Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria

DOTTORATO DI RICERCA IN INGEGNERIA
DELL'INFORMAZIONE

Algorithms for the verification, computation and learning of equilibria in extensive-form games

Doctoral Dissertation of:
Fabio Panozzo

Advisors:
Prof. Nicola Gatti
Tutor:
Prof. Donatella Sciuto

Supervisor of the Doctoral Program:
Prof. Carlo E. Fiorini

2013 - XXVI

Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Piazza Leonardo da Vinci 32, I-20133 - Milano

Abstract

The study of strategic-interaction situations has recently received increasing attention in artificial intelligence with the aim of designing autonomous software agents able to act optimally. These situations are customarily modeled as *games* [19] in which the *mechanism* describes the rules and *strategies* describe the behavior of the agents. Particular attention is focused on the study of formal methods to theoretically guarantee the optimality of the agents' behavior. Game theory and microeconomics provide mathematical tools to model strategic-interaction situations and characterize the appropriate solution concepts. However, they do not provide computational tools to find solutions. This problem, commonly called *equilibrium computation*, is instead central in computer science, whose aim includes assessing the complexity of finding an exact or approximate solution, designing exact or approximate algorithms, and evaluating the application of the algorithms in practical settings [59].

In this thesis, we focus on non-cooperative general-sum extensive-form games. Extensive-form games [19] are more suitable to represent real-world situations providing a richer representation than strategic-form games, where the sequential structure of decision-making is described explicitly and each agent is allowed to be free to change her mind as events unfold. While in zero-sum extensive-form games the main problem is to solve games of large dimension (e.g., poker [29]), in general-sum extensive-form games there are other important issues. The most important is the choice of the most suitable solution concept: it depends on the scenario modelled by the game, e.g. the agents' common knowledge, and it depends on its computability, i.e. the amount of time and memory necessary to solve the game. We can identify two main scenarios, characterized through the knowledge available to agents: the case where the agents have common information and the case where they have incomplete information.

Interestingly, with common information, the appropriate solution concepts for extensive-form games refine the concept of Nash equilibrium, while, when agents learn, the appropriate solution concepts relax the concept of Nash equilibrium. More precisely, in the first scenario, it is well known that the Nash equilibrium is not suitable, allowing strategies to be non-sequentially rational. The game theoretic literature provides a

number of refinements of Nash equilibrium for extensive-form games that are the appropriate solution concepts. For these solution concepts the verification and computation problems can be hard. In this thesis we show that for some solution concepts the verification problem is easy and we design algorithms to find some refinements of Nash equilibrium. When the information is not common every agent needs to have a set of beliefs over the opponents' behaviour. This beliefs are updating during the learning process, i.e. while the agents repeat the game. In this thesis we discuss the suitable solution concepts for this scenario and we develop efficient evolutionary game theory techniques and algorithms that works with the sequence form representation of extensive-form games allowing an exponential reduction of time and space w.r.t. the algorithms presented in literature.

Finally, we experimental evaluate each presented algorithm.

Sommario

Lo studio di situazione di interazione strategica ha recentemente ricevuto un'attenzione crescente nel campo dell'intelligenza artificiale al fine di progettare agenti autonomi capaci di agire ottimamente. Queste situazioni sono comunemente modellate come *giochi* [19] in cui il *meccanismo* descrive le regole e le *strategie* descrivono il comportamento degli agenti. Particolare attenzione è focalizzata sullo studio di metodi formali per garantire teoricamente l'ottimalità del comportamento degli agenti. La teoria dei giochi e la microeconomia forniscono gli strumenti matematici per modellare la situazioni di interazione strategica e caratterizzare gli appropriati concetti di soluzione. Comunque, essi non forniscono gli strumenti computazionali per cercare le soluzioni. Questo problema, comunemente chiamato *calcolo degli equilibri*, è invece centrale in informatica, il cui obiettivo include la valutazione della complessità di trovare una soluzione esatta o approssimata, progettare algoritmi esatti o approssimati, e valutare l'applicabilità degli algoritmi in situazioni reali [59].

In questa tesi, ci focalizziamo sui giochi non cooperativi in forma estesa a somma generale. I giochi in forma estesa [19] sono più adatti a rappresentare le situazioni del mondo reale fornendo una rappresentazione più ricca dei giochi in forma strategica, dove la struttura sequenziale delle decisioni è descritta esplicitamente e ogni agente ha la possibilità di cambiare strategia dopo le azioni dell'avversario. Mentre i giochi in forma estesa a somma zero il problema principale è la risoluzione di giochi di grande dimensione (per esempio il poker [29]), nei giochi in forma estesa a somma generale ci sono altri importanti problemi. Il più importante è la scelta del più adatto concetto di soluzione: esso dipende dallo scenario modellato dal gioco, per esempio la conoscenza comune degli agenti, e dipende dalla sua commutabilità, cioè la quantità di tempo e di memoria necessari a risolvere il gioco. Possiamo identificare due scenari principali, caratterizzare dalla conoscenza disponibile agli agenti: il caso in cui gli agenti hanno informazione comune e il caso in cui essi hanno informazione incompleta.

È interessante come, con informazione comune, i concetti di soluzione appropriati per i giochi in forma estesa sono i raffinamenti dell'equilibrio di Nash, mente, quando gli agenti devono apprendere, i concetti di soluzione appropriati rilassano il concetto di equilibrio di Nash. Più precisamente, nel primo

scenario, è noto che l'equilibrio di Nash non è adatto, permettendo agli agenti di giocare strategie che non sono sequenzialmente razionali. La letteratura di teoria dei giochi fornisce un numero di raffinamenti dell'equilibrio di Nash per giochi in forma estesa che sono i concetti di soluzione appropriati. Per questi concetti di soluzione i problemi di verifica e computazione possono essere difficili. In questa tesi mostriamo che per alcuni concetti di soluzione il problema della verifica è facile e progettiamo algoritmi in grado di trovare alcuni raffinamenti dell'equilibrio di Nash. Quando l'informazione non è comune ogni agente ha bisogno di un insieme di credenze sul comportamento degli avversari. Queste credenze sono aggiornate durante il processo di apprendimento, cioè mentre gli agenti ripetono il gioco. In questa tesi discutiamo i possibili concetti di soluzione per questo scenario e sviluppiamo tecniche di teoria dei giochi evoluzionaria efficienti e algoritmi che si applicano direttamente alla rappresentazione in forma sequenza del gioco in forma estesa permettendo una riduzione esponenziale del tempo e dello spazio rispetto agli algoritmi presenti in letteratura.

Infine valutiamo sperimentalmente gli algoritmi presentati in questa tesi.

Contents

Introduction 1

The study of strategic-interaction situations has recently received increasing attention in artificial intelligence with the aim of designing autonomous software agents able to act optimally. These situations are customarily modeled as *games* [19] in which the *mechanism* describes the rules and *strategies* describe the behavior of the agents. Particular attention is focused on the study of formal methods to theoretically guarantee the optimality of the agents' behavior. *Game theory* and *microeconomics* provide the most elegant formal methods for strategic-interaction scenarios [19]. Well-known successful applications based on these methods are, e.g., in physical security [5, 40], poker [28] and billiard games [1], economic transactions [35, 52]. Game theory and microeconomics provide mathematical tools to model strategic-interaction situations and characterize the appropriate solution concepts. However, they do not provide computational tools to find solutions. This problem, commonly called *equilibrium computation*, is instead central in computer science, whose aim includes assessing the complexity of finding an exact or approximate solution, designing exact or approximate algorithms, and evaluating the application of the algorithms in practical settings [59].

In this thesis, we work with non-cooperative extensive-form games. Extensive-form games [19] provide a richer representation than strategic-form games, where the sequential structure of decision-

making is described explicitly and each agent is allowed to be free to change her mind as events unfold. The study of extensive-form games is customarily carried out in game theory by translating the games by means of tabular representations [59]. The most common is the *normal form*. Its advantage is that all the techniques applicable to strategic-form games can be adopted also with this representation. However, the size of the normal-form representation grows exponentially with the size of the game tree, thus being impractical. To circumvent these issues, *sequence form* was proposed [69]. The sequence form provides a number of advantages: its size is linear in the size of the game tree and it is more expressive than the normal form. On the other hand, standard algorithmic techniques for strategic-form games cannot be adopted with the sequence form requiring alternative *ad hoc* techniques.

The central solution concept in non-cooperative game theory is the *Nash equilibrium* (NE). It constrains the strategy of each agent to be optimal given the strategies of the opponents. It is well known that NE is not suitable for extensive-form games, allowing strategies to be non-sequentially rational. The game theoretic literature provides a number of refinements of Nash equilibrium for extensive-form games that are the appropriate solution concepts in the scenarios where the assumption of NE (e.g., *common* and *complete* information over payoffs) are verified. The most common solution concepts are [66]: *subgame perfect equilibrium* (SPE) [55] when information is perfect and *sequential equilibrium* (SE) [41], *quasi-perfect equilibrium* (QPE) [65], and *extensive-form perfect equilibrium* (EFPE) [56] when information is imperfect. Each solution concept is appropriate for a specific situation. For instance, SE is commonly adopted in dynamic bargaining [22] and signaling games [19], while perfect equilibrium concepts are particularly suitable for learning agents. Indeed, during learning dynamics, agents explore non-optimal strategies with a small probability [63] and this exploration behaves as a tremble in the agents' strategies.

On the other hand, there are many real-world situations where the epistemic requirements of NE are not satisfied. In these scenarios the NE is not the appropriate solution concept and also its refinements that require the same epistemic assumptions are not suitable. Even when NE is used as *descriptive* tool to study what are stable states of learning agents, some problems arise, learning agents having non-NE stable states, and therefore NE may have a non-

satisfactory descriptive power [18]. The aforementioned drawbacks of NE pushed researchers to design alternative (relaxed and/or non-equilibrium) solution concepts taking also into account learning dynamics, e.g., the closed under rational behavior (CURB) set [7] is a non-equilibrium concept defined as a set of strategies that contains the best responses to any mixture over itself and any best-response dynamics will stay within it [34]. With extensive-form games, relaxing the epistemic requirements of NE, it is possible to have stable (equilibrium) states that are not NEs [18]. These equilibria, called *self-confirming equilibria* (SCEs), require that each agent plays best-response strategies to her beliefs, but the beliefs can be *incorrect* off the equilibrium path (while they are *confirmed on*). This concept is perfectly suitable for learning agents: if agents can entirely explore the strategy profile space, they would have correct beliefs everywhere on the game tree and they would play an SPE or an SE, but, in practice, learning agents cannot explore the whole space and therefore they can have incorrect beliefs over some portion of the tree, playing thus an SCE.

1.1 The aim of this work

In this thesis we aim to address the problem of designing efficient algorithms for the verification, computation, and learning of equilibria in extensive-form games. Specifically, we study the following problems:

- verifying if a given assessment is an SE,
- given a perturbation over the behavioral strategies of the agents, computing a strategy profile that is an EFPE,
- designing an algorithm to find a Markov perfect equilibrium (MPE) in two-player zero-sum continuous stochastic games with switching control,
- designing an apposite version of replicator dynamics that deals with the sequence form,
- defining a variation of Q-learning algorithm that is directly applicable to the sequence form,

- defining and computing the appropriate solution concepts in game with populations.

1.2 Research contributions

In this section we separately describe our research contribution for the problems highlighted above.

Verification and computation of Nash equilibrium refinements

- When the assessment is given as input, we design an efficient algorithm certifying that the assessment is an SE with an arbitrary number of agents.

- When only the behavioral strategy profile is given as input, we design an efficient algorithm certifying that the strategy profile is an SE with two-agent games; this algorithm can be employed also to derive the agents' consistent (in the sense of Kreps and Wilson) beliefs from the strategies and a simple variation can be employed for the verification of QPE with two-agent games.

These results are presented in [21]:

N. Gatti and F. Panozzo. New results on the verification of Nash refinements for extensive-form games. In AAMAS, pages 813-820, Valencia, Spain, June 4-8 2012.

N. Gatti and F. Panozzo. Efficient algorithms for the verification problem of sequential equilibrium. Submitted to Journal of Artificial Intelligence Research.

- Given a perturbation over the behavioral strategies of the agents, an EFPE with two-agent general-sum games can be computed exactly by means of a variation of the Lemke's algorithm which the perturbation is over the coefficients of the variables and therefore the problem is in $\mathcal{PPAD}$.

- We provide an iterative semidefinite programming (SDP) based algorithm that converges to an MPE when both the reward and the state transitions are polynomial functions and returns an ϵ-approximate MPE (ϵ-MPE) with $\epsilon \leq \bar{\epsilon}$, where $\bar{\epsilon}$ is given as input.

- We use our algorithm to approximate solutions of non-polynomial games: approximating the reward and state transitions given as input with polynomials, solving the approximated game, and providing theoretical upper bounds on the quality of the solutions as functions of the approximation error with infinity norm.

- We experimentally evaluate the performance of our algorithm in terms of iterations and compute time.

These results are presented in [9]:

G. Bonomi, N. Gatti, F. Panozzo, and M. Restelli. Computing equilibria with two-player zero-sum continuous stochastic games with switching control. In AAAI, Toronto, Canada, July 22-26 2012.

Learning in extensive-form games

- We show that the standard replicator dynamics for normal form cannot be adopted with the sequence form, the strategies produced by replication not being well-defined sequence-form strategies.

- We design an *ad hoc* version of the discrete-time replicator dynamics for sequence form and we show that it is sound, the strategies produced by replication being well-defined sequence-form strategies.

- We show that our replicator dynamics is realization equivalent to the standard discrete-time replicator dynamics for normal form and therefore that the two replicator dynamics evolve in the same way.

- We extend our discrete-time replicator dynamics to the continuous-time case, showing that the same properties are satisfied and extending standard tools to study the stability of the strategies to our replicator.

These results are presented in [25]:

N. Gatti, F. Panozzo, and M. Restelli. Efficient evolutionary dynamics with extensive-form games. In AAAI, Bellevue, Washington, USA, July 14-18 2013.

- We define a variation of the standard Q-learning algorithm that is applicable to the sequence form of an extensive-form game, showing that, in addition to an exponential reduction of the learning times, it allows to learn strategy profiles that cannot be learned using the normal form representation.
- We show that, in expectation, the time-limit learning dynamics of our algorithm can be described by means of a new sequence-form replicator dynamics with a mutation term and that, when the mutation term tends to zero, the replicator dynamics converge to a Nash equilibrium with perfect-information games.
- We experimentally evaluate our Q-learning based algorithm applied to the sequence form, comparing its dynamics w.r.t. the dynamics obtained when the standard Q-learning algorithm is applied to the normal form and evaluating the accuracy of our replicator dynamics model as the learning parameters vary.

N. Gatti, F. Panozzo, and M. Restelli. Q-learning based algorithm for the sequence form of extensive-form games. Submitted to AAAI, Quebec city, Quebec, July 27-31 2014.

- We design some solution concepts extending SCE to capture situations where agents may be of different (Bayesian) finite *types* and *individuals*.

- We study the relationships between different SCE extensions as the number of individuals and types change.
- With two agents, we formulate the problems of finding a solution with finite and infinite populations as two mixed-integer linear mathematical programs (MILP). Furthermore, we show that the problem of verifying whether or not a solution is an SCE extension is in $\mathcal{P}$ even when the input to the verification problem specifies only the strategies and not the beliefs (this is common when we observe learning agents and we ask whether their strategies are a solution for some given solution concept), while searching for an SCE is $\mathcal{PPAD}$-complete. Furthermore, we discuss to enumerate the equilibria.
- We study the replicator dynamics with multiple individuals and we show how SCEs affect learning dynamics in games with perfect information: they are reachable states only when learning agents are purely greedy, otherwise they are saddle points attracting and then repelling the dynamics. New SCEs emerge as the individuals increase, changing the learning trajectories and delaying their convergence.

These results are presented in [23, 24, 26]:

Mathematical programming formulations to compute steady states in two-player extensive-form games. In Interactive Decision Theory and Game Theory Workshop of AAAI, Atlanta, USA, July 11 2010.

N. Gatti, F. Panozzo, and S. Ceppi. Computing a self-confirming equilibrium in two-player extensive-form games. pages 981-988, Taipei, Taiwan, 2010.

N. Gatti, F. Panozzo, and M. Restelli. Extensive-form games with heterogeneous populations: solution concepts, equilibria characterization, and learning dynamics. In AAMAS, Saint Paul, USA, May 6-10 2013.

1.3 Thesis structure

The thesis is structured as follows. Chapter 2 is dedicated to game-theoretic preliminaries. Specifically, we introduce the concept of

game and its representations, we list some important solution concepts that we use in this thesis and the relationship between them, we describe the computational complexity classes of equilibrium computational problems treated in this work and we close the chapter with the basic concepts of evolutionary game theory and reinforcement learning.

Part I is focus on the verification and computation of Nash equilibrium refinements. Chapter 4 presents the designing of an efficient algorithms for the verification problem of sequential equilibrium. In Chapter 5 we focus on the problem of design an efficient algorithm that, given a specific perturbation over the sequence-form strategies of a two-player extensive-form game, it computes an EFPE. In Chapter 6 we provide an iterative SDP based algorithm that converges to an MPE when both the reward and the state transitions are polynomial functions.

In Part II we address the problem of learning agent in extensive-form games. In Chapter 7 we explore the adoption of evolutionary game theory tools with sequence form for the study of extensive-form games designing an *ad hoc* version of the replicator dynamics for sequence form and we show that the strategies produced by replication being well-defined sequence-form strategies. In Chapter 8 we develop the first Q-learning based algorithm working with the sequence form of an extensive-form game, thus allowing an exponential reduction of the length of the learning dynamics w.r.t. normal form. Chapter 9 is dedicated to the problem of learning in extensive-form games with heterogeneous populations

Chapter 10 concludes the thesis and outlines some directions of future research.

Preliminaries 2

In this chapter we give the game-theoretic preliminaries necessary for the development of the thesis.

