding the optimal strategies for robotic patrolling with adversaries in topologically-represented environments. In *Proceedings of the 26th Int. Conf. on Robotics and Automation (ICRA'09)*, pages 819–824, 2009.

[12] K. E. Avrachenkov and E. Sanchez. Fuzzy Markov Chains and Decision-Making. *Fuzzy Optimization and Decision Making*, 1:143–159, 2002.

[13] N. Basilico, N. Gatti, and F. Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184-185:78–123, 2012.

[14] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, and F. Amigoni. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *Proc. of the IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology*, volume 2, pages 557–564, 2009.

[15] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing: an international journal*, 8:239–287, 2009.

[16] J. Blanc and D. d. Hertog. On Markov Chains with Uncertain Data. Discussion Paper 2008-50, Tilburg University, Center for Economic Research, 2008.

[17] P. Bonissone and K. Decker. Selecting uncertainty calculi and granularity: An experiment in trading-off precision and complexity.

In *Proc. of the Conf. Annual Conf. on Uncertainty in Artificial Intelligence (UAI-85)*, pages 57 – 66, 1985.

[18] S. D. Bopardikar, F. Bullo, , and J. ao P. Hespanha. On discrete-time pursuit-evasion games with sensing limitations. *IEEE Transactions on Robotics*, 24(6):1429–1439, 2008.

[19] B. Bosanský, O. Vanek, and M. Pechoucek. Extending security games to defenders with constrained mobility. In *AAAI Spring Symposium: Game Theory for Security, Sustainability and Health*, volume SS-12-03 of *AAAI Technical Report*, pages 75–82, 2012.

[20] G. Brassard and P. Bratley. *Fundamentals of Algorithmics.* Prentice Hall, Englewood Cliffs, NJ, 1995.

[21] G. Brown, M. Carlyle, D. Diehl, J. Kline, and K. Wood. A Two-Sided Optimization for Theater Ballistic Missile Defense. *Operations Research*, 53(5):745–763, 2005.

[22] J. J. Buckley. Uncertain Probabilities III: the Continuous Case. *Soft Computing*, 8:200–206, 2004.

[23] J. J. Buckley. *Fuzzy Probabilities: New Approach and Applications, 2nd Edition*, volume 115 of *Studies in Fuzziness and Soft Computing.* Springer-Verlag, 2005.

[24] J. J. Buckley and E. Eslami. Uncertain Probabilities I: the Discrete Case. *Soft Computing*, 7:500–505, 2003.

[25] J. J. Buckley and E. Eslami. Uncertain Probabilities II: the Continuous Case. *Soft Computing*, 8:193–199, 2004.

[26] J. J. Buckley and E. Eslami. Fuzzy Markov Chains: Uncertain Probabilities. *Mathware & Soft Computing*, 9(1), 2008.

[27] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proc. of the IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology*, pages 302–308, 2004.

[28] W.-K. Ching and M. K. Ng. *Markov Chains: Models, Algorithms and Applications.* Springer-Verlag, New York, 2006.

[29] V. Conitzer. Computing Game-Theoretic Solutions and Applications to Security. In *Proc. of the 26th AAAI Conf. on Artificial Intelligence*, pages 2106–2112, 2012.

[30] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *Proc. of the 7th ACM Conf. on Electronic Commerce*, pages 82–90, 2006.

[31] W. Dent and R. Ballintine. A Review of the Estimation of Transition Probabilities in Markov Chains. *Australian Journal of Agricultural Economics*, 15(2):69–81, 1971.

[32] D. Dubois and H. Prade. Operations on fuzzy numbers. *Int. Journal of Systems Science*, 9(6):613–626, 1978.

[33] Y. Elmaliach, N. Agmon, and G. Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3):293–320, 2009.

[34] F. Fang, A. X. Jiang, and M. Tambe. Protecting Moving Targets with Multiple Mobile Resources. *Journal of Artificial Intelligence Research*, 48:583–634, 2013.

[35] R. Q. Feitosa, G. A. O. P. Costa, G. L. A. Mota, and B. Feijó. Modeling Alternatives for Fuzzy Markov Chain-Based Classification of Multitemporal Remote Sensing Data. *Pattern Recognition Letters*, (32):927 – 940, 2011.