In Section 2.1 we introduce the formal definition of game and its representation in strategic form and extensive form. In Section 2.2 we introduce the main operational representations of an extensive-form game. In Section 2.2 we show the relationship between the different representations introduced. In Section 2.3 we give the concept of perturbed strategies. The Section 2.4 is dedicated to games with population. In Section 2.5 we list the main solution concepts and the relationship between them. In Section 2.6 we describe the most important computational complexity classes of problems that always admit a solution. In Section 2.7 we show a further representation of game, the so called stochastic games, and its properties. In Section 2.8 we give the basic concepts of evolutionary game theory. Finally, in Section 2.9 we present the Q-learning algorithm and its relationship with the evolutionary game theory.

2.1 Game definition

Game theory is the study of strategic decision making. More formally, it is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers". Mathematical

models were introduced to describe strategic-interaction situations; these model are called *games*.

There are different representations of game. The first one that we introduce are the *strategic-form games* that describe situations where all the agents play simultaneously.

Definition 2.1 (Strategic-form game) *A (finite, n-agent) strategic-form game is a tuple* $(N, A, \mathbf{u})$*, where*

- N *is a finite set of agents,*
- $A = A_1 \times \cdots \times A_n$*, where* A_i *is a finite set of actions available to agent* i*. Each vector* $\mathbf{a} = (a_1, \ldots, a_n) \in A$ *is called an action profile,*
- $\mathbf{u} = (u_1, \ldots, u_n)$*, where* $u_i : A \mapsto \mathbb{R}$ *is a real-valued utility (or payoff) function for agent* i*.*

A two-player strategic-form game in literature known as "Prisoner's dilemma" is represented in Tab 2.1 .

	C	D
c	3 , 3	0 , 5
d	5 , 0	1 , 1

Table 2.1: Example of two-agent strategic-form game; c and d are actions available to agent 1, while C and D are actions available to agent 2.

The *extensive form* provides a richer representation than the strategic form. It provides the sequential structure of decision-making being described explicitly and each agent being allowed to be free to change her mind as events unfold.

Definition 2.2 (Perfect-information extensive-form game) *A perfect-information extensive-form game [19] is formally defined as a tuple* $(N, A, V, T, \iota, \rho, \chi, \mathbf{u})$*, where*

- N *is the set of agents (we denote by* $i \in N$ *a generic agent),*
- A *is the set of actions (we denote by* $A_i \subseteq A$ *the set of actions of agent* i *and by* $a \in A$ *a generic action),*

- *V is the set of decision nodes (we denote by $V_i \subseteq V$ the set of decision nodes of agent i),*
- *T is the set of terminal nodes (we denote by $w \in V \cup T$ a generic node and by w_0 the root node),*
- *$\iota : V \to N$ is the function that specifies the agent that acts at a given decision node,*
- *$\rho : V \to \wp(A)$ returns the actions available to agent $\iota(w)$ at a decision node w,*
- *$\chi : V \times A \to V \cup T$ assigns the next (decision or terminal) node to each pair composed of a decision node w and an action a available at w, and*
- *$\mathbf{u} = (u_1, \ldots, u_n)$ is the vector of agents' utility functions where $u_i : T \to \mathbb{R}$.*

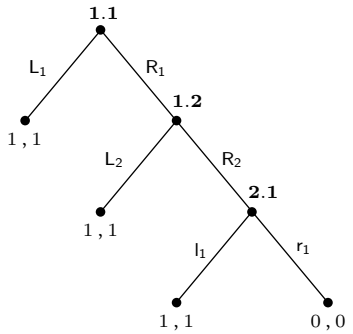


Figure 2.1: Example of two-agent perfect-information extensive-form game (we denote by $i.j$ the j-th information set of agent i, while $\mathsf{L}_1, \mathsf{R}_1, \mathsf{L}_2, \mathsf{R}_2$ are actions available to agent 1 and $\mathsf{l}_1, \mathsf{r}_1$ are actions available to agent 2). This game may seem equal to a game with three actions available to node 1.1, L_1, R_1L_2 and R_1R_2. We show in the following that this equivalence does not hold.

Perfect information captures exclusively situations in which all the agents can observe all the actions undertaken by the opponents. *Imperfect information* relaxes this constraint, capturing situations in

which some agent does not observe some opponents' action[1]. Imperfect-information games are based on the concept of *information set*. An information set is a collection of decision nodes with the property that, when an agent plays at one of them, she cannot distinguish the specific node in which she is playing. We denote by $V_{i,h}$ the information set of agent i whose index is h. Under the assumption that all the information sets in the game tree have a different index h, we can uniquely identify an information set by h.

Definition 2.3 (Imperfect-information extensive-form game) *An imperfect-information extensive-form game is a tuple* $(N, A, V, T, \iota, \rho, \chi, \mathbf{u}, H)$ *where*

- $(N, A, V, T, \iota, \rho, \chi, \mathbf{u})$ *is a perfect-information extensive-form game and*
- $H = (H_1, \ldots, H_n)$ *induces a partition* $V_i = \bigcup_{h \in H_i} V_{i,h}$ *such that for all* $w, w' \in V_{i,h}$ *we have* $\rho(w) = \rho(w')$.

With abuse of notation we use throughout the thesis $\rho(h)$ *in place of* $\rho(w)$ *where* $w \in V_{i,h}$.

We focus on games with *perfect recall* [19] where each agent recalls all her own previous actions and all her previous observations. Perfect recall places non-trivial constraints over the structure of every $V_{i,h}$. We omit their description here, not being necessary for our work, and we point the interested reader to [19]. We report an example of extensive-form game in Fig. 2.1.

2.2 Game and strategy representations

There are three main operational representations for an extensive-form game: *normal form* [68], *agent form* [42, 56], and *sequence form* [69], each based on a specific representation of the space of the agents' strategies.

[1] Games with uncertain information can be described as games with imperfect information with a special player, the *nature*, that selects nodes according to a given distribution probability.

Agent form

In the agent form, each agent is replicated in a number of fictitious agents and each of them is assigned to a different information set. In this way, each fictitious agent plays only at one single information set. A strategy in the agent form is commonly said *behavioral strategy* and we denote a behavioral strategy profile by $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \ldots, \boldsymbol{\sigma}_{|N|})$ where $\boldsymbol{\sigma}_i$ is the strategy of agent i and we denote by $\sigma_i(a)$ the probability associated with action $a \in A_i$. A behavioral strategy $\boldsymbol{\sigma}_i$ assigns independently each information set $h \in H_i$ a probability distribution over the actions $\rho(h)$ and therefore it is such that $\sum_{a \in \rho(h)} \sigma_i(a) = 1$. The agent form is linear in the size of the game tree.

Example 2.1 *Consider the game depicted in Fig. 2.1, the agent form is:*

		agent 2.1	
		l_1	r_1
agent 1.1	L_1	$1, 1$	$1, 1$
	R_1	$1, 1$	$1, 1$

L_2

		agent 2.1	
		l_1	r_1
agent 1.1	L_1	$1, 1$	$1, 1$
	R_1	$1, 1$	$0, 0$

R_2

agent 1.2

where we assign each fictitious agent the label of the information set in which the fictitious agent plays. An example of behavioral strategies is:

$$\boldsymbol{\sigma}_1 = \begin{cases} \sigma_1(\mathsf{L}_1) & = 1.0 \\ \sigma_1(\mathsf{R}_1) & = 0.0 \\ \sigma_1(\mathsf{L}_2) & = 0.0 \\ \sigma_1(\mathsf{R}_2) & = 1.0 \end{cases} \quad \boldsymbol{\sigma}_2 = \begin{cases} \sigma_2(\mathsf{l}_1) & = 0.8 \\ \sigma_2(\mathsf{r}_1) & = 0.2 \end{cases}$$

Normal form

In the normal form, a strategy profile is represented as a probability distribution over *plans*. A plan of agent i, denoted by $p \in P_i$ where $P_i \subseteq P$ is the set of plans of agent i and P is the set of all the plans, is a tuple of game tree actions specifying one action $a \in A_i$ per information set $h \in H_i$. We denote a normal-form strategy profile by $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{|N|})$ where $\boldsymbol{\pi}_i$ is the strategy of agent i and we denote by $\pi_i(p)$ the probability associated with plan $p \in P_i$. A normal-form

strategy is such that $\sum_{p \in P_i} \pi_i(p) = 1$. The agent i's utility is represented as a multi-dimensional array, denoted by U_i, specifying the value associated to every combination of plans of all the agents. The number of plans in an extensive-form game (and therefore the size of the normal form) is exponential in the size of the game tree.

Example 2.2 *Consider the game depicted in Fig. 2.1. A plan of agent* 1 *is a pair specifying one action at information set* $h = \mathbf{1.1}$ *and one action at information set* $h = \mathbf{1.2}$*, e.g.,* $\mathsf{R}_1\mathsf{L}_2$*. The normal-form utility bimatrix and an example of strategies are:*

		agent 2	
		l_1	r_1
agent 1	$\mathsf{L}_1\mathsf{L}_2$	$1, 1$	$1, 1$
	$\mathsf{L}_1\mathsf{R}_2$	$1, 1$	$1, 1$
	$\mathsf{R}_1\mathsf{L}_2$	$1, 1$	$1, 1$
	$\mathsf{R}_1\mathsf{R}_2$	$1, 1$	$0, 0$

$$\boldsymbol{\pi}_1 = \begin{cases} \pi_1(\mathsf{L}_1\mathsf{L}_2) & = 0.5 \\ \pi_1(\mathsf{L}_1\mathsf{R}_2) & = 0.0 \\ \pi_1(\mathsf{R}_1\mathsf{L}_2) & = 0.0 \\ \pi_1(\mathsf{R}_1\mathsf{R}_2) & = 0.5 \end{cases} \qquad \boldsymbol{\pi}_2 = \begin{cases} \pi_2(\mathsf{l}_1) & = 0.8 \\ \pi_2(\mathsf{r}_1) & = 0.2 \end{cases}$$

There exists a reduced version of normal form, so called *reduced normal form*, that is obtained from the previous one by deleting replicated strategies [67]. Although reduced normal form can be much smaller than normal form, it is exponential in the size of the game tree.

Example 2.3 *Consider the game depicted in Fig. 2.1. The reduced-normal-form utility bimatrix and an example of strategies are:*

		agent 2	
		l_1	r_1
agent 1	L_1*	$1, 1$	$1, 1$
	$\mathsf{R}_1\mathsf{L}_2$	$1, 1$	$1, 1$
	$\mathsf{R}_1\mathsf{R}_2$	$1, 1$	$0, 0$

$$\boldsymbol{\pi}_1 = \begin{cases} \pi_1(\mathsf{L}_1*) & = 0.5 \\ \pi_1(\mathsf{R}_1\mathsf{R}_2) & = 0.0 \\ \pi_1(\mathsf{R}_1\mathsf{R}_2) & = 0.5 \end{cases} \qquad \boldsymbol{\pi}_2 = \begin{cases} \pi_2(\mathsf{l}_1) & = 1.0 \\ \pi_2(\mathsf{r}_1) & = 0.0 \end{cases}$$

Sequence form

In the sequence form, a strategy is represented as a probability distribution over *sequences*. A sequence $q \in Q_i$ of agent i is a set of consecutive actions $a \in A_i$ where $Q_i \subseteq Q$ is the set of sequences of agent i and Q is the set of all the sequences. A sequence can be *terminal* (e.g., $q = \mathsf{L}_1$ in Fig. 2.1), if, combined with some sequence of the opponents, it leads to a terminal node, or *non-terminal* (e.g., $q = \mathsf{R}_1$ in Fig. 2.1), if it cannot lead to any terminal node for each opponents' sequence. Each agent has a fictitious initial sequence, denoted by $\mathsf{q}_\varnothing$ and called *empty sequence*. Furthermore, given a sequence $q \in Q_i$ leading to some information set $h \in H_i$, we say that sequence q' *extends* q (and we denote it by $q' = q|a$) if the last action of q' is some action $a \in \rho(h)$. We denote a sequence-form strategy profile by $\mathbf{x} = [\mathbf{x}_1, \dots, \mathbf{x}_{|N|}]$ where $\mathbf{x}_i$ is the strategy of agent i and we denote by $x_i(q)$ the probability associated with sequence $q \in Q_i$. Well-defined strategies are such that, for every information set $h \in H_i$, the probability $x_i(q)$ assigned to the sequence q leading to h is equal to the sum of the probabilities $x_i(q')$s where q' extends q at h, formally, $x_i(q) = \sum_{a \in \rho(h) : q|a \in Q} x_i(q|a)$ for all $h \in H_i$. Sequence form constraints can be conveniently described as $F_i \cdot \mathbf{x}_i = \mathbf{f}_i$, where F_i is an opportune matrix and $\mathbf{f}_i$ is an opportune vector, both with entries in $\{-1, 0, 1\}$. These matrixes guarantee that agent i's strategy $\mathbf{x}_i$ is well defined. We denote by $\mathbf{v}_i$ the vector of $v_{i,h}$, where each $v_{i,h}$ expresses the value agent i expects from playing at information set h where $\iota(h) = i$. The agent i's utility is represented as a sparse multi-dimensional array, denoted by U_i, specifying the value associated to every combination of agents' terminal sequences leading to an outcome. The size of the sequence-form representation is linear in the size of the game tree. Exactly, excluded the empty sequence, we have one sequence $q \in Q_i$ per action $a \in A_i$. We provide an example of sequence-form representation.

Example 2.4 *Consider the game depicted in Fig. 2.1, the sequence-form*

utility bimatrix and an example of strategies are:

		agent 2		
		$q_\varnothing$	l_1	r_1
agent 1	$q_\varnothing$			
	L_1	$1,1$		
	R_1			
	R_1L_2	$1,1$		
	R_1R_2		$1,1$	$0,0$

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

the constraints $F_1 \cdot \mathbf{x}_1 = \mathbf{f}_1$ *and* $F_2 \cdot \mathbf{x}_2 = \mathbf{f}_2$ *are:*

$$F_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 \end{bmatrix}, \mathbf{f}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$F_2 = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 1 \end{bmatrix}, \mathbf{f}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Since in this thesis we frequently exploit the correspondence between actions $a \in A$ and sequences $q \in Q$, we denote by $a(q)$ the last action of sequence q and by $q(a)$ the sequence whose last action is a. Given a sequence-form strategy, a behavioral strategy can be easily derived as:

$$\sigma_i(a) = \begin{cases} \dfrac{x_i(q|a)}{x_i(q)} & \text{if } x_i(q) \text{ is strictly positive} \\ \text{any feasible probability value} & \text{otherwise} \end{cases}$$

Relationships between the representations

We briefly describe the relationships among the above three representations. First, a sequence-form strategy $\mathbf{x}_i$ is equivalent to a number (precisely, a compact set) of normal-form strategies $\boldsymbol{\pi}_i$ and the relationship is linear (see [69]). Second, a sequence-form strategy $\mathbf{x}_i$ is equivalent to a number (precisely, a compact set) of behavioral strategies $\boldsymbol{\sigma}_i$ and the relationship is non-linear. More precisely, given an information set $h \in H_i$ and called $q \in Q_i$ the sequence leading to h, the behavioral strategy $\sigma_i(a)$ related to the actions $a \in \rho(h)$ and $q' = q|a$ is $\sigma_i(a(q')) = \frac{x_i(q')}{x_i(q)}$ if $x_i(q) > 0$ and is arbitrary otherwise. Third, a behavioral strategy $\boldsymbol{\sigma}_i$ is equivalent to a number (precisely, a compact set) of normal-form strategies $\boldsymbol{\pi}_i$ and the relationship is non-linear (see [42]). The three representations have different

degrees of expressiveness, e.g., sequence-form strategies, differently from behavioral and normal-form strategies, do not specify the actions an agent would play at the information sets that are reached with a probability of zero, and normal-form strategies, differently from behavioral strategies, correlate the actions a single agent would play at every information set.

There is a well-known relation, based on the concept of *realization*, between normal-form and sequence-form strategies. In order to exploit it, we introduce two results from [39].

Definition 2.4 (Realization equivalent) *Two strategies of an agent are* realization equivalent *if, for any fixed strategies of the other agents, both strategies define the same probabilities for reaching the nodes of the game tree.*

Proposition 2.1 *For an agent with perfect recall, for any normal-form strategy there is a realization equivalent sequence-form strategy.*

We recall in addition that each pure sequence-form strategy corresponds to a pure normal-form strategy in the reduced normal form [39].

Given a behavioral strategy $\boldsymbol{\sigma}_i$, we can derive the (realization) equivalent normal-form strategy and sequence-form strategy as follows

$$\pi_i(p) = \prod_{a \in p : p \in P} \sigma_i(a) \qquad \forall p \in P_i \tag{2.1}$$

$$x_i(q) = \prod_{a \in q : q \in Q} \sigma_i(a) \qquad \forall q \in Q_i \tag{2.2}$$

Given a normal-form strategy $\boldsymbol{\pi}_i$, we can derive the (realization) equivalent behavioral strategy:

$$\sigma_i(a) = \sum_{p \in P : a \in p} \pi_i(p) \tag{2.3}$$

Given a normal-form strategy $\boldsymbol{\pi}_i$ in *reduced* normal form, we can derive the (realization) equivalent sequence-form strategy:

$$x_i(q|a) = \sum_{p \in P : a \in p} \pi_i(p) \tag{2.4}$$

2.3 Perturbation over strategies

Trembles over strategies are captured by *perturbations*. Call $l_i(a, \epsilon) > 0$ the perturbation (in terms of probability) over action $a \in A_i$ such that $\lim_{\epsilon \to 0} l_i(a, \epsilon) = 0$ and ϵ is a positive value. We denote by $\mathbf{l}_i(\epsilon)$ the perturbation over all the agent i's actions.

Definition 2.5 (Perturbed strategy) *A perturbed strategy profile $\boldsymbol{\sigma}(\epsilon)$ of $\boldsymbol{\sigma}$ is a mixed strategy where $\sigma_i(a, \epsilon) \geq l_i(a, \epsilon)$ for all $a \in A_i$ and $\lim_{\epsilon \to 0} \boldsymbol{\sigma}(\epsilon) = \boldsymbol{\sigma}$.*

Similar to behavioral strategies, also sequence-form strategies can be subject to perturbations. We denote by $l_i(q, \epsilon) > 0$ the perturbation over sequence $q \in Q_i$ such that $\lim_{\epsilon \to 0} l_i(q, \epsilon) = 0$ and ϵ is a positive value. A perturbed strategy profile $\mathbf{x}(\epsilon)$ of $\mathbf{x}$ is a mixed strategy where $x_i(q, \epsilon) \geq l_i(q, \epsilon)$ for all $a \in Q_i$ and $\lim_{\epsilon \to 0} \mathbf{x}(\epsilon) = \mathbf{x}$.

2.4 Games with populations

In the learning scenario, we focus on extensive-form games that are repeatedly played by different individuals as described in [18]. More precisely, for each agent (representing a *role*), there is a population of *individuals* and, at each repetition of the game, one individual is drawn from each population and the drawn individuals are matched and then play the game. At each repetition, different individuals may play. A common example is a market in which bilateral negotiations are carried out: there are two agents/roles (i.e., buyer and seller), but different buyers and different sellers can match. Other economic examples are given by auctions.

In our model [26], we allow populations to be finite or infinite and to be heterogeneous including individuals of different types, where types differentiate for their preferences. We denote by $\Theta = \Theta_1 \times \cdots \Theta_{|N|}$ the set of all the types (θ denotes a generic type), where Θ_i is the set of types of agent i. We assume that the number of types is finite. We denote by Θ_{-i} the strategy profile of the agent i's opponents. Utility functions are defined also over types (in addition to the strategies of the agents). When types are *non-interdependent*, we have $u_i = u_i(\theta, \boldsymbol{\sigma}_i, \boldsymbol{\sigma}_{-i})$ where $\theta \in \Theta_i$; instead when types are *interdependent* we have $u_i = u_i(\theta, \theta', \boldsymbol{\sigma}_i, \boldsymbol{\sigma}_{-i})$ where $\theta \in \Theta_i$ and $\theta' \in \Theta_{-i}$. For

each type $\theta \in \Theta_i$, there is a (possibly infinite) population of individuals Λ_θ (we denote by λ a generic individual and by Λ_i the set of all the individuals of agent i). Each type $\theta \in \Theta_i$ is associated with a probability $\omega_{i,\theta}$ with which an individual of the pertinent population is drawn. Obviously, $\sum_{\theta \in \Theta_i} \omega_{i,\theta} = 1$. Similarly, each individual $\lambda \in \Lambda_\theta$ is associated with a probability $\omega_{i,\theta,\lambda}$ with which the individual is drawn such that $\sum_{\lambda \in \Lambda_\theta} \omega_{i,\theta,\lambda} = \omega_{i,\theta}$. For the sake of presentation, we assume that each individual has a different index λ and therefore we can refer to an individual $\lambda \in \Lambda_\theta$ by using λ in place of $\langle i, \theta, \lambda \rangle$. Similarly, we assume each type has a different θ, therefore we can refer to a type $\theta \in \Theta_i$ by using θ in place of $\langle i, \theta \rangle$.

Different individuals may adopt different strategies. $\boldsymbol{\sigma}_\lambda$ denotes the strategy of individual λ. The *aggregate strategy* of the individuals of type $\theta \in \Theta_i$ is $\boldsymbol{\sigma}_\theta = \sum_{\lambda \in \Lambda_\theta} \boldsymbol{\sigma}_\lambda \cdot \omega_\lambda$, and the aggregate strategy of agent i is $\boldsymbol{\sigma}_i = \sum_{\theta \in \Theta_i} \boldsymbol{\sigma}_\theta \cdot \omega_\theta$.

As in [18], we assume that agents have no information about the opponents. Specifically, we assume:

- each individual has no information about the utility functions of the other agents;
- each individual has no information about the individuals of the other agents;
- when utilities are interdependent, each individual knows the types of the opponents, but she does not know the pertinent probabilities and utilities;
- when utilities are not interdependent, no assumption about the knowledge of the opponents' types is made.

Customarily, a game with types is said Bayesian. Each individual forms a belief over the opponent and adjusts it during the play. More precisely, when utilities are not interdependent, each individual λ of agent i has a (potentially different) belief $\hat{\boldsymbol{\sigma}}_\lambda^{-i}$ over the strategy of agent $-i$. This is because each individual λ does not know the types and the individuals of the opponents and therefore she cannot form any belief over the single individual or type of the opponents. When utilities are interdependent, each individual λ of agent i has a (potentially different) belief $\hat{\boldsymbol{\sigma}}_\lambda^\theta$ over the strategy of type $\theta \in \Theta_{-i}$ and a (potentially different) belief $\hat{\omega}_\lambda^\theta$ over the pertinent probability ω_θ

(also in this case individuals have not beliefs over the single individuals of the opponent).

Example 2.5 *Agent 1 is of three types* $\theta_{1.1}, \theta_{1.2}, \theta_{1.3}$*, each with probability* $\omega_{1.1}, \omega_{1.2}, \omega_{1.3}$ *(with* $\omega_{1.1} + \omega_{1.2} + \omega_{1.3} = 1$*). Two individuals* $\lambda_{1.1.1}, \lambda_{1.1.2}$ *are of type* $\theta_{1.1}$ *(with* $\omega_{1.1.1} + \omega_{1.1.2} = \omega_{1.1}$*), while only one individual* $\lambda_{1.2.1}$ *is of type* $\theta_{1.2}$ *(with* $\omega_{1.2.1} = \omega_{1.2}$*) and only one individual* $\lambda_{1.3.1}$ *is of type* $\theta_{1.3}$ *(with* $\omega_{1.3.1} = \omega_{1.3}$*).*

2.5 Solution concepts

A solution concept is a formal rule for predicting how a game will be played. These predictions are called "solutions", and describe which strategies will be adopted by players and, therefore, the result of the game. In game theory there are many solution concepts but, given a specific kind of game, not all are suitable.

The central solution concept in non-cooperative game theory is the *Nash equilibrium* (NE) [49]. It constrains the strategy of each agent to be optimal given the strategies of the opponents. To formally define it, we have to introduce the concept of *best response*. Given a normal-form game and an agent i, we denote by $\boldsymbol{\pi}_{-i} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_{i-1}, \boldsymbol{\pi}_{i+1}, \ldots, \boldsymbol{\pi}_n)$ the strategy profile without the agent i's strategy. Thus we can write $\boldsymbol{\pi} = (\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i})$.

Definition 2.6 (Best response) *Agent* i*'s best response to the strategy profile* $\boldsymbol{\pi}_{-i}$ *is a mixed strategy* $\boldsymbol{\pi}_i^*$ *such that* $U_i(\boldsymbol{\pi}_i^*, \boldsymbol{\pi}_{-i}) \geq U_i(\boldsymbol{\pi}_i, \boldsymbol{\pi}_{-i})$ *for all strategies* $\boldsymbol{\pi}_i$*.*

Definition 2.7 (Nash equilibrium) *A strategy profile* $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_n)$ *is a Nash equilibrium if, for all agents* i*,* $\boldsymbol{\pi}_i$ *is a best response to* $\boldsymbol{\pi}_{-i}$*.*

A NE of an extensive-form game is defined as a NE of the associated normal form [42]. It is known that the concept of NE is not satisfactory for extensive-form games, due to two main drawbacks:

- it allows to agents to play non-credible threats [19], this leads to exploring new solution concepts based on concept of perfection: subgame-perfect equilibrium, sequential equilibrium, quasi-perfect equilibrium, perfect equilibrium and proper equilibrium,

- in real-world settings is often impractical due to its too restrictive assumptions (e.g. common knowledge of payoffs and strategies), this leads agents to have some (potentially wrong) beliefs and the solution concepts become steady states of learning processes; for these reason alternative solution concepts (epistemic-based) like self-confirming equilibria were introduced.

Refinements of Nash equilibrium

The concept of *subgame perfect equilibrium* (SPE) refines the concept of NE.

Definition 2.8 (Subgame perfect equilibrium [19]) *An SPE is a strategy profile that is an NE in every subgame.*

A subgame is a portion of the game tree defined as follows: it is composed of a single root and all the nodes reachable from the root are such that for every node $w \in V_{i,h}$ (including the root) belonging to the subgame the whole information set $V_{i,h}$ belongs to the subgame.

The concept of SPE is satisfactory with perfect-information games, while it is not when information is imperfect. The "natural" extension of the SPE to situations with imperfect information is the *sequential equilibrium* (SE). In order to introduce the concept of SE, we need to introduce *beliefs* and *assessments*. We denote by $\boldsymbol{\mu}_i = (\mu_i(1), \dots, \mu_i(w))$ for every $w \in V_{i,h}$, for every $h \in H_i$ the beliefs of agent i where $\mu_i(w)$ is the probability with which agent i believes to be at node $w \in V_{i,h}$ when she plays at information set h. Obviously, well defined beliefs μ_is are such that $\sum_{w \in V_{i,h}} \mu_i(w) = 1$ for every information set $h \in H_i$. We denote by $\boldsymbol{\mu} = (\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_{|N|})$ the profile of beliefs. An assessment, denoted by $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, is a pair specifying the beliefs and the behavioral strategies of the agents.

Definition 2.9 (Sequential equilibrium [41]) *An SE is an assessment* $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ *such that:*

- *every* $\boldsymbol{\sigma}_i$ *is* sequentially optimal *(in the sense of* backward induction *[19]) w.r.t.* $\boldsymbol{\mu}_i$*;*
- *every* $\boldsymbol{\mu}_i$ *is* consistent *(in the sense of Kreps and Wilson) w.r.t.* $\boldsymbol{\sigma}_{-i}$.

The definition of consistency resorts to perturbed strategies. Consistency of $\boldsymbol{\mu}$ w.r.t. $\boldsymbol{\sigma}$ requires that there exists a perturbed strategy profile $\boldsymbol{\sigma}(\epsilon)$ of $\boldsymbol{\sigma}$ such that, letting $\boldsymbol{\mu}(\epsilon)$ to be the sequence of beliefs derived from $\boldsymbol{\sigma}(\epsilon)$ by using the Bayes rule, $\lim_{\epsilon \to 0} \boldsymbol{\mu}(\epsilon) = \boldsymbol{\mu}$.

With perfect information every SPE is also an SE and *vice versa*. Instead, when information is imperfect, the SEs constitute a subset of the SPEs.

Example 2.6 *Consider the game depicted in Fig. 2.1. The pure strategy SEs (and SPEs, being the same) is:* $(\sigma_1(\mathsf{L}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_2(\mathsf{l}_1) = 1), (\sigma_1(\mathsf{L}_1) = 1, \sigma_1(\mathsf{R}_2) = 1, \sigma_2(\mathsf{l}_1) = 1), (\sigma_1(\mathsf{R}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_2(\mathsf{l}_1) = 1), (\sigma_1(\mathsf{R}_1) = 1, \sigma_1(\mathsf{R}_2) = 1, \sigma_2(\mathsf{l}_1) = 1)$.

The idea of *perfection*, introduced by Selten in [56], is strictly correlated with the idea of perturbed strategy. Basically, a strategy profile is perfect when it is optimal even when there are perturbations. The common interpretation of the perturbations is based on the idea that agents do not perfectly play a strategy, but they can tremble with a very small probability. There are three concepts of perfection.

Definition 2.10 (Normal-form perfect equilibrium [56]) *A strategy profile $\boldsymbol{\pi}$ is a NFPE if there exists a perturbed strategy profile $\boldsymbol{\pi}(\epsilon)$ of $\boldsymbol{\pi}$ such that $\pi_i(p, \epsilon) \geq l_i(p, \epsilon)$ and $\boldsymbol{\pi}_i$ is a best response to $\boldsymbol{\pi}_{-i}(\epsilon)$ for every $\epsilon \leq \overline{\epsilon}$ for some $\overline{\epsilon} > 0$.*

Every game admits at least one NFPE and, for every combination of vectors of perturbation $\mathbf{l}_i(\epsilon)$, there is a potentially different NFPE. Normal-form perfection can be captured also by posing a generic (fully mixed) perturbation over the sequences of the sequence form.

Definition 2.11 (Quasi-perfect equilibrium [65]) *A strategy profile $\boldsymbol{\sigma}$ is a QPE if there exists a perturbed strategy profile $\boldsymbol{\sigma}(\epsilon)$ of $\boldsymbol{\sigma}$ such that $\sigma_{i,a}(\epsilon) \geq l_i(a, \epsilon)$ and every $\boldsymbol{\sigma}_i$ is a best response to $\boldsymbol{\sigma}_{-i}(\epsilon)$ for every $\epsilon \leq \overline{\epsilon}$ for some $\overline{\epsilon} > 0$.*

In a QPE every agent takes into account opponents' trembles, but not their own. Every game admits at least one QPE and for every combination of $\mathbf{l}_i(\epsilon)$ there is a potentially different QPE. The authors show in [48] that quasi perfection can be captured by using a specific class of perturbations with the sequence form. In this case, we denote by $\mathbf{x}_i(\epsilon)$ the perturbed sequence form strategy and by $x_i(q, \epsilon)$ the

perturbed strategy over $q \in Q_i$. Call $\mathbf{l}_i(\epsilon^k)$ the coefficients of ϵ^k in $\mathbf{l}_i(\epsilon)$.[2] Quasi perfection can be captured imposing that for every pair of sequences $q, q' \in Q_i$ where q' extends q the minimum degree of k such that $l_i(q', \epsilon^k)$ is strictly positive is strictly smaller than the minimum degree of k such that $l_i(q, \epsilon^k)$ is strictly positive.

Definition 2.12 (Extensive-form perfect equilibrium [56]) *A strategy profile $\boldsymbol{\sigma}$ is an EFPE if there exists a perturbed strategy profile $\boldsymbol{\sigma}(\epsilon)$ of $\boldsymbol{\sigma}$ such that $\sigma_i(a, \epsilon) \geq l_i(a, \epsilon)$ and every $\boldsymbol{\sigma}_i$ is a best response to $\boldsymbol{\sigma}(\epsilon)$ for every $\epsilon \leq \bar{\epsilon}$ for some $\bar{\epsilon} > 0$.*

This concept strengthens the concept of QPE, constraining $\boldsymbol{\sigma}_i$ to be a best response, in addition to $\boldsymbol{\sigma}_{-i}(\epsilon)$, also to $\boldsymbol{\sigma}_i(\epsilon)$. As above, every game admits at least one EFPE and for every combination of $\mathbf{l}_i(\epsilon)$ there is a potentially different EFPE. It is not known whether the constraints of an EFPE can be formulated with the sequence form. More precisely, it is well known that the use of additive perturbations over the sequence form, such as those for the QPE, cannot be used for an EFPE. This is because given any additive perturbation over the sequences, it is not possible to derive a perturbation over the behavioral strategies that is independent of the strategy itself (as it would be required by the definition of EFPE). However, it is not known whether alternative perturbations can be used.

Example 2.7 *Consider the game represented in Fig. 2.1. The set of NFPEs coincides with the set of SEs. The pure strategy QPEs are: $(\sigma_1(\mathsf{L}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_2(\mathsf{l}_1) = 1), (\sigma_1(\mathsf{R}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_2(\mathsf{l}_1) = 1)$. Notice that $(\sigma_1(\mathsf{R}_1) = 1, \sigma_1(\mathsf{R}_2) = 1, \sigma_2(\mathsf{l}_1) = 1)$ is an SE, but is not a QPE. This is because, accounting for any perturbed $\sigma_2(\mathsf{l}_1, \epsilon)$, the utility expected by agent 1 from making action R_2 (i.e., $\sigma_2(\mathsf{l}_1, \epsilon) < 1$) is strictly smaller than the utility she expects from making action L_2 (i.e., 1). The unique EFPE is: $(\sigma_1(\mathsf{L}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_2(\mathsf{l}_1) = 1)$. Indeed, accounting also any perturbed strategy $\sigma_1(\mathsf{L}_2, \epsilon)$, the utility expected by agent 1 from making action R_1 (i.e., $\sigma_1(\mathsf{L}_2, \epsilon) < 1$) is strictly smaller than the utility she expects from making action L_1 (i.e., 1).*

The idea of properness introduces additional constraints over the perturbation w.r.t. the three previous concepts: if two actions, say

[2] Analogously, we use $l_i(q, \epsilon^k)$ to consider the perturbation restricted to sequence q, and we use $\mathbf{x}_i(\epsilon^k)$ and $x_i(q, \epsilon^k)$ as for the case of behavioral strategies.

a, a', give different expected utility to agent i, then the perturbation over the worse action, say a, must be $l_i(a, \epsilon) \leq \epsilon \cdot l_i(a', \epsilon)$. In other words, the tremble of an agent over its worse actions must be smaller than the tremble over its best actions. Two different concepts of proper equilibrium exist: the *normal-form proper equilibrium* (NFPrE) refining the NFPE and the *extensive-form proper equilibrium* (EFPrE) refining the EFPE, differing for the strategy representation to which the concept of properness is applied.

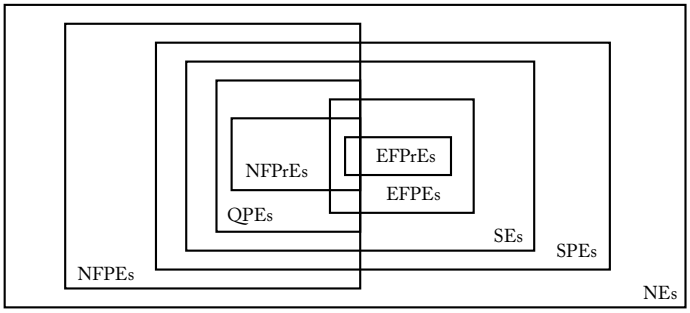


Figure 2.2: Relationships between the solution concepts.

We briefly survey the relationships between the solution concepts (see Fig. 2.2). All the above solution concepts refine the NE and the SPE (except for the NFPE). QPEs constitute a subset of NFPEs. QPEs and EFPEs are subsets of SEs. Although the concept of EFPE can seem to be a refinement of the concept of QPE, this is not the case [46]. Indeed, the set of EFPEs is not strictly included in the set of QPEs, but some EFPEs may not be QPEs as discussed below in Example 2.8. Finally, NFPrEs constitute a subset of QPEs and EFPrEs constitute a subset of EFPEs.