[36] M. Gagolewski. *FuzzyNumbers Package: Tools to Deal with Fuzzy Numbers in R*, 2012.

[37] M. Gendreau and J.-Y. Potvin, editors. *Handbook of Metaheuristics.* Springer, 2010.

[38] R. C. Gentleman, J. F. Lawless, J. C. Lindsey, and P. Yan. Multi-State Markov Models for Analyzing Incomplete Disease History Data with

Illustrations for HIV Disease. *Statistics in Medicine*, 13(8):805–821, 1994.

[39] S. Gomez, A. Arenas, J. Borge-Holthoefer, S. Meloni, and Y. Moreno. Discrete-Time Markov Chain Approach to Contact-Based Disease Spreading in Complex Networks. *Euro Physics Letters (EPL)*, 89(38009):6, 2010.

[40] J. Halliwell and Q. Shen. Linguistic probabilities: Theory and application. *Soft Computing*, 13(2):169–183, 2008.

[41] R. Isaacs. *Differential Games: a Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization, year = 1965, publisher = John Wiley and Sons, address = New York.*

[42] R. Ivanek, Y. T. Grohn, A. J.-J. Ho, and M. Wiedmann. Markov Chain Approach to Analyze the Dynamics of Pathogen Fecal Shedding - Example of Listeria Monocytogenes Shedding in a Herd of Dairy Cattle. *Journal of Theoretical Biology*, 245(1):44–58, 2007.

[43] M. Jain, E. Karder, C. Kiekintveld, F. Ordoñez, and M. Tambe. Security games with arbitrary schedules: A branch-and-price approach. In *Proc. of the 24th AAAI Conf. on Artificial Intelligence*, pages 797–797, 2010.

[44] M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, and F. Ordónez. Software Assistants for Randomized Patrol Planning for the LAX Airport Police and the Federal Air Marshal Service. *Interfaces*, 40:267–290, 2010.

[45] C. I. Jones. On the Evolution of the World Income Distribution. *Journal of Economic Perspectives*, 11(3):19–36, 1997.

[46] B. H. Juang and L. R. Rabiner. Hidden Markov Models for Speech Recognition. *Technometrics*, 33(3):251–272, 1991.

[47] R. Kleyle and A. de Korvin. Transition Probabilities for Markov Chains Having Fuzzy States. *Stochastic Analysis and Applications*, 15(4):527–546, 1997.

[48] A. Kott and W. M. McEneany. *Adversarial Reasoning: Computational Approaches to Reading the Opponents Mind.* Chapman and Hall/ CRC Boca Raton, 2007.

[49] A. Kott and M. Ownby. Tools for real-time anticipation of enemy actions in tactical ground operations. In *Proceedings of the 10th Int. Command and Control Research and Technology Symposium*, 2005.

[50] R. Kruse, R. Buck-Emden, and R. Cordes. Processor Power Considerations - An Application of Fuzzy Markov Chains. *Fuzzy Sets and Systems*, 21(3):289–299, 1987.

[51] M. Kurano, M. Yasuda, J. Nakagami, and Y. Yoshida. A limit theorem in some dynamic fuzzy systems. *Fuzzy Sets and Systems*, 51(1):83 – 88, 1992.

[52] D. Li and J. B. Cruz. Game of defending a target: A linear quadratic differential game approach. In *Proc. of the 17th IFAC World Congress*, pages 2643–2648, 2008.

[53] K.-w. Lye and J. M. Wing. Game strategies in network security. *International Journal of Information Security*, 4(1-2):71–86, 2005.

[54] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *Proceedings ACM Symposium on Theory of Computing*, pages 322–333, 1988.

[55] S. Markovitch and R. Reger. Learning and exploiting relative weaknesses of opponent agents. *Autonomous Agents and Multi-Agent Systems*, 10:103–130, 2005.

[56] W. L. May and W. D. Johnson. Constructing two-sided simultaneous confidence intervals for multinomial proportions for small counts in a large number of cells. *Journal of Statistical Software*, 5(6):1–24, 2000.

[57] A. McLennan and R. Tourky. From Imitation Games to Kakutani, 2006. Unpublished.