Example 2.8 *Consider the game represented in Fig. 2.3, where two agents choose who will be the agent (i.e., 'me' or the 'other') that will take a decision (i.e.,* R*/*r *or* W*/*w*). The game presents a chance node. The EFPEs are* $(\sigma_1(\mathsf{O}) = 1, \sigma_1(\mathsf{R}) = 1, \sigma_2(\mathsf{m}) = 1, \sigma_2(\mathsf{r}) = 1), (\sigma_1(\mathsf{M}) = 1, \sigma_1(\mathsf{R}) = 1, \sigma_2(\mathsf{o}) = 1, \sigma_2(\mathsf{r}) = 1)$. *The unique QPE is* $(\sigma_1(\mathsf{M}) = 1, \sigma_1(\mathsf{R}) = 1, \sigma_2(\mathsf{m}) = 1, \sigma_2(\mathsf{r}) = 1)$.

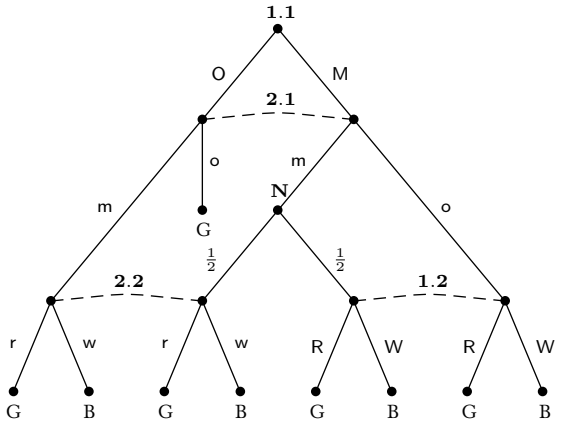


Figure 2.3: N is nature ($\frac{1}{2}$ is a probability); G and B are outcomes: both players prefer G to B. In this game, proposed in [46], the set of EFPEs is not included in the set of QPEs.

Relaxations of Nash equilibrium

The concept of *self-confirming equilibrium* (SCE) [18] relaxes NE, capturing settings where opponents' utilities and types are unknown. In order to introduce the concept of SCE, we need to introduce the *belief over the opponents' strategies*. We denote by $\hat{\boldsymbol{\sigma}}_{-i}$ the agent i's belief over the opponents' strategies $\boldsymbol{\sigma}_{-i}$. Obviously, well-defined beliefs $\hat{\boldsymbol{\sigma}}_i$s are such that $\sum_{a \in \rho(h)} \hat{\sigma}_i(a) = 1$ for every information set $h \in H_i$. We denote by $\hat{\boldsymbol{\sigma}} = (\hat{\boldsymbol{\sigma}}_1, \ldots, \hat{\boldsymbol{\sigma}}_{|N|})$ the profile of beliefs.

Definition 2.13 (Self-confirming equilibrium [18]) *An SCE is a pair* $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$ *such that:*

- *for every agent* i, $\boldsymbol{\sigma}_i$ *is a best response to* $\hat{\boldsymbol{\sigma}}_{-i}$,
- *for every agent* i, $\hat{\boldsymbol{\sigma}}_{-i}$ *is equal to* $\boldsymbol{\sigma}_i$ *on the equilibrium path.*

Similarly to an SE, an SCE is a pair $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$ where $\boldsymbol{\sigma}$ are best responses to $\hat{\boldsymbol{\sigma}}$. Differently, $\hat{\boldsymbol{\sigma}}$ can be wrong off the equilibrium path (instead they must be correct/confirmed on). Thus, every NE is an SCE, while an SCE may be not an NE.

2.6 Computational complexity classes

Two of the most important complexity classes are $\mathcal{P}$ (polynomial) and $\mathcal{NP}$ (nondeterministic polynomial). The former is the class of problems that can be solved efficiently, such as finding shortest paths in networks, parse context-free grammars, matrix multiplication, sorting, an so on. It contains all the problems that can be solved by a deterministic Turing machine in polynomial time. The latter is the class of problems whose solutions can be verified quickly, by a deterministic Touring machine in polynomial time, but a solution is found in polynomial time by a nondeterministic Touring machine, e.g., SAT. The problems in $\mathcal{NP}$ return a YES/NO answer.

When dealing with problems whose answer is not YES/NO answer, but it is the value of a function, other complexity classes are used. The first is $\mathcal{FP}$ (function polynomial) that is the class of function problems that can be computed by a deterministic Touring machine in polynomial time (it is a function problem version of the decision problem class $\mathcal{P}$). The second is $\mathcal{FNP}$ (function nondeterministic polynomial).

Formally, let $R \subseteq \Sigma^* \times \Sigma^*$ be a polynomial-time computable, polynomially balanced relation, that is, there exists a polynomial p such that for any x and y satisfy $(x, y) \in R$ implies $|y| \leq p(|x|)$. The *search problem* associated with R is: Given input $x \in \Sigma^*$, return a $y \in \Sigma^*$ such that $(x, y) \in R$, if such a y exists, and return the string "no" otherwise. $\mathcal{FNP}$ is the class of all such search problems. $\mathcal{FP}$ is the subclass of $\mathcal{FNP}$ containing all those search problems that can be solved in polynomial time.

There is another interesting class that contains some important problems is $\mathcal{TFNP}$ (total function nondeterministic polynomial), that lies between $\mathcal{FNP}$ and $\mathcal{FP}$ and contains the problems where a solution is guaranteed to exist. It is clear that $\mathcal{FP} \subseteq \mathcal{TFNP} \subseteq \mathcal{FNP}$, and it is open whether these inclusions are strict [50]. But telling whether R is total is undecidable, so $\mathcal{TFNP}$ is a "semantic" class (i.e., it has no generic complete problem). For this reason many syntactic subclasses of $\mathcal{TFNP}$ were introduced: $\mathcal{PLS}$, $\mathcal{PPA}$, $\mathcal{PPP}$, $\mathcal{PPAD}$. All the problems in these classes are *total search* and the proof of existence is based on a specific lemma for each class. In order to show that some problems discussed in this thesis belong to a specific computational complexity class, we need to introduce formally the class $\mathcal{PPAD}$.

Polynomial parity arguments on directed graphs

Papadimitriou defined in [50] a complexity class, *polynomial parity argument on directed graph* ($\mathcal{PPAD}$), that captures the basic principles of path-following algorithms: There is a finite number of candidate solutions, and an underlying directed graph of moves between the solutions where each solution has at most one forward and one backward move, i.e., the graph consists of a set of directed paths, cycles and isolated nodes; a source of one path is an artificial starting solution, and every other endpoint (source or sink) of every path is an answer to the problem. Therefore, totality for functions in $\mathcal{PPAD}$ is established by invoking the following lemma.

Lemma 2.1 *If a directed graph has an unbalanced node (indegree ≠ outdegree), then it must have another unbalanced node.*

Formally in [74], a problem Π is in $\mathcal{PPAD}$ if each instance I has a set $S(I)$ of solutions which are (strings) polynomially bounded in the input size $|I|$, and there are polynomial time algorithms for the following tasks:

- test whether a given string I is an instance of Π and if so compute a (initial) solution in $x_0 \in S(I)$,
- given I, x, test whether $x \in S(I)$ and if so compute a successor $succ_I(x) \in S(I)$ and a predecessor $pred_I(x) \in S(I)$, such that $pred_I(x_0) = x_0$, $succ_I(x_0) \neq x_0$, and $pred_I(succ_I(x_0)) = x_0$.

The *pred* and *succ* functions induce a directed graph $G = (S(I), E)$ where $E = \{u, v | u \neq v, succ_I(u) = v, pred_I = u\}$ and the set of answers to the instance I is the set of nodes of G other than x_0 that have indegree + outdegree = 1.

There is a problem, so called END-OF-THE-LINE, strongly correlated with the $\mathcal{PPAD}$ complexity class. It is the following: given two circuits S and P, each with n input and output bits, such that $S(P(0^n)) \neq 0^n = P(S(0^n))$, find an input $x \in \{0,1\}^n$ such that $P(S(x)) \neq x$ or $S(P(x)) \neq x \neq 0^n$.

Proposition 2.2 *$\mathcal{PPAD}$ is the class of all the problems solvable by a path-following algorithm, e.g., Lemke–Howson's and Lemke's algorithms*

[43, 44] or, on the other hand, the class of problems that are polynomial reducible to the END-OF-THE-LINE problem.

2.7 Stochastic games and solution concepts

A *stochastic game* $\mathcal{G}$, introduced in [58], is a tuple (N, S, X, P, R, γ), where: S is a finite set of states ($s \in S$ denotes a generic state), N is the set of agents ($i \in N$ denotes a generic agent), X is the set of actions ($X_{i,s} \subseteq X$ denotes the set of actions available to agent i at state s, and $x_i \in X_{i,s}$ a generic action of agent i), P is a set of maps $p_{s,s'} : \times_{i \in N} X_{i,s} \to [0, 1]$ assigning the probability to move from state s to state s' given the actions of all the agents, R is a set of maps $r_s : \times_{i \in N} X_{i,s} \to \mathbb{R}$ assigning each action profile a reward, $\gamma \in (0, 1)$ is the temporal discount factor. To the aim of our work, we focus on two-player zero-sum stochastic-games and without loss of generality, we assume that agent 1 is the max agent. We denote with u_s, where $s \in S$, the utility function of agent 1, while $-u_s$ is the utility function of agent 2. By Bellman equation, u_s is defined as $u_s(\mathbf{x}_1, \mathbf{x}_2) = r_s(x_{1,s}, x_{2,s}) + \sum_{s'} p_{s,s'}(x_{1,s}, x_{2,s}) \cdot u_{s'}(\mathbf{x}_1, \mathbf{x}_2)$, where $\mathbf{x}_i$ is the vector of actions of agent i over all the states and $x_{i,s}$ is the specific action of agent i at state s.

A *continuous* stochastic game is a stochastic game (N, S, X, P, R, γ) where action spaces $X_{i,s}$ are limited subspaces (usually compact) of the Euclidean space and maps $p_{s,s'}$ and r_s are generic functions. We focus on continuous stochastic games with *switching controller*, in which at each state s functions $p_{s,s'}$ depend either on $x_1 \in X_{1,s}$ or on $x_2 \in X_{2,s}$. The agent i who drives the transition at state s is said the *controller* of such state and it is denoted by c_s. Notice that different states can be controlled by different agents. We partition the state space S as $S = S_1 \cup S_2$ where S_i is the set of states where $c_s = i$. When the game is polynomial, we have: $p_{s,s'}(x_{c_s}) = \sum_{k=0}^{m} p_{s,s',k} \cdot (x_{c_s})^k$ and $r_s(x_1, x_2) = \sum_{k=0}^{m} \sum_{j=0}^{m} r_{s,k,j} \cdot (x_1)^k \cdot (x_2)^j$, where m is the maximum degree of all the polynomials and $p_{s,s',k}, r_{s,k,j} \in \mathbb{R}$ are coefficients.

A strategy profile $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ specifies the strategy $\sigma_{i,s}$ of each agent i at each state s as a probability measure over the space of actions $X_{i,s}$. A *Markov perfect equilibrium* (MPE) is a strategy profile $(\boldsymbol{\sigma}_1^*, \boldsymbol{\sigma}_2^*)$ where each strategy is conditioned only on the local state of the agent, and such that no agent can improve her utility u_s (or

$-u_s$) in any state s by changing her strategy. With zero-sum games, an MPE corresponds to maxmin/minmax strategies. In this paper, we resort also to the ϵ-MPE concept, defined as: a strategy profile $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ is an ϵ-MPE if no agent can improve her utility u_s (or $-u_s$) in some state s more than ϵ by changing her strategy. Obviously, an ϵ-MPE with $\epsilon = 0$ is an MPE. Furthermore, while an MPE may not exist with continuous games, it is always possible to find ϵ-MPEs for some ϵ.

2.8 Evolutionary game theory

Evolutionary game theory (EGT) is the application of game theory to evolving populations of lifeforms in biology. EGT is useful in this context by defining a framework of contests, strategies, and analytics into which Darwinian competition can be modelled. In EGT a game is played over and over again by biologically or socially conditioned agents who are randomly drawn from large populations and each agent is programmed to play a specific strategy (in general a pure strategy) [73]. The populations evolving during the repetition of the game because each agent as a law of reproduction relating to the fitness of the agent's strategy. The fitness is a function of utility of the agent's strategy w.r.t. the opponent's strategy, greater the agent's utility, greater the offspring (in terms of number of agent with the same strategy in the new population). An evolutionary process combines two fundamental concepts: a *mutation mechanism* that provides variety and a *selection mechanism* that favors some varieties over others.

The most important concept that highlights the role of mutation is the *evolutionary stable strategies* (ESS) that was introduced in [61]. Given a population Φ_i we denote by $\phi_i(a)$ the number of agents of Φ_i that play action a and by $\pi_i(a) = \frac{\phi_i(a)}{|\Phi_i|}$ the fraction of population that plays a. Suppose that a population is playing the strategy $\boldsymbol{\pi}_i$ and a small fraction ϵ of this population, called *mutant*, changes her mind and it plays a new strategy $\boldsymbol{\pi}'_i$. The strategy $\boldsymbol{\pi}_i$ is said ESS if it is robust to the invasion, means that the reproductive success of the new strategy $\boldsymbol{\pi}'_i$ is smaller than the original one and it will not overrule the original strategy and will eventually disappear.

Definition 2.14 (Evolutionary stable strategy [61]) *$\boldsymbol{\pi}_i$ is an EES if*

for every $\boldsymbol{\pi}'_i$ *there exists some* $\overline{\epsilon}_{\boldsymbol{\pi}'_i} \in (0,1)$ *such that*

$$u_i(\boldsymbol{\pi}_i, \epsilon\boldsymbol{\pi}'_i + (1-\epsilon)\boldsymbol{\pi}_i) > u_i(\boldsymbol{\pi}'_i, \epsilon\boldsymbol{\pi}'_i + (1-\epsilon)\boldsymbol{\pi}_i)$$

holds for all $\epsilon \in (0, \overline{\epsilon}_{\boldsymbol{\pi}'_i})$.

On the other hand, the fundamental concept that shows the importance of selection mechanism is the replicator dynamics. In the standard formulation it is a system of ordinary equations without any mutation mechanism. It describes the dynamics of populations that play repeated games, it can be defined in a discrete or in a continuous time interval.

Definition 2.15 (Replicator dynamics [15]) *The standard discrete-time replicator equation with two agents is:*

$$\pi_1(p, t+1) = \pi_1(p,t) \cdot \frac{\mathbf{e}_p^T \cdot U_1 \cdot \boldsymbol{\pi}_2(t)}{\boldsymbol{\pi}_1^T(t) \cdot U_1 \cdot \boldsymbol{\pi}_2(t)} \tag{2.5}$$

$$\pi_2(p, t+1) = \pi_2(p,t) \cdot \frac{\boldsymbol{\pi}_1^T(t) \cdot U_2 \cdot \mathbf{e}_p}{\boldsymbol{\pi}_1^T(t) \cdot U_2 \cdot \boldsymbol{\pi}_2(t)} \tag{2.6}$$

while the continuous-time one is

$$\dot{\pi}_1(p) = \pi_1(p) \cdot [(\mathbf{e}_p - \boldsymbol{\pi}_1)^T \cdot U_1 \cdot \boldsymbol{\pi}_2] \tag{2.7}$$

$$\dot{\pi}_2(p) = \pi_2(p) \cdot [\boldsymbol{\pi}_1^T \cdot U_2 \cdot (\mathbf{e}_p - \boldsymbol{\pi}_2)] \tag{2.8}$$

Where $\mathbf{e}_p$ is the vector in which the p-th component is "1" and the others are "0".

2.9 Q-learning

$\mathcal{Q}$-learning is an algorithm proposed in [72] and it is used by learning agents in Markovian domains that allows them to learn the optimal policy without having prior knowledge about a map of the domain. $\mathcal{Q}$-learning is an adaptive model-free value-iteration method whose aim is to estimate the value $\mathcal{Q}_{t+1}(s,a)$ of each action a in each state s at time $t+1$ given the estimations of $\mathcal{Q}_t(s',a')$, where s' is any state reached by taking action a in s and a' is the greedy action in s', through the following rule:

$$\mathcal{Q}_{t+1}(s,a) = (1-\alpha)\mathcal{Q}_t(s,a) + \alpha\left(r(s,a) + \gamma \max_{a'} \mathcal{Q}_t(s',a')\right),$$

where $r(s,a)$ is the immediate reward received being in state s by taking action a, α is the learning rate and γ is the discount factor [63]. Given the set of rewards,

$$\mathcal{Q}_t(s,a) \in \left[\frac{\min_{s',a'} r(s',a')}{1-\gamma}, \frac{\max_{s',a'} r(s',a')}{1-\gamma}\right] \quad \forall s \in S, a \in A. \tag{2.9}$$

The evolutionary perspective of $\mathcal{Q}$-learning [64]

The time limit of the $\mathcal{Q}$-learning model, when agents play in self play stateless repeated normal-form games and the action-selection mechanism is based on the Boltzmann distribution, is the replicator dynamics with a mutation term. More precisely, the updating rule used in these cases is:

$$\mathcal{Q}_{t+1}(p) = (1-\alpha)\mathcal{Q}_t(p) + \alpha(r(p) + \gamma \max_{p'} \mathcal{Q}_t(p')), \tag{2.10}$$

where $r(p)$ is the reward of plan p, defined as the utility $U_i(p, p_{-i})$ where p_{-i} are the plans actually played by the other agents. In this case we have

$$\mathcal{Q}_t(p) \in \left[\frac{\min_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})}{1-\gamma}, \frac{\max_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})}{1-\gamma}\right] \qquad \forall p \in P. \tag{2.11}$$

The Boltzmann action-selection strategy is:

$$\pi_i(p,t) = \frac{e^{\tau \mathcal{Q}_t(p)}}{\sum_{p' \in P_i} e^{\tau \mathcal{Q}_t(p')}}, \tag{2.12}$$

where τ is the exploitation parameter[3] of the $\mathcal{Q}$-learning algorithm and the time-limit dynamics in expectation of two agents are described by the following replicator:

[3]Note that the exploitation parameter τ is the inverse of the commonly known temperature T. In [64] it is considered as a constant.

$$\dot{\pi}_1(p,t) = \alpha\tau\pi_1(p,t)\cdot[(\mathbf{e}_p - \boldsymbol{\pi}_1(t))^T \cdot U_1 \cdot \boldsymbol{\pi}_2(t)] + \alpha\pi_1(p,t)\cdot \sum_{p'\in P_1} \pi_1(p',t)\log\left(\frac{\pi_1(p',t)}{\pi_1(p,t)}\right) \quad (2.13)$$

$$\dot{\pi}_2(p,t) = \alpha\tau\pi_2(p,t)\cdot[\boldsymbol{\pi}_1(t)^T \cdot U_2 \cdot (\mathbf{e}_p - \boldsymbol{\pi}_2(t))] + \alpha\pi_2(p,t)\cdot \sum_{p'\in P_2} \pi_2(p',t)\log\left(\frac{\pi_2(p',t)}{\pi_2(p,t)}\right), \quad (2.14)$$

where $\alpha\tau\pi_i(p,t)\cdot[(\mathbf{e}_p-\boldsymbol{\pi}_i(t))^T\cdot U_i\cdot\boldsymbol{\pi}_{-i}(t)]$ is the selection term and $\alpha\pi_i(p,t)\cdot\sum_{p'\in P_i}\pi_i(p',t)\log\left(\frac{\pi_i(p',t)}{\pi_i(p,t)}\right)$ is the mutation term [64].

State of the art 3

The design of intelligent autonomous agents that are able to act optimally in strategic-interaction situations is one of the most important tasks in *artificial intelligence* because they can be used to play in many real-world situations. Many examples can be found in different scenarios:

- from *security*, a robot that has to find survivors after a military attack or a robot that has to protect a resource from an enemy attack [5, 40],
- from *economy*, an agent that maximises the gain from the sale of certain assets or a system that optimally allocate online advertising [35, 52],
- from *games*, software agents that are able to play chess, poker or billiard [1, 28].

The solution concept of a game is not unique and depends on the assumptions underlying the real-world situation represented by the game. For this reason different *solution concepts* were introduced in microeconomics. We need to characterise them from a computational point of view to apply to real-world situations because if we cannot efficiently compute a solution concept we have to resort to

approximate solution concepts. The problem of characterise the solution concepts is the central one in *equilibrium computation*. This characterisation include the following problems:

- the *verification problem*: given the agents' strategies verify whether the strategies fulfil the requirements of a specific solution concept; the *desideratum* is that the verification problem is in $\mathcal{P}$ because that means the problem of searching for the solution concept is in $\mathcal{FNP}$ thus the solution concept is efficiently computable,
- the *computational problem*: compute the agents' strategies according to specific solution concept; the *desideratum* is that the computation is as efficient as possible; in the case where the solution concept is a NE refinement, the complexity class of the problem have to be at most the same class of computation of NE, i.e. $\mathcal{PPAD}$, and
- the *learning problem*: the *desideratum* is designing an efficient algorithm where the learning strategies converge to a specific solution concept.

In this thesis, we work with non-cooperative extensive-form games. This choice is due to two reasons: first this representation is more expressive than the strategic-form one, second the results present in game theoretic literature are incomplete and, in general, negative from a computational point of view.

The central solution concept in non-cooperative game theory is the *Nash equilibrium* (NE). It constrains the strategy of each agent to be optimal given the strategies of the opponents. When an extensive-form game represents a scenario where the agents have common knowledge. the NE is not suitable, allowing strategies to be non-sequentially rational. For this reason the game theoretic literature provides a number of appropriate solution concepts in the scenarios where the assumption of NE are verified, each of them appropriate for a specific situation [66]:

- the *sequential equilibrium* (SE) [41] that requires the sequential rationality of the strategies (i.e., at each stage of game each agent will do her best - she chooses the action that maximise her own utility); it is commonly adopted in dynamic bargaining [22] and signaling games [19],

- the *quasi-perfect equilibrium* (QPE) [65] that prescribes for every agent an optimal strategy that takes into account the possibility of opponent's mistakes (i.e. the agent's opponents can take some wrong action that do not maximise the opponents' utility)and

- the *extensive-form perfect equilibrium* (EFPE) [56] that requires the optimality of the strategies when all the agents can take mistakes.

On the other hand, there are many real-world situations where the agents do not have common knowledge. In these scenarios the NE is again the non-appropriate solution concept because without common knowledge the agents are unable to take the optimal decision w.r.t. the real scenario, but they can only play optimally given the partial knowledge of the environment and also the the solution concepts previously introduced, that require common knowledge assumptions, are not suitable. The aforementioned drawbacks of NE pushed researchers to design alternative (relaxed and/or non-equilibrium) solution concepts taking also into account learning dynamics. With extensive-form games, relaxing the epistemic requirements of NE, it is possible to have stable (equilibrium) states that are not NEs [18]. These equilibria, called *self-confirming equilibria* (SCEs), require that each agent plays optimal strategies to her beliefs, but the beliefs can be *incorrect* off the equilibrium path (while they are *confirmed on*).

The game theoretical **state of the art about the verification problem** is: the verification problem for NE is known to be easy just requiring the verification that a finite number of constraints are satisfied. The only known result on the verification for SE is [38]. The authors provided a finite-step algorithm to verify whether an assessment is an SE, but, as they state it, the algorithm is exponential in the worst case. Furthermore, it is known that, when the input is partially specified, the verification problem for an SE can be harder. More precisely, the authors show in [32] that with three or more agents, verifying whether there is an SE with a given strategy is $\mathcal{NP}$-hard. The verification problem for QPE is $\mathcal{NP}$-hard with three or more agents [32]. The verification problem for EFPE is open with two agents. With three or more agents the problem is shown to be $\mathcal{NP}$-hard [32]. The problem of verifying whether a strategy profile is a

perfect equilibrium for a given perturbation is easy for all the perfect equilibrium solution concepts (i.e., QPE and EFPE). Indeed, it requires one only to check whether a finite number of constraints are satisfied.

The game theoretical **state of the art about the searching problem** is: the computation of an NE with two-agent zero-sum games can be easily tackled by resorting to linear programming, computing a maxmin/minmax strategy. Instead, solving general-sum games is harder. The main results on the computation of an NE are with two agents. The problem to search for an NE is formulated as a LCP and solved by employing the Lemke's algorithm. An alternative LCP approach to compute an NE is proposed in [70]. The complexity class is $\mathcal{PPAD}$-complete. It is not known whether or not there is a polynomial time algorithm to compute an NE, but it is commonly believed it is not. The computation of an SE is open [32]. The computation of an SE with two agents can be done by computing a QPE, but there is no algorithm to find a generic SE. In [48] the authors use the Lemke's algorithm with perturbations to compute a QPE when agents are two. This shows that the problem to compute a QPE with two agents is in $\mathcal{PPAD}$ when a specific perturbation is given. The problem of computing an EFPE without an input perturbation over the strategies is open. In particular, it is not known whether this problem can or cannot be formulated with the sequence form. In the latter case, the problem to compute an EFPE would require non-linear optimization tools and, the problem not being convex, no exact solution could be computed.

The **results originally provided** in the present thesis are: the problem of verifying whether if a strategy profile is an SE is in $\mathcal{P}$, in two-player case with the strategy profile as input and when there are more than two players with the assessment as input. This means that the SE is a suitable solution concept for extensive-form games from a computational point of view and push the research to develop an algorithm for the computational problem of finding a SE. The problem of verifying whether a strategy profile is a QPE with two agents is in $\mathcal{P}$. The algorithm to solve this problem is a variation of the algorithm to verify if a strategy profile with two agents is a SE. This result joint with the previous one suggest that a variation of the algorithm to find a QPE can be used to find a SE. If true, the problem of computing an SE would be efficiently solved. The computation of an EFPE is $\mathcal{PPAD}$ and this result is surprising because the definition of pertur-

bation due to Selten [56] is a multiagent problem, where there is an independent player in each node and this suggests that the computation cannot be efficient. The verification problem related to EFPE is currently open. When the common knowledge assumption is not verified, the agent can efficiently learn the optimal strategy using a variation of Q-learning which expected behaviour is characterisable with a system of differential equations, the replicator dynamics of evolutionary game theory.

In Tab. 3.1 we summarise the results present in game theoretic literature and the contributions presented in this thesis (a detailed analyses is present in the following chapters). Specifically, the verification problem is separately addressed with two agents and with more than two agents because the two cases are radically different, while the computational problem is addressed only with two agents because with more than two agents the solution is in general irrational, thus it cannot be represented in efficient amount of space.

	Verification		Computation	Learning	
EFPE	$\geq 2p$	[32]	(*)		Common knowledge
	$2p$				
QPE	$\geq 2p$	[32]	[48]		
	$2p$	(*)			
SE	$\geq 2p$	(*)			
	$2p$	(*)			
NE	$\geq 2p$	[49]	[43]		
	$2p$	[49]			
SCE	(*)		(*)	(*)	

Table 3.1: State of the art and original contributions of verification, computation and learning problems with extensive-form games; the results with '(*)' are originally provided in the present thesis.

Part I

Verification and computation of Nash equilibrium refinements

Efficient algorithms for the verification problem of sequential equilibrium 4

In this chapter, we focus on the problem of verifying whether a given solution is a SE. The study of the verification problem is of extraordinary importance, because it is necessary to certify that a software agent is playing or not the optimal strategy.

In this chapter, we show that

- when the assessment is given as input, there is an efficient algorithm certifying that the assessment is an SE with an arbitrary number of agents,

- when only the behavioral strategy profile is given as input, there is an efficient algorithm certifying that the strategy profile is an SE with two-agent games; this algorithm can be employed also to derive the agents' consistent (in the sense of Kreps and Wilson) beliefs from the strategies and a simple variation can be employed for the verification of QPE with two-agent games.

The chapter is structured as follows. In Section 4.1, we study the verification problem for SE when an assessment is given as input. In Section 4.2, we study the verification problem for SE with two agents when only the strategy profile is given as input and we show

how our result can be extended to QPE. In Section 4.3, we survey related works.

4.1 SE verification when the assessment is given

We focus on the problem of deciding whether or not an assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ is an SE when the number of agents is arbitrary. We recall that $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ is an SE if and only if:

- $\boldsymbol{\sigma}$ is sequentially rational w.r.t. $\boldsymbol{\mu}$, and
- $\boldsymbol{\mu}$ is consistent w.r.t. $\boldsymbol{\sigma}$.

Initially, we focus on the sequential rationality of the strategies and we state the following proposition.

Proposition 4.1 *Given $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, there exists an efficient algorithm certifying that $\boldsymbol{\sigma}$ is sequentially rational w.r.t. $\boldsymbol{\mu}$.*

Proof. We say that a decision node $w \in V$ is *penultimate* if, for every $a \in \rho(w)$, $\chi(w, a) \in T$ (i.e., $\chi(w, a)$ is a terminal node). Furthermore, we denote by $\Omega(a)$ the probability with which the nature performs action a once decision node w such that $a \in \rho(w)$ is reached. Sequential rationality of $\boldsymbol{\sigma}$ can be verified by means of a variation of the backward induction algorithm [19] incorporating $\boldsymbol{\mu}$ as described in Algorithm 1. Basically, Algorithm 1 verifies, at each information set h such that all the decision nodes belonging to $V_{\iota(h),h}$ are penultimate, that $\boldsymbol{\sigma}_{\iota(h)}$ is optimal in expectation w.r.t. beliefs $\mu_{\iota(h)}(w)$ where $w \in V_{\iota(h),h}$. Algorithm 1 requires a number of maximizations that is linear in the size of the game, and each single maximization is over a number of elements (i.e., actions) that is linear in the size of the game. This completes the proof of the proposition. □

Now, we focus on the consistency of the beliefs. Verifying consistency of $\boldsymbol{\mu}$ w.r.t. $\boldsymbol{\sigma}$ requires one to find a fully mixed perturbed strategy profile $\boldsymbol{\sigma}(\epsilon)$ with the properties that the beliefs $\boldsymbol{\mu}(\epsilon)$ derived from $\boldsymbol{\sigma}(\epsilon)$ by Bayes rule converge to $\boldsymbol{\mu}$ as ϵ goes to zero. Kreps and Wilson show that the problem of searching for such a $\boldsymbol{\sigma}(\epsilon)$ can be formulated as the problem of searching for a *b-labeling*. Initially, we introduce the following definition.

Algorithm 1 verifySequentialRationality($\boldsymbol{\mu}, \boldsymbol{\sigma}$)

1: **for all** information sets h such that all the nodes $w \in V_{\iota(h),h}$ are penultimate **do**
2: **if** $\iota(h)$ is the nature **then**
3: eliminate all the nodes following $w \in V_{\iota(h),h}$ from the game tree and consider w as terminal nodes with utility $u_i(w) = \sum_{a \in \rho(w)} \Omega(a) \cdot u_i(\chi(w,a))$ for each agent i
4: **if** $\iota(h)$ is not the nature **then**
5: **if** there is some action a' such that $\sigma_{\iota(h)}(a') > 0$ and $a' \notin \arg\max_{a \in \rho(h)} \sum_{w \in V_{\iota(h),h}} \mu_{\iota(h)}(w) \cdot u_{\iota(w)}(\chi(w,a))$ **then**
6: **return** non-sequentially rational
7: **for all** $w' \in V_{\iota(h),h}$ **do**
8: eliminate all the nodes following w' from the game tree and consider w' as terminal nodes with utility $u_i(w') = \sum_{a \in \rho(w')} \sigma_{\iota(w')}(a) \cdot u_i(\chi(w',a))$ for each agent i
9: **return** sequentially rational

Definition 4.1 (path(w_0, w)) *We define* path(w_0, w) *as the sequence of pairs* (w', a) *with* $a \in \rho(w')$ *connecting the root node* w_0 *of the game tree to decision node* w.

Now, we review the results of Kreps and Wilson.

Definition 4.2 (B-labeling [41]) *A b-labeling for an assessment* $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ *is a function* $\lambda : A \to \mathbb{N}$ *that assigns a label (expressed as a non-negative integer number) to all the actions* $a \in A$ *such that:*[1]

$$\lambda_a = 0 \iff \sigma_i(a) > 0 \qquad \forall a \in A_i, i \in N$$

$$\sum_{\substack{a:\exists w'', \\ (w'',a) \in \text{path}(w_0,w)}} \lambda_a = \arg \min_{w' \in V_{i,h}} \sum_{\substack{a:\exists w'', \\ (w'',a) \in \text{path}(w_0,w')}} \lambda_a \iff$$

$$\mu_i(w) > 0 \qquad \forall w \in V_{i,h}, h \in H_i, i \in N$$

In words, a b-labeling assigns zero to the labels of actions played with strictly positive probability and requires that, if and only if the belief

[1] Notice that $\sum_{a:\exists w'', (w'',a) \in \text{path}(w_0,w)}$ is summing over all the actions a leading to node w from root w_0.

over a node is strictly positive, then the sum of the labels of the actions leading to such node from the root node is the smallest among the sums of the labels of the actions leading to all the other nodes of the same information set.

Theorem 4.1 [41] *Let*

$$\overline{\sigma}_i(a,\epsilon) = \begin{cases} c(\epsilon,h)\cdot\sigma_i(a)\cdot\epsilon^{\lambda_a} & \text{if } \sigma_i(a) > 0 \\ c(\epsilon,h)\cdot\epsilon^{\lambda_a} & \text{otherwise} \end{cases}$$

where a is an action played by $\iota(h)$ at information set h, and $c(\epsilon,h)$ is the appropriate normalizing constant; $\boldsymbol{\mu}$ is consistent w.r.t. $\boldsymbol{\sigma}$ if and only if $\overline{\boldsymbol{\sigma}}(\epsilon)$ is well defined (i.e., a b-labeling exists) and $\boldsymbol{\mu}(\epsilon)$, derived from $\overline{\boldsymbol{\sigma}}(\epsilon)$ by Bayes rule, converges to $\boldsymbol{\mu}$ as ϵ goes to zero.

It is worth remarking that an assessment $(\boldsymbol{\mu},\boldsymbol{\sigma})$ may admit multiple b-labelings. In this case, all the b-labelings lead to the same result: if, with a given b-labeling, $\boldsymbol{\mu}(\epsilon)$ derived from $\overline{\boldsymbol{\sigma}}(\epsilon)$ by Bayes rule converges to $\boldsymbol{\mu}$ as $\epsilon \to 0$, then the same happens with all the other b-labelings. Therefore, it is sufficient to search for a (generic) b-labeling.

Starting from the results of Kreps and Wilson, we can show the following.