[58] A. McLennan and R. Tourky. Imitation games and computation. *Games and Economic Behavior*, 70(1):4 – 11, 2010.

[59] A. McLennan and R. Tourky. Simple complexity from imitation games. *Games and Economic Behavior*, 68(2):683 – 688, 2010.

[60] J. Medhi. *Stochastic Models in Queueing Theory, 2nd Edition*. Academic Press, Boston, 2002.

[61] K. M. Mullen, D. Ardia, D. L. Gil, D. Windover, and J. Cline. Deoptim: An r package for global optimization by differential evolution. *Journal of Statistical Software*, 40(6):1–26, 2011.

[62] C. Negoita and D. Ralescu. *Applications of Fuzzy Sets to Systems Analysis*. John Wiley & Sons, 1975.

[63] P. K. Newton, J. Mason, K. Bethel, L. A. Bazhenova, J. Nieva, and P. Kuhn. A Stochastic Markov Chain Model to Describe Lung Cancer Growth and Metastasis. *PLoS ONE*, 7(4):e34637, 04 2012.

[64] M. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, MA, 1994.

[65] P. Paruchuri, J. P. Pearce, and S. Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'08)*, pages 895–902, 2008.

[66] D. Pelta and R. Yager. Dynamic vs. static decision strategies in adversarial reasoning. In *Proceedings of the Joint 2009 Int. Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conf. (IFSA-EUSFLAT'09)*, pages 472–477, 2009.

[67] D. Pelta and R. Yager. On the conflict between inducing confusion and attaining payoff in adversarial decision making. *Information Sciences*, 179:33–40, 2009.

[68] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed ARMOR Protection: The Application of a Game Theoretic Model for Security at the Los Angeles Int. Airport. In *Proc. of the 7th Int. Joint Conf. on Autonomous Agents and Multiagent Systems: Industrial Track*, pages 125–132, 2010.

[69] J. Pita, M. Tambe, C. Kiekintveld, S. Cullen, and E. Steigerwald. GUARDS: Game Theoretic Security Allocation on a National Scale. In *Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems - Volume 1*, pages 37–44, 2011.

[70] R. Popp and J. Yen. *Emergent Information Technologies and Enabling Policies for Counter-Terrorism.* John Wiley and Sons Hoboken, NJ, 2006.

[71] K. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization.* Springer - Natural Computing Series, 2005.

[72] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[73] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2015.

[74] J. E. Ricart. Una introducción a la teoría de los juegos. *Cuadernos Económicos*, 40, 1988.

[75] A. D. Sahin and Z. Sen. First-Order Markov Chain Approach to Wind Speed Modelling. *Journal of Wind Engineering and Industrial Aerodynamics*, 89(3-4):263–269, 2001.

[76] T. Sandler and D. G. A. M. Terrorism & Game Theory. *Simulation and Gaming*, 34(3):319–337, 2003.

[77] G. A. Satten and I. M. Longini. Markov Chains With Measurement Error: Estimating the 'True' Course of a Marker of the Progression of Human Immunodeficiency Virus Disease. *Journal of the Royal Statistical Society C*, 45(3):275–309, 1996.

[78] W. Schulze and B. van der Merwe. Music Generation with Markov Models. *IEEE Multimedia*, 18:78–85, 2011.

[79] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, , and G. Meyer. PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States. In *Proc. of the 11th Int. Conf. on Autonomous Agents and Multiagent Systems - Volume 1*, pages 13–20, 2012.

[80] C. P. Sison and J. Glaz. Simultaneous confidence intervals and sample size determination for multinomial proportions. *Journal of the American Statistical Association*, 90(429):366–369, 1995.

[81] D. C. Skinner. *Introduction to Decision Analysis, 3rd Ed.* Probabilistic Publishing, Gainesville, Florida, 2009.

[82] D. Sleator and R. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

[83] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(10):341–359, 1997.

[84] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned.* Cambdrige University Press, New York, 2011.

[85] D. Tran and M. Wagner. Fuzzy Hidden Markov Models for Speech and Speaker Recognition. In *Proc. of 18th Int. Conf. of the North American Fuzzy Information Processing Society (NAFIPS)*, pages 426–430, 1999.