Proposition 4.2 *Given an assessment $(\boldsymbol{\mu},\boldsymbol{\sigma})$ and a b-labeling, there exists an efficient algorithm verifying that $\boldsymbol{\mu}(\epsilon)$, derived from $\overline{\boldsymbol{\sigma}}(\epsilon)$ by Bayes rule, converges to $\boldsymbol{\mu}$ as ϵ goes to zero.*

Proof. Beliefs can be derived from fully mixed strategies by Bayes rule as follows:

$$\mu_i(w,\epsilon) = \frac{\prod_{(w',a)\in\mathsf{path}(w_0,w)} \overline{\sigma}_{\iota(w')}(a,\epsilon)}{\sum_{w''\in V_{\iota(h),h}: w\in V_{\iota(h),h}} \prod_{(w',a)\in\mathsf{path}(w_0,w'')} \overline{\sigma}_{\iota(w')}(a,\epsilon)}$$

The derivation of $\mu_i(w,\epsilon)$ requires a number of operations that is linear in the size of the game because both $|\mathsf{path}(w_0,w)|$ and $|V_{\iota(h),h}|$ are linear in the size of the game. We study the complexity of computing $\lim_{\epsilon\to 0}\mu_i(w,\epsilon)$. The normalization constants $c(\epsilon,h)$ are:

$$c(\epsilon,h) = \frac{1}{\sum_{a\in\rho(h):\sigma_{\iota(h)}(a)>0} \sigma_{\iota(h)}(a)\cdot\epsilon^{\lambda_a} + \sum_{a\in\rho(h):\sigma_{\iota(h)}(a)=0} \epsilon^{\lambda_a}}$$

Then, let:

$$
\begin{aligned}
c'(\epsilon, h) &= \frac{1}{c(\epsilon, h)} \\
\psi_w &= \prod_{(w',a)\in \mathsf{path}(w_0,w)} \epsilon^{\lambda_a} \\
\phi_w &= \prod_{(w',a)\in \mathsf{path}(w_0,w):\sigma_{\iota(w')}(a)>0} \sigma_{\iota(w')}(a) \\
C'_w &= \prod_{(w',a)\in \mathsf{path}(w_0,w), w'\in V_{\iota(h),h}} c'(\epsilon, h)
\end{aligned}
$$

Notice that $c'(\epsilon, h)$ is a polynomial in ϵ. We can write $\mu_i(w, \epsilon)$ as

$$
\mu_i(w,\epsilon) = \frac{\psi_w \cdot \phi_w \cdot \frac{1}{C_w}}{\sum_{w''\in V_{\iota(h),h}: w\in V_{\iota(h),h}} E_{w''} \cdot \phi_{w''} \cdot \frac{1}{C_{w''}}} = \frac{\psi_w \cdot \phi_w \cdot \prod_{w''\in V_{\iota(h),h}: w\in V_{\iota(h),h}, w''\neq w} C_{w''}}{\sum_{w''\in V_{\iota(h),h}: w\in V_{\iota(h),h}} \left(\psi_{w''} \cdot \phi_{w''} \cdot \prod_{w'''\in V_{\iota(h),h}: w\in V_{\iota(h),h}, w'''\neq w''} C_{w'''} \right)}
$$

In order to compute $\lim_{\epsilon\to 0} \mu_i(w, \epsilon)$, it is sufficient to isolate, for each $c'(\epsilon, h)$, the minimum degree of ϵ and its coefficient, discarding all the higher degrees of ϵ from $c'(\epsilon, h)$, and then calculate the multiplications and the sums to obtain the form $\frac{A_{\text{num}}\cdot\epsilon^{B_{\text{num}}}}{A_{\text{den}}\cdot\epsilon^{B_{\text{den}}}}$. This requires a number of operations that is linear in the size of the game. Then, the calculation of the limit is customary:

- if $B_{\text{num}} > B_{\text{den}}$, then $\lim_{\epsilon\to 0} \mu_{i,w}(\epsilon) = 0$;
- if $B_{\text{num}} = B_{\text{den}}$, then $\lim_{\epsilon\to 0} \mu_{i,w}(\epsilon) = \frac{A_{\text{num}}}{A_{\text{den}}}$.

Notice that the case in which $B_{\text{num}} < B_{\text{den}}$ is not possible. Hence, the proof of the proposition is complete. □

We focus on the problem of searching for a b-labeling. Initially, we provide an integer mathematical program to find a b-labeling.

Proposition 4.3 *Given* $(\boldsymbol{\mu}, \boldsymbol{\sigma})$*, the problem of searching for a b-labeling can be formulated as an integer linear mathematical program as follows:*

$$\min \sum_{a \in A} \lambda_a \tag{4.1}$$

$$\lambda_a = 0 \qquad \forall a \in A : \sigma_i(a) > 0, i \in N \tag{4.2}$$

$$\lambda_a - s_a = 1 \qquad \forall a \in A : \sigma_i(a) = 0, i \in N \tag{4.3}$$

$$\gamma_{w_0} = 0 \tag{4.4}$$

$$\gamma_{w'} + \lambda_a - \gamma_w = 0 \qquad \forall w, w' \in V, a \in A : w = \chi(w', a) \tag{4.5}$$

$$\gamma_w - \nu_h = 0 \qquad \forall h \in H_i, w \in V_{i,h} : \mu_i(w) > 0, i \in N \tag{4.6}$$

$$\gamma_w - \nu_h - t_w = 1 \qquad \forall h \in V_{i,h}, \mu_i(w) = 0, i \in N \tag{4.7}$$

$$\lambda_a \in \mathbb{N} \qquad \forall a \in A \tag{4.8}$$

$$s_a \geq 0 \qquad \forall a \in A \tag{4.9}$$

$$t_w \geq 0 \qquad \forall w \in V \tag{4.10}$$

where $\boldsymbol{\gamma}$ *and* $\boldsymbol{\nu}$ *are auxiliary variables, while* $\mathbf{s}$ *and* $\mathbf{t}$ *are slack variables.*

Proof. We show that the above program captures the definition of b-labeling. Constraints (4.2) force labels of actions played with positive probability to be equal to '0'; constraints (4.3) force labels of actions played with zero probability to be larger than or equal to '1'; constraint (4.4) assigns '0' to auxiliary variable γ_{w_0} associated with root node w_0 (auxiliary variable γ_w expresses the sum of the labels of all the actions leading to node w from the root node); constraints (4.5) assign the auxiliary variable γ_w associated with node w a value equal to the sum of the value $\gamma_{w'}$ of the parent node w' and the label of the action connecting w' to w; constraints (4.6) force the values of all the γ_ws associated with the nodes ws with $\mu_i(w) > 0$ belonging to the same information set to be same (i.e., ν_h); constraints (4.7) force the other nodes (those with $\mu_i(w) = 0$) to have a value γ strictly larger than the minimum value of the information set (i.e., ν_h); constraints (4.8)-(4.10) fix the domains of the variables (notice that, with these domains, all the variables have non-negative values). Objective function $\sum_{a \in A} \lambda_a$ is lower bounded given that labels are non-negative. Therefore, the minimization (4.1) always returns a b-labeling if this exists. □

Although solving an integer linear mathematical program is $\mathcal{NP}$-hard in the worst case, we can show that program (4.1)-(4.10) can be

solved in polynomial time. To prove it we exploit the property of *total unimodularity* provided in [27]. Given that a matrix Ξ is totally unimodular if and only if the transpose Ξ^T is totally unimodular [12], we can restate the definition of total unimodularity provided by Ghoulia-Houri as follows.

Definition 4.3 (Total unimodularity) *Matrix Ξ is totally unimodular if and only if for every subset Ξ' of columns of Ξ it is possible to find a partition of columns $\{\Xi'_1, \Xi'_2\}$ such that:*

$$\forall k \left(\sum_{j:\xi_{kj} \in \Xi'_1} \xi_{kj} - \sum_{j:\xi_{kj} \in \Xi'_2} \xi_{kj} \right) \in \{-1, 0, 1\} \tag{4.11}$$

where ξ_{kj} a generic entry of matrix Ξ.

In order to resort to total unimodularity for program (4.1)-(4.10), we need to formulate such program in standard form and isolate the matrix of constraints. More precisely, constraints (4.2)-(4.10) can be expressed as $M \cdot \mathbf{y} = \mathbf{b}$ with $\mathbf{y} \geq 0$ and $\boldsymbol{\lambda} \in \mathbb{N}^{|A|}$, where:

$$M = \begin{bmatrix} C & 0 & 0 & 0 & 0 \\ C' & 0 & 0 & -I & 0 \\ D & E & 0 & 0 & 0 \\ 0 & G & K & 0 & 0 \\ 0 & G' & K' & 0 & -I \end{bmatrix}, \mathbf{y} = \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\gamma} \\ \boldsymbol{\nu} \\ \mathbf{s} \\ \mathbf{t} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

such that:

- $C \cdot \boldsymbol{\lambda} = 0$ captures constraints (4.2),
- $C' \cdot \boldsymbol{\lambda} - I \cdot \mathbf{s} = \mathbf{1}$ captures constraints (4.3),
- $D \cdot \boldsymbol{\lambda} + E \cdot \boldsymbol{\gamma} = 0$ captures constraints (4.4) and (4.5),
- $G \cdot \boldsymbol{\gamma} + K \cdot \boldsymbol{\nu} = 0$ captures constraints (4.6), and
- $G' \cdot \boldsymbol{\gamma} + K' \cdot \boldsymbol{\nu} - I \cdot \mathbf{t} = \mathbf{1}$ captures constraints (4.7).

We can show that M is totally unimodular.

Proposition 4.4 *Matrix M is totally unimodular.*

Proof. Initially, we remark that all the entries of M belongs to $\{-1, 0, 1\}$ and that the submatrices of M have the following properties:

- C, C', G, and G' have one '1' per row and zero or one '1' per column;
- D is composed of a row of '0's and identity matrix I;
- E has one '1' in the first row and one '1' and one '-1' in all the other rows;
- K and K' present one '-1' per row.

Let Λ_k the k-th block of rows of M (from the top to the bottom) and let Δ_j the j-th block of columns of M (from the left to the right), as shown below:

$$\begin{array}{cccccc} C & 0 & 0 & 0 & 0 & \Lambda_1 \\ C' & 0 & 0 & -I & 0 & \Lambda_2 \\ D & E & 0 & 0 & 0 & \Lambda_3 \\ 0 & G & K & 0 & 0 & \Lambda_4 \\ 0 & G' & K' & 0 & -I & \Lambda_5 \\ \hline \Delta_1 & \Delta_2 & \Delta_3 & \Delta_4 & \Delta_5 & \end{array}$$

Let M' a subset of columns of M and m_{kj} a generic element of M. We need to show that, for any M', we can find a partition of columns $\{M'_1, M'_2\}$ such that constraints (4.11) are satisfied.

At first, we notice that the total unimodularity of M is not conditioned by the blocks of columns Δ_4 and Δ_5.

Lemma 4.1 *If matrix $\{\Delta_1 \cup \Delta_2 \cup \Delta_3\}$ is totally unimodular, then M is totally unimodular.*

Proof. Assume that for every M' it is possible to find a partition of columns $\{M'_1, M'_2\}$ such that

$$\forall k \left(\sum_{\substack{j: m_{kj} \in M'_1 \cap \\ \{\Delta_1 \cup \Delta_2 \cup \Delta_3\}}} m_{kj} - \sum_{\substack{j: m_{kj} \in M'_2 \cap \\ \{\Delta_1 \cup \Delta_2 \cup \Delta_3\}}} m_{kj} \right) \in \{-1, 0, 1\} \qquad (4.12)$$

In words: we are requiring that constraints (4.11) are satisfied under the additional constraint $m_{kj} \in M' \cap \{\Delta_1 \cup \Delta_2 \cup \Delta_3\}$. Now, we prove the Lemma, providing an iterative procedure to assign each column of $M' \cap \{\Delta_4 \cup \Delta_5\}$ to M_1 or M_2. Initially, we observe that

each column of $\{\Delta_4 \cup \Delta_5\}$ contains one '−1', while all the other entries are '0s', and that each row contains at most one '−1'. Therefore, each column can be assigned to M_1 or M_2 independently from the assignment of the others. Thus, take a column $\overline{r}$ of $M' \cap \{\Delta_4 \cup \Delta_5\}$ and call $\overline{k}$ the row in which there is '−1'. Assign column $\overline{k}$ to M'_1 or M'_2 as follows:

- if the left hand of constraints (4.11) for $k = \overline{k}$ is equal to '−1', then assign column $\overline{r}$ to M'_2, in this way $\sum_{j:m_{\overline{k}j} \in M'_1} m_{\overline{k}j} - \sum_{j:m_{\overline{k}j} \in M'_2} m_{\overline{k}j} = 0$,
- if the left hand of constraints (4.11) for $k = \overline{k}$ is equal to '1', then assign column $\overline{r}$ to M'_1, in this way $\sum_{j:m_{\overline{k}j} \in M'_1} m_{\overline{k}j} - \sum_{j:m_{\overline{k}j} \in M'_2} m_{\overline{k}j} = 0$,
- if the left hand of constraints (4.11) for $k = \overline{k}$ is equal to '0', then assign column $\overline{r}$ to M'_1 or M'_2 indifferently, in this way $\sum_{j:m_{\overline{k}j} \in M_1} m_{\overline{k}j} - \sum_{j:m_{\overline{k}j} \in M'_2} m_{\overline{k}j} \in \{-1, 1\}$.

Since all the other rows k of $\overline{r}$ contain '0s', $\sum_{j:m_{kj} \in M'_1} m_{kj} - \sum_{j:m_{kj} \in M'_2} m_{kj}$ is equal to $\sum_{j,m_{kj} \in M'_1 \cap \{\Delta_1 \cup \Delta_2 \cup \Delta_3\}} m_{kj} - \sum_{j,m_{kj} \in M'_2 \cap \{\Delta_1 \cup \Delta_2 \cup \Delta_3\}} m_{kj}$ that, by above assumption, belongs to $\{-1, 0, 1\}$. This proves that, if constraints (4.11) are satisfied for the first three blocks of columns, then such constraints can be satisfied also for all the blocks of columns with an opportune assignment of the columns of the last two blocks. □

By Lemma 4.1, M is totally unimodular if matrix $\{\Delta_1 \cup \Delta_2 \cup \Delta_3\}$ is totally unimodular. Thus, from here on, we study only the total unimodularity of $\{\Delta_1 \cup \Delta_2 \cup \Delta_3\}$.

Lemma 4.2 *Matrix* $\{\Delta_1 \cup \Delta_2 \cup \Delta_3\}$ *is totally unimodular.*

Proof. We build M'_1 and M'_2 as follows. Assign all the columns of $M' \cap \{\Delta_2 \cup \Delta_3\}$ to M'_1, while the columns of $M' \cap \Delta_1$ will be assigned to M'_1 or M'_2 to satisfy the total unimodularity condition as described below. Consider the rows belonging to Λ_4 and Λ_5 and sum the entries on these rows in M'_1 independently of whether the columns of $M' \cap \Delta_1$ are assigned to M'_1 or M'_2: given that all the columns of $M' \cap \{\Delta_2 \cup \Delta_3\}$ are assigned to M'_1 and all the entries of columns Δ_1 are zeros, the sum belongs to $\{-1, 0, 1\}$ (we recall, as

discussed above, that G and G' have one '1' per row, while K and K' have one '-1' per row). Thus, the rows belonging to Λ_4 and Λ_5 satisfy constraints (4.11) independently of whether the columns of $M' \cap \Delta_1$ are assigned to M'_1 or M'_2. Consider the columns of M'_1 belonging to Δ_2: the sum of the entries of the rows belonging to Λ_3 can be $\{-1, 0, 1\}$ (we recall, as discussed above, that E has one '1' in the first row and one '1' and one '-1' in all the other rows). It can be easily seen that, D having no more than one '1' per column and per row, we can always assign the columns of Δ_1 to M'_1 or M'_2 to make that constraints (4.11) are satisfied on the rows belonging to Λ_3. Finally, we observe that constraints (4.11) are always satisfied on the rows belonging to Λ_1 and Λ_2 given that C and C' have one '1' per row. Therefore, matrix $\{\Delta_1 \cup \Delta_2 \cup \Delta_3\}$ is totally unimodular. □

Since, by Lemma 4.1, if $\{\Delta_1 \cup \Delta_2 \cup \Delta_3\}$ is totally unimodular, then M is totally unimodular and, by Lemma 4.2, $\{\Delta_1 \cup \Delta_2 \cup \Delta_3\}$ is totally unimodular, we have that M is totally unimodular. This completes the proof of the proposition. □

Now, we are in the position to prove what follows.

Proposition 4.5 *Given* $(\boldsymbol{\mu}, \boldsymbol{\sigma})$*, there exists an efficient algorithm verifying that* $\boldsymbol{\mu}$ *is consistent w.r.t.* $\boldsymbol{\sigma}$*.*

Proof. Consistency can be verified by means of Algorithm 2. By Proposition 4.3, a b-labeling can be found by solving the linear integer program (4.1)-(4.10). Since the matrix of constraints M of this program is totally unimodular, as shown by Proposition 4.4, and the vector of constants $\mathbf{b}$ is integer, all the vertices of the polytope $\{\mathbf{z}|M\mathbf{z} = \mathbf{b}\}$ with $\mathbf{z}$ real-value variables are integers [27]. Thus, we can find a solution of program (4.1)-(4.10) by searching for a basic solution of its continuous relaxation by means of linear mathematical programming techniques. Since a linear mathematical program can be solved in polynomial time, program (4.1)-(4.10) can be solved in polynomial time. If the program does not admit any b-labeling, then, by Theorem 4.1, $\boldsymbol{\mu}$ is not consistent. If the program admits a b-labeling $\boldsymbol{\lambda}$, then, by Theorem 4.1, $\boldsymbol{\mu}$ is consistent if and only if $\overline{\boldsymbol{\sigma}}(\epsilon)$ - defined in Theorem 4.1 - is such that $\boldsymbol{\mu}(\epsilon)$, derived from $\overline{\boldsymbol{\sigma}}(\epsilon)$ by Bayes rule, converges to $\boldsymbol{\mu}$ as ϵ goes to zero. Verifying such a convergence requires polynomial time in the size of the game, as shown by Proposition 4.2. This completes the proof of the proposition. □

Algorithm 2 verifyConsistency($\boldsymbol{\mu}, \boldsymbol{\sigma}$)

1: solve the continuous relaxation of program (4.1)-(4.10) by linear mathematical programming
2: **if** the program does not admit any solution $\boldsymbol{\lambda}$ **then**
3: **return** non-consistent
4: derive $\overline{\boldsymbol{\sigma}}(\epsilon)$ from $\boldsymbol{\lambda}$
5: derive $\boldsymbol{\mu}(\epsilon)$ from $\overline{\boldsymbol{\sigma}}(\epsilon)$ by Bayes rule
6: **if** $\lim_{\epsilon \to 0} \boldsymbol{\mu}(\epsilon) \neq \boldsymbol{\mu}$ **then**
7: **return** non-consistent
8: **return** consistent

We can leverage on the above results to prove the following theorem.

Theorem 4.2 *There exists an efficient algorithm certifying that an assessment* $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ *given as input is an SE.*

Proof. Algorithm 3 certificates in polynomial time that $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ is an SE. Indeed, by Proposition 4.1, we can certify whether $\boldsymbol{\sigma}$ is sequentially rational w.r.t. $\boldsymbol{\mu}$ in polynomial time and, by Proposition 4.5, we can certify whether $\boldsymbol{\mu}$ is consistent w.r.t. $\boldsymbol{\sigma}$ in polynomial time. This concludes the proof of the theorem. □

Algorithm 3 SEcertifying($\boldsymbol{\mu}, \boldsymbol{\sigma}$)

1: **if** verifySequentialRationality($\boldsymbol{\mu}, \boldsymbol{\sigma}$) returns non-sequentially rational or verifyConsistency($\boldsymbol{\mu}, \boldsymbol{\sigma}$) returns non-consistent **then**
2: **return** non-SE
3: **return** SE

The above theorem shows that there is an efficient algorithm for the verification problem of SE with any number of agents, and therefore such a problem is easier than the verification problems for QPE, EFPE, and NFPrE that are instead $\mathcal{NP}$-hard with three or more agents [32].

We provide two examples to which we apply Algorithm 3.

Example 4.1 *Consider the game depicted in Fig. 4.1, where* $\boldsymbol{\sigma} = (\sigma_1(\mathsf{L}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_1(\mathsf{R}_3) = 1, \sigma_2(\mathsf{l}_1) = 1, \sigma_2(\mathsf{r}_2) = 1, \sigma_2(\mathsf{l}_3) = 1)$,

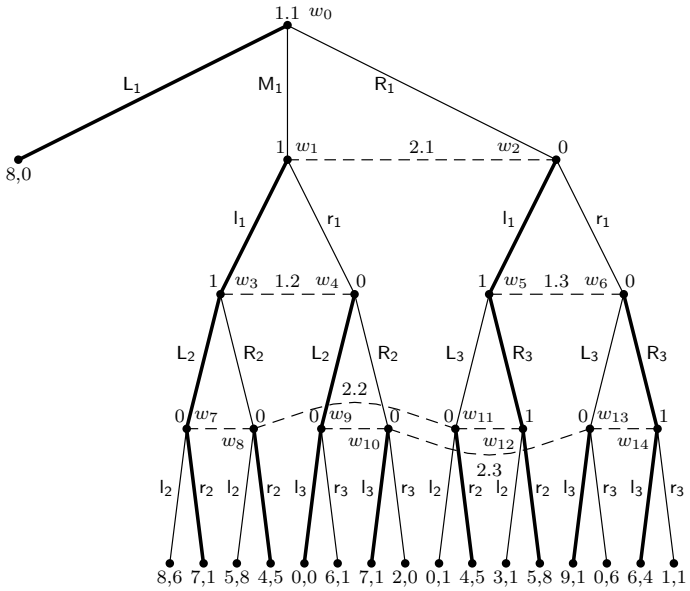


Figure 4.1: Example of assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\sigma}$ is sequentially rational, but $\boldsymbol{\mu}$ is not consistent ($\boldsymbol{\sigma}$ is represented by using bold lines to denote actions played with positive probability and $\boldsymbol{\mu}$ is represented reporting the beliefs close to the nodes of each information set).

while beliefs $\boldsymbol{\mu}$ are reported in the figure aside the corresponding nodes. Below we instantiate constraints (4.1)-(4.10).

$$\lambda_{\mathsf{L}_1} = \lambda_{\mathsf{L}_2} = \lambda_{\mathsf{R}_3} = \lambda_{\mathsf{l}_1} = \lambda_{\mathsf{r}_2} = \lambda_{\mathsf{l}_3} = 0$$
$$\lambda_{\mathsf{M}_1} - s_{\mathsf{M}_1} = \lambda_{\mathsf{R}_1} - s_{\mathsf{R}_1} = \lambda_{\mathsf{R}_2} - s_{\mathsf{R}_2} = \lambda_{\mathsf{R}_3} - s_{\mathsf{R}_3} = 1$$
$$\lambda_{\mathsf{r}_1} - s_{\mathsf{r}_1} = \lambda_{\mathsf{l}_2} - s_{\mathsf{l}_2} = \lambda_{\mathsf{r}_3} - s_{\mathsf{r}_3} = 1$$
$$\gamma_{w_0} = \gamma_{w_0} + \lambda_{\mathsf{M}_1} - \gamma_{w_1} = \gamma_{w_0} + \lambda_{\mathsf{R}_1} - \gamma_{w_2} = 0$$
$$\gamma_{w_1} + \lambda_{\mathsf{l}_1} - \gamma_{w_3} = \gamma_{w_1} + \lambda_{\mathsf{r}_1} - \gamma_{w_4} =$$
$$\gamma_{w_2} + \lambda_{\mathsf{l}_2} - \gamma_{w_5} = \gamma_{w_2} + \lambda_{\mathsf{r}_2} - \gamma_{w_6} = 0$$
$$\gamma_{w_3} + \lambda_{\mathsf{L}_2} - \gamma_{w_7} = \gamma_{w_3} + \lambda_{\mathsf{R}_2} - \gamma_{w_8} =$$
$$\gamma_{w_4} + \lambda_{\mathsf{L}_2} - \gamma_{w_9} = \gamma_{w_4} + \lambda_{\mathsf{R}_2} - \gamma_{w_{10}} = 0$$

$$\gamma_{w_5} + \lambda_{\mathsf{L}_3} - \gamma_{w_{11}} = \gamma_{w_5} + \lambda_{\mathsf{R}_3} - \gamma_{w_{12}} =$$
$$\gamma_{w_6} + \lambda_{\mathsf{L}_3} - \gamma_{w_{13}} = \gamma_{w_6} + \lambda_{\mathsf{R}_3} - \gamma_{w_{14}} = 0$$
$$\gamma_{w_3} - \nu_{1.2} = \gamma_{w_5} - \nu_{1.3} = \gamma_{w_1} - \nu_{2.1} = \gamma_{w_{12}} - \nu_{2.2} = \gamma_{w_{14}} - \nu_{2.3} = 0$$
$$\gamma_{w_4} - \nu_{1.2} - t_{w_4} = \gamma_{w_6} - \nu_{1.3} - t_{w_6} = 1$$
$$\gamma_{w_2} - \nu_{2.1} - t_{w_2} = \gamma_{w_7} - \nu_{2.2} - t_{w_7} =$$
$$\gamma_{w_8} - \nu_{2.2} - t_{w_8} = \gamma_{w_{11}} - \nu_{2.2} - t_{w_{11}} = 1$$
$$\gamma_{w_9} - \nu_{2.3} - t_{w_9} = \gamma_{w_{10}} - \nu_{2.3} - t_{w_{10}} = \gamma_{w_{13}} - \nu_{2.3} - t_{w_{13}} = 1$$

Now we show that there is a contradiction:

(a) from $\gamma_{w_0} + \lambda_{\mathsf{M}_1} - \gamma_{w_1} = 0$ *and* $\gamma_{w_0} = 0$, *we have* $\gamma_{w_1} = \lambda_{\mathsf{M}_1}$,

(b) from $\gamma_{w_0} + \lambda_{\mathsf{R}_1} - \gamma_{w_2} = 0$ *and* $\gamma_{w_0} = 0$, *we have* $\gamma_{w_2} = \lambda_{\mathsf{R}_1}$,

(c) from $\gamma_{w_1} - \nu_{2.1} = 0$ *and* $\gamma_{w_1} = \lambda_{\mathsf{M}_1}$ *by (a), we have* $\nu_{2.1} = \lambda_{\mathsf{M}_1}$,

(d) from $\gamma_{w_2} - \nu_{2.1} \geq 1$, $\gamma_{w_2} = \lambda_{\mathsf{R}_1}$ *by (b) and* $\nu_{2.1} = \lambda_{\mathsf{M}_1}$ *by (c), we have* $\lambda_{\mathsf{R}_1} \geq 1 + \lambda_{\mathsf{M}_1}$,

(e) from $\lambda_{L_2} = \lambda_{\mathsf{l}_1} = 0$, $\gamma_{w_1} + \lambda_{\mathsf{l}_1} - \gamma_{w_3} = \gamma_{w_3} + \lambda_{\mathsf{L}_2} - \gamma_{w_7} = 0$ *and* $\gamma_{w_1} = \lambda_{\mathsf{M}_1}$ *by (a), we have* $\gamma_{w_7} = \lambda_{\mathsf{M}_1}$,

(f) from $\lambda_{\mathsf{R}_3} = \lambda_{\mathsf{l}_1} = 0$, $\gamma_{w_2} + \lambda_{\mathsf{l}_2} - \gamma_{w_5} = \gamma_{w_5} + \lambda_{\mathsf{R}_3} - \gamma_{w_{12}} = 0$, $\gamma_{w_{12}} - \nu_{2.2} = 0$ *and* $\gamma_{w_2} = \lambda_{\mathsf{R}_1}$ *by (b), we have* $\lambda_{\mathsf{R}_1} = \nu_{2.2}$,

(g) from $\gamma_{w_7} - \nu_{2.2} \geq 1$, $\gamma_{w_7} = \lambda_{\mathsf{M}_1}$ *by (e) and* $\lambda_{\mathsf{R}_1} = \nu_{2.2}$ *by (f), so we have* $\lambda_{\mathsf{M}_1} \geq 1 + \lambda_{\mathsf{R}_1}$,

(h) (d) and (g) are in contradiction.

Thus, the above set of constraints does not admit any feasible assignment and therefore the assessment given as input is not an SE.

Example 4.2 *Consider the game depicted in Fig. 4.2, where* $\boldsymbol{\sigma} = (\sigma_1(\mathsf{M}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_1(\mathsf{R}_3) = 1, \sigma_2(\mathsf{l}_1) = 1, \sigma_2(\mathsf{l}_2) = 1, \sigma_2(\mathsf{r}_3) = 1)$, *while beliefs* $\boldsymbol{\mu}$ *are reported in the figure aside the corresponding nodes. Below we instantiate constraints (4.1)-(4.10).*

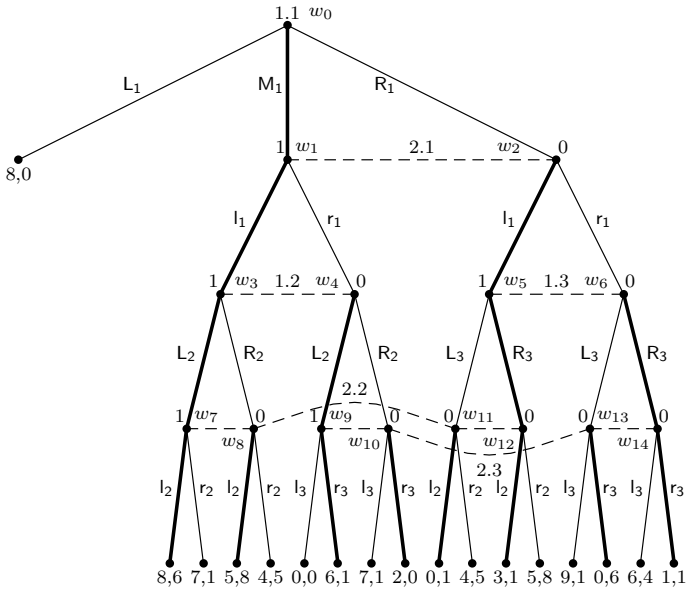


Figure 4.2: Example of assessment $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\sigma}$ is sequentially rational and $\boldsymbol{\mu}$ is consistent ($\boldsymbol{\sigma}$ is represented by using bold lines to denote actions played with positive probability and $\boldsymbol{\mu}$ is represented reporting the beliefs close to the nodes of each information set).

$$\lambda_{\mathsf{M}_1} = \lambda_{\mathsf{L}_2} = \lambda_{\mathsf{R}_3} = \lambda_{\mathsf{l}_1} = \lambda_{\mathsf{l}_2} = \lambda_{\mathsf{r}_3} = 0$$
$$\lambda_{\mathsf{L}_1} - s_{\mathsf{L}_1} = \lambda_{\mathsf{R}_1} - s_{\mathsf{R}_1} = \lambda_{\mathsf{R}_2} - s_{\mathsf{R}_2} = \lambda_{\mathsf{L}_3} - s_{\mathsf{L}_3} = 1$$
$$\lambda_{\mathsf{r}_1} - s_{\mathsf{r}_1} = \lambda_{\mathsf{r}_2} - s_{\mathsf{r}_2} = \lambda_{\mathsf{l}_3} - s_{\mathsf{l}_3} = 1$$
$$\gamma_{w_0} = \gamma_{w_0} + \lambda_{\mathsf{M}_1} - \gamma_{w_1} = \gamma_{w_0} + \lambda_{\mathsf{R}_1} - \gamma_{w_2} = 0$$
$$\gamma_{w_1} + \lambda_{\mathsf{l}_1} - \gamma_{w_3} = \gamma_{w_1} + \lambda_{\mathsf{r}_1} - \gamma_{w_4} =$$
$$\gamma_{w_2} + \lambda_{\mathsf{l}_2} - \gamma_{w_5} = \gamma_{w_2} + \lambda_{\mathsf{r}_2} - \gamma_{w_6} = 0$$
$$\gamma_{w_3} + \lambda_{\mathsf{L}_2} - \gamma_{w_7} = \gamma_{w_3} + \lambda_{\mathsf{R}_2} - \gamma_{w_8} =$$
$$\gamma_{w_4} + \lambda_{\mathsf{L}_2} - \gamma_{w_9} = \gamma_{w_4} + \lambda_{\mathsf{R}_2} - \gamma_{w_{10}} = 0$$
$$\gamma_{w_5} + \lambda_{\mathsf{L}_3} - \gamma_{w_{11}} = \gamma_{w_5} + \lambda_{\mathsf{R}_3} - \gamma_{w_{12}} =$$
$$\gamma_{w_6} + \lambda_{\mathsf{L}_3} - \gamma_{w_{13}} = \gamma_{w_6} + \lambda_{\mathsf{R}_3} - \gamma_{w_{14}} = 0$$
$$\gamma_{w_3} - \nu_{1.2} = \gamma_{w_5} - \nu_{1.3} = \gamma_{w_1} - \nu_{2.1} = \gamma_{w_7} - \nu_{2.2} = \gamma_{w_9} - \nu_{2.3} = 0$$
$$\gamma_{w_4} - \nu_{1.2} - t_{w_4} = \gamma_{w_6} - \nu_{1.3} - t_{w_6} = 1$$

$$\gamma_{w_2} - \nu_{2.1} - t_{w_2} = \gamma_{w_8} - \nu_{2.2} - t_{w_8} = \gamma_{w_{11}} - \nu_{2.2} - t_{w_{11}} = 1$$
$$\gamma_{w_{12}} - \nu_{2.2} - t_{w_{12}} = \gamma_{w_{14}} - \nu_{2.3} - t_{w_{14}} =$$
$$\gamma_{w_{10}} - \nu_{2.3} - t_{w_{10}} = \gamma_{w_{13}} - \nu_{2.3} - t_{w_{13}} = 1$$

A b-labeling is: $\lambda_a = 1$ *for all* $a \in A_i$ *with* $\sigma_i(a) = 0$. *Therefore, the assessment given as input is an SE.*

On the basis of the previous result, we can state the following corollary.

Corollary 4.1 *The problems of*

- *searching for an exact SE of a game with with two agents, and*
- *searching for a weakly approximate SE of a game with three or more agents*

are in $\mathcal{TFNP}$.

Proof. The membership of the two above problems to $\mathcal{TFNP}$ follows from:

- the functional problem is total since every game admits at least one SE,
- there exists an efficient algorithm for the verification problem of SE as shown by Theorem 4.2.

This completes the proof of the corollary. □

4.2 SE verification with two agents when only the strategy profile is given

We focus on the problem of deciding whether or not a strategy profile given in behavioral strategies is an SE with two-agent games. As discussed in Section 2, the SE concept relaxes the QPE concept, not requiring the agents' strategies to be optimal in presence of perturbations. In the following, we show that there exists an efficient algorithm for the verification problem of QPE with two agents and subsequently we show that a simple variation of the algorithm can be employed for the verification of SE with two agents.

In order to verify whether a strategy profile $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ is a QPE, we need to search for (if they exist):

- a fully mixed $\boldsymbol{\sigma}_1(\epsilon)$ such that $\lim_{\epsilon \to 0} \boldsymbol{\sigma}_1(\epsilon) = \boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ is a best response to $\boldsymbol{\sigma}_1(\epsilon)$ for $\epsilon \leq \bar{\epsilon}$ for some $\bar{\epsilon}$,
- a fully mixed $\boldsymbol{\sigma}_2(\epsilon)$ such that $\lim_{\epsilon \to 0} \boldsymbol{\sigma}_2(\epsilon) = \boldsymbol{\sigma}_2$ and $\boldsymbol{\sigma}_1$ is a best response to $\boldsymbol{\sigma}_2(\epsilon)$ for $\epsilon \leq \bar{\epsilon}$ for some $\bar{\epsilon}$.

Instead of searching directly for perturbed behavioral strategies $\boldsymbol{\sigma}_i(\epsilon)$ that require non-linear best-response constraints (necessary in our treatment), we can work with perturbed sequence-form strategies $\mathbf{x}_i(\epsilon)$ - similarly to [48] - that require instead linear best-response constraints. More precisely, our goal is to search for a pair of fully mixed $\mathbf{x}_i(\epsilon)$ with the property that the fully mixed $\boldsymbol{\sigma}_i(\epsilon)$ derived from $\mathbf{x}_i(\epsilon)$ converges to the input $\boldsymbol{\sigma}_i$ as ϵ goes to zero.

Furthermore, although in principle the perturbation of $\mathbf{x}_i(\epsilon)$ can be any function of ϵ, we can safely limit our search to polynomial symbolic perturbations in ϵ and to deal with them lexicographically [8, 30]. We provide the details.

Definition 4.4 (Polynomially perturbed sequence-form strategy) *A sequence-form strategy $x_i(q, \epsilon)$ with polynomial symbolic perturbation in ϵ is defined as $x_i(q, \epsilon) = \sum_k x_i(q, \epsilon^k) \cdot \epsilon^k$ where $x_i(q, \epsilon^k)$ is the coefficient of ϵ^k.*

A sequence-form strategy $x_i(q, \epsilon)$ with symbolic polynomial perturbation can be conveniently represented as a vector whose elements are the coefficients $x_i(q, \epsilon^k)$ from $k = 0$ on as:

$$x_i(q, \epsilon) = \begin{bmatrix} x_i(q, \epsilon^0) & x_i(q, \epsilon^1) & x_i(q, \epsilon^2) & \dots \end{bmatrix}$$

The condition $x_i(q, \epsilon) \geq 0$ as ϵ goes to zero can be expressed by resorting to the relation $\geq_{\text{lex}}$, i.e. 'lexicographically larger than or equal to', defined as follows.