[86] I. Triguero, S. Garcia, and F. Herrera. Differential evolution for optimizing the positioning of prototypes in nearest neighbor classification. *Pattern Recognition*, 44(4):901–916, 2011.

[87] J. van der Hoek and R. J. Elliott. American Option Prices in a Markov Chain Market Model. *Applied Stochastic Models in Business and Industry*, 28(1):35–59, 2012.

[88] O. Vanek, B. Bosanský, M. Jakob, V. Lisý, and M. Pechoucek. Extending security games to defenders with constrained mobility. In *AAAI Spring Symposium: Game Theory for Security, Sustainability and Health*, volume SS-12-03 of *AAAI Technical Report*, pages 75–82, 2012.

[89] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry. Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, 2002.

[90] P. Villacorta and D. Pelta. Evolutionary design and statistical assessment of strategies in an adversarial domain. In *Proceedings of the IEEE Conf. on Evolutionary Computation (CEC'10)*, pages 2250–2256, 2010.

[91] P. Villacorta and D. Pelta. Expected payoff analysis of dynamic mixed strategies in an adversarial domain. In *Proceedings of the 2011 IEEE Symposium Series on Computational Intelligence (SSCI'11), Intelligent Agents Conf.*, 2011. In press.

[92] P. Villacorta and D. Pelta. Theoretical analysis of expected payoff in an adversarial domain. *Information Sciences*, 186(1):93–104, 2012.

[93] P. J. Villacorta. *MultinomialCI: Simultaneous Confidence Intervals for Multinomial Proportions According to the Method by Sison and Glaz*, 2012. R package available in http://cran.r-project.org/package=MultinomialCI.

[94] P. J. Villacorta and D. A. Pelta. A repeated imitation model with dependence between stages: decision strategies and rewards. *Int. Journal of Applied Mathematics and Computer Science*, 25(3):617 – 630, 2015.

[95] P. J. Villacorta, D. A. Pelta, and M. T. Lamata. Forgetting as a way to avoid deception in a repeated imitation game. *Autonomous Agents and Multi-Agent Systems*, 27(3):329 – 354, 2013.

[96] P. J. Villacorta and J. L. Verdegay. FuzzyStatProb: an R package for the estimation of fuzzy stationary probabilities from a sequence of observations of an unknown Markov chain. *Journal of Statistical Software*, 2015. In press.

[97] P. J. Villacorta, J. L. Verdegay, and D. A. Pelta. Towards Fuzzy Linguistic Markov Chains. In *Proc. of the 8th Conf. of the European Society for Fuzzy Logic and Technology (EUSFLAT)*, pages 707–713, 2013.

[98] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. John Wiley and Sons, New York, 1944.

[99] H. von Stackelberg. *Marktform und Gleichgewicht*. Springer, Vienna, 1934.

[100] N. J. Welton and A. E. Ades. Estimation of Markov Chain Transition Probabilities and Rates from Fully and Partially Observed Data: Uncertainty Propagation, Evidence Synthesis, and Model Calibration. *Medical Decision Making*, 25(6):633–645, 2005.

[101] R. Yang, B. Ford, M. Tambe, and A. Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *Proc. of the 13th Int. Conf. on Autonomous Agents and Multi-agent Systems*, pages 453–460, 2014.

[102] Z. Yin, A. X. Jiang, M. Tambe, C. Kiekintveld, K. Leyton-Brown, T. Sandholm, and J. P. Sullivan. TRUSTS: Scheduling Randomized Patrols for Fare Inspection in Transit Systems Using Game Theory. *AI Magazine*, 33(4), 2012.

[103] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.

[104] L. A. Zadeh. The Concept of a Linguistic Variable and its Application to Approximate Reasoning - I. *Information Sciences*, 8(3):199 – 249, 1975.

[105] L. A. Zadeh. The Concept of a Linguistic Variable and its Application to Approximate Reasoning - III. *Information Sciences*, 9(1):43–80, 1975.

[106] L. A. Zadeh. Maximizing Sets and Fuzzy Markoff Algorithms. *IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, (28):9 – 15, 1998.

[107] I. Zuckerman, S. Kraus, and J. Rosenschein. The adversarial activity model for bounded rational agents. *Autonomous Agents and Multi-Agent Systems*, pages 1–36, 2010.