Definition 4.5 ($\geq_{\text{lex}}$) *Given a vector $\mathbf{y}$, we have $\mathbf{y} \geq_{\text{lex}} \mathbf{0}$ if the first (in lexicographic order) non-zero element of $\mathbf{y}$ is positive.*

Definition 4.6 ($>_{\text{lex}}$) *Given a vector $\mathbf{y}$, we have $\mathbf{y} >_{\text{lex}} \mathbf{0}$ if exists at least one positive element in $\mathbf{y}$ and the first (in lexicographic order) non-zero element of $\mathbf{y}$ is positive.*

We can state the condition for the positiveness of a strategy and the condition to have a fully mixed strategy as discussed in [48].

Proposition 4.6 (Perturbed strategy positiveness) *A sequence-form strategy $x_i(q, \epsilon)$ with symbolic polynomial perturbation is positive as ϵ goes to zero when $x_i(q, \epsilon) \geq_{\text{lex}} 0$.*

Proposition 4.7 (Fully mixed strategy) *A sequence-form strategy $\mathbf{x}_i(\epsilon)$ with symbolic polynomial perturbation is fully mixed as ϵ goes to zero when $x_i(q, \epsilon) >_{\text{lex}} 0$ for every $q \in Q$.*

Now we can formulate the conditions such that a given $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ is a QPE in terms of fully mixed $\mathbf{x}_1(\epsilon)$ and $\mathbf{x}_2(\epsilon)$.

Proposition 4.8 (QPE conditions in sequence form) *Given strategy profile $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ as input, $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ is a QPE if and only if, for every $i \in N$, the following program*

$$F_i \cdot \mathbf{x}_i(\epsilon) = \mathbf{f}_i \tag{4.13}$$

$$\mathbf{x}_i(\epsilon) >_{\text{lex}} \mathbf{0} \tag{4.14}$$

$$F_{-i}^T \cdot \mathbf{v}_{-i}(\epsilon) - U_{-i}^T \cdot \mathbf{x}_i(\epsilon) \geq_{\text{lex}} \mathbf{0} \tag{4.15}$$

$$(F_{-i}^T \cdot \mathbf{v}_{-i}(\epsilon) - U_{-i}^T \cdot \mathbf{x}_i(\epsilon))_q = 0 \qquad \forall q \in Q_{-i} : \sigma_{-i}(a(q)) > 0 \tag{4.16}$$

$$\lim_{\epsilon \to 0} \frac{x_i(q|a, \epsilon)}{x_i(q, \epsilon)} = \sigma_i(a) \qquad \forall a \in A_i : x_i(q, \epsilon) >_{\text{lex}} 0 \tag{4.17}$$

where $\mathbf{v}_i$ and $\mathbf{x}_i$ are variables, admits a solution.

Proof. The proof follows from the definitions of sequence form and QPE. However, we provide the details because we exploit them in the subsequent propositions. The above program captures the definition of QPE with two agents since:

- $\mathbf{x}_i(\epsilon)$ is a well-defined perturbed sequence-form strategy due to constraints (4.13) - these constraints apply the definition of sequence form (see Section 2.2);
- $\mathbf{x}_i(\epsilon)$ is fully mixed due to constraints (4.14) - these constraints apply the definition of strict lexico positiveness (see Proposition 4.7);

- $\boldsymbol{\sigma}_{-i}$ is a best response to $\mathbf{x}_i(\epsilon)$ even for $\epsilon > 0$ due to constraints (4.15) and constraints (4.16) - more precisely, constraints (4.15) are the dual best-response constraints derived as in [48], forcing the value associated with an information set to be at least the value given by the best action agent $-i$ can play at such information set, and constraints (4.15) force that sequences q are the best sequences agent $-i$ can play;

- $\mathbf{x}_i(\epsilon)$ and $\boldsymbol{\sigma}_i$ are the same strategy as ϵ goes to zero due to constraints (4.17) and constraints (4.14) - indeed, constraints (4.17) force the two strategies to be same as $\epsilon \to 0$ for every strictly positive $x_i(q)$ and constraints (4.14) force all the $x_i(q)$ to be strictly positive.

Hence, if program (4.13)-(4.17) admits a solution, then there exists a well-defined fully mixed $\mathbf{x}_i(\epsilon)$ from which we can derive a fully mixed $\boldsymbol{\sigma}_i(\epsilon)$ such that $\lim_{\epsilon \to 0} \boldsymbol{\sigma}_i(\epsilon) = \boldsymbol{\sigma}_i$ and $\boldsymbol{\sigma}_{-i}$ is best response to $\boldsymbol{\sigma}_i(\epsilon)$. Thus, if program (4.13)-(4.17) admits a solution for $i \in N$, $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ is a QPE. This completes the proof. □

We introduce a sufficient condition to have equivalence between perturbed strategies, whose proof is omitted being straightforward - we point the interested reader to [66] - ; we use this condition in the following.

Proposition 4.9 (Equivalent perturbed strategies) *Given two perturbed strategies $\mathbf{x}_i(\epsilon)$ and $\mathbf{x}'_i(\epsilon)$, a sufficient condition such that $\mathbf{x}_i(\epsilon)$ and $\mathbf{x}'_i(\epsilon)$ are equivalent is: for some k it holds that $x_i(q, \epsilon^k) = \alpha \cdot x'_i(q, \epsilon^k)$ with $\alpha > 0$ for every $q \in Q$.*

Obviously, given two equivalent perturbed strategies $\mathbf{x}_i(\epsilon)$ and $\mathbf{x}'_i(\epsilon)$, if $\mathbf{x}_i(\epsilon)$ satisfies program (4.13)-(4.17), then also $\mathbf{x}'_i(\epsilon)$ does and *vice versa*.

We can show that there exists an efficient algorithm for program (4.13)-(4.17).

Proposition 4.10 *Given $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ as input, there is an efficient algorithm solving program (4.13)-(4.17).*

Proof. We show that Algorithm 6 solves program (4.13)-(4.17) in polynomial time. The basic idea of the algorithm is the following:

- we initialize $x_i(q, \epsilon^k) = 0$ for every q and $k \in \{0, \ldots, |Q_i|\}$,
- the algorithm is iterative,
- at each iteration k the algorithm attempts to increase the number of strictly lexico positive strategies w.r.t. iteration $k-1$,
- at each iteration k the values of $\mathbf{x}_i(\epsilon^k)$ satisfy constraints (4.13), (4.15), (4.16), (4.17) and satisfy $\mathbf{x}_i(\epsilon) \geq_{\text{lex}} 0$.

Thus, if at some iteration a fully mixed $\mathbf{x}_i(\epsilon)$ is found, such a $\mathbf{x}_i(\epsilon)$ satisfies constraints (4.13)-(4.17).

Algorithm 4 findPerturbedStrategy($\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, i$)

```
1: x_i(q, ε^k) = 0 for every q and k ∈ {0, ..., |Q_i|}
2: verify constraints (4.18)-(4.19) for agent i
3: if constraints are not satisfied then
4:     return non-existence
5: while x_i(ε) is not strictly lexico positive do
6:     solve program (4.20)-(4.26) for agent i
7:     if objective function is 0 then
8:         return non-existence
9: return x_i(ε)
```

Iteration $k = 0$. We derive $\mathbf{x}_i(\epsilon^0)$ from $\boldsymbol{\sigma}_i$ as $x_i(q, \epsilon^0) = \prod_{a \in q} \sigma_i(a)$ and we verify whether the following constraints hold:

$$F_{-i}^T \cdot \mathbf{v}_{-i}(\epsilon^0) - U_{-i}^T \cdot \mathbf{x}_i(\epsilon^0) \geq \mathbf{0} \tag{4.18}$$

$$(F_{-i}^T \cdot \mathbf{v}_{-i}(\epsilon^0) - U_{-i}^T \cdot \mathbf{x}_i(\epsilon^0))_q = 0 \quad \forall q \in Q_{-i} : \sigma_{-i}(a(q)) > 0 \tag{4.19}$$

where constraints (4.18) and (4.19) correspond to constraints (4.15) and (4.16), respectively, in absence of perturbations. If the above constraints are satisfied by $\mathbf{x}_i(\epsilon^0)$, then $\boldsymbol{\sigma}_{-i}$ is a best response to $\boldsymbol{\sigma}_i$ and the algorithm goes to the next iteration. Furthermore, if the above constraints are satisfied by $\mathbf{x}_i(\epsilon^0)$, then $\mathbf{x}_i(\epsilon^0)$ satisfies constraints (4.13), (4.15), (4.16), (4.17) and satisfy $\mathbf{x}_i(\epsilon^0) \geq_{\text{lex}} 0$. Otherwise, the algorithm terminates returning non-existence, $\boldsymbol{\sigma}$ not being neither an NE.

Iteration $k \geq 1$. From $k = 1$ on, the algorithm solves the following linear mathematical program until $\mathbf{x}_i(\epsilon)$ is not strictly lexico positive:

$$\max \quad \sum_{\forall k' < k, x_i(q,\epsilon^{k'})=0} x_i(q, \epsilon^k) \tag{4.20}$$

$$F_i \cdot \mathbf{x}_i(\epsilon^k) = \mathbf{0} \tag{4.21}$$

$$x_i(q, \epsilon^k) \geq 0 \quad \forall q \in Q_i : x_i(q, \epsilon^{k'}) = 0, \ \forall k' < k \tag{4.22}$$

$$x_i(q, \epsilon^k) \leq 1 \quad \forall q \in Q_i \tag{4.23}$$

$$x_i(q|a, \epsilon^k) = \sigma_i(a) \cdot z_h \quad \forall h \in H_i, q \in Q_i : a \in \rho(h), \ \forall a' \in \rho(h), x_i(q|a', \epsilon^{k'}) = 0, \ \forall k' < k \tag{4.24}$$

$$z_h \geq 0 \quad \forall h \in H_i \tag{4.25}$$

$$(F_{-i}^T \cdot \mathbf{v}_{-i}(\epsilon^k) - U_{-i}^T \cdot \mathbf{x}_i(\epsilon^k))_q \geq 0 \quad \forall q \in Q_{-i} : \forall k' < k, \ (F_{-i}^T \cdot \mathbf{v}_{-i}(\epsilon^{k'}) - U_{-i}^T \cdot \mathbf{x}_i(\epsilon^{k'}))_q = 0 \tag{4.26}$$

$$(F_{-i}^T \cdot \mathbf{v}_{-i}(\epsilon^k) - U_{-i}^T \cdot \mathbf{x}_i(\epsilon^k))_q = 0 \quad \forall q \in Q_{-i} : \sigma_{-i}(a(q)) > 0 \tag{4.27}$$

where

- constraints (4.21) assure the perturbation of degree k to be well defined according to the definition of sequence form, forcing $\mathbf{x}_i(\epsilon)$ to satisfy constraints (4.13);
- constraints (4.22) assure that $\mathbf{x}_i(\epsilon) \geq_{\text{lex}} \mathbf{0}$, forcing $x_i(q)\epsilon^k) \geq 0$ for all the strategies $x_i(q, \epsilon)$ that are not strictly lexico positive yet (i.e. for $k' < k$) - instead, the constraints $x_i(q, \epsilon^k) \geq 0$ are not applied to all the strategies that are already (i.e. for $k' < k$) strictly lexico positive;
- constraints (4.23) pose - without loss of generality as shown by Proposition 4.9 - an upper bound of 1 over the coefficients of ϵ^k;

- constraints (4.24) and (4.25) force $\mathbf{x}_i(\epsilon)$ to satisfy constraints (4.17); this is accomplished forcing that if the strategy $x_i(q|a,\epsilon)$ becomes strictly lexico positive then all the strategies $x_i(q|a'\epsilon)$ where a and a' are available to the same information set h must assume values that are consistent to $\boldsymbol{\sigma}_i$. More precisely, if the strategy $x_i(q|a,\epsilon)$ becomes strictly lexico positive, it means that z_h with $a \in \rho(h)$ is strictly positive. Then, the sum of all the $x_i(q|a,\epsilon^k)$ with $a \in \rho(h)$ is equal to z_h (given that $\sum_{a\in\rho(h)} \sigma_i(a) = 1$) that is equal, by definition of sequence form, to $x_i(q,\epsilon^k)$ and therefore $\lim_{\epsilon\to 0} \frac{x_i(q|a,\epsilon^k)}{x_i(q,\epsilon^k)} = \sigma_i(a)$;

- constraints (4.26) and (4.27) force $\mathbf{x}_i(\epsilon)$ to satisfy constraints (4.15) and (4.16), respectively, and therefore that sequences q with $\sigma_{-i}(a(q)) > 0$ are the best sequences agent $-i$ can play at each information set; constraints (4.26) are applied only to sequences q that provide the maximum utility for all the previous $k' < k$, because for the other sequences q the values $v_{-i,h}$ are already (i.e. for $k' < k$) lexicographically strictly larger than the expected utility provided by q and therefore, for these sequences, constraints (4.15) are satisfied independently of the perturbation of degree k;

- objective function (4.20) aims at maximizing the sum of the coefficients $x_i(q,\epsilon^k)$ such that $x_i(q,\epsilon^{k'}) = 0$ for all $k' < k$. In this way, if it is possible to make some $x_i(q,\epsilon)$ strictly lexico positive, it will be done.

Notice that the solution $x_i(q,\epsilon^k) = 0$ for every q is always a feasible solution. If at the optimal solution of program (4.20)-(4.27) the objective function has a value of zero, then the algorithm terminates returning non-existence. The algorithm goes to the next iteration otherwise.

We study the properties of the algorithm. Initially, we state the following lemma, whose proof is omitted being straightforward.

Lemma 4.3 *If Algorithm 6 terminates returning a solution* $\mathbf{x}_i(\epsilon)$*, then* $\mathbf{x}_i(\epsilon)$ *is strictly lexico positive.*

Then, we state the following lemma.

Lemma 4.4 *If Algorithm 6 terminates with* non-existence*, then there is no strictly lexico-positive strategy* $\mathbf{x}_i(\epsilon)$ *such that* $\boldsymbol{\sigma}_{-i}$ *is best response to* $\mathbf{x}_i(\epsilon)$.

Proof. If the algorithm terminates at $k = 0$, the proof is straightforward. We prove the lemma for the case in which the algorithm terminates at $k \geq 1$ into two steps:

- *Step 1: we prove that, given* $\mathbf{x}_i(\epsilon)$ *returned at iteration* $k-1$*, if the algorithm terminates at iteration* k *returning* non-existence*, then there is no perturbation with degrees* $k' \geq k$ *such that, once applied to* $\mathbf{x}_i(\epsilon)$*,* $\mathbf{x}_i(\epsilon)$ *satisfies constraints (4.13)-(4.17).* If the objective function is zero, then it is not possible to introduce any perturbation in $\mathbf{x}_i(\epsilon)$ of degree k without violating the best-response constraints (4.26)-(4.27) of agent $-i$ and/or constraints (4.24) and (4.25). Therefore $\mathbf{x}_i(\epsilon)$ cannot be made strictly lexico positive unless violating constraints (4.15)-(4.17).
- *Step 2: we prove that, given* $\mathbf{x}_i(\epsilon)$ *returned at iteration* $k-1$*, if the algorithm terminates at iteration* k *returning* non-existence*, then there is no other perturbed strategy satisfying constraints (4.13)-(4.17).* By construction, any perturbed strategy fulfilling program (4.13)-(4.17) satisfies constraints (4.18)-(4.19) at $k = 0$ and constraints (4.21)-(4.26) at $k \geq 1$. Therefore, each such a perturbed strategy can be found by Algorithm 6 with an opportune objective function in place of objective function (4.20). However, constraints (4.21)-(4.27) strictly relax from an iteration to the subsequent one. More precisely, constraints (4.22), (4.24), and (4.26) relax, while the others keep to be the same. Therefore, all the possible paths that Algorithm 6 can follow with any possible objective function lead to the same result in terms of existence or non-existence of a strategy $\mathbf{x}_i(\epsilon)$ fulfilling constraints (4.13)-(4.17). We notice that the perturbed strategies found by the algorithm could be different if different paths are followed. Among all the objective functions, objective function (4.20) minimizes the number of iterations, maximizing at each iteration the number of strictly lexico-positive strategies.

Therefore, the algorithm is complete. □

Lemma 4.5 *Algorithm 6 runs in polynomial time in the size of the game.*

Proof. We prove the lemma by showing that the number of iterations of the algorithm is in the worst case linear in the size of the game. At each iteration k, either some $x_i(q, \epsilon^k)$ that is zero for every $k' < k$ becomes strictly positive or the algorithm stops. In the worst case, only one sequence becomes strictly lexico positive at each iteration and therefore the number of iterations is equal to the number of sequences $|Q_i|$. Thus, since linear mathematical programming runs in polynomial time, the lemma is proved. □

Hence, Algorithm 6 solves program (4.13)-(4.17) by Lemma 4.3 and Lemma 4.4, and it requires polynomial time by Lemma 4.5. This completes the proof. □

We are now in the position to prove the following theorem.

Theorem 4.3 *There exists an efficient algorithm certifying that a behavioral strategy profile $\boldsymbol{\sigma}$ given as input is a QPE with two-agent games.*

Proof. By Proposition 4.8, the problem of certifying that a behavioral strategy profile $\boldsymbol{\sigma}$ is a QPE with two-agent games can be formulated as a pair of mathematical programs whose resolution can be achieved, by Proposition 4.10, in polynomial time. □

We provide two examples to which we apply Algorithm 6.

Example 4.3 *Consider the game depicted in Fig. 4.2, where* $\boldsymbol{\sigma} = (\sigma_1(\mathsf{M}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_1(\mathsf{R}_3) = 1, \sigma_2(\mathsf{l}_1) = 1, \sigma_2(\mathsf{l}_2) = 1, \sigma_2(\mathsf{r}_3) = 1)$. *We verify whether or not it is a QPE. Initially, we report the utility bimatrix in sequence form:*

		agent 2						
		$\mathsf{q}_\varnothing$	l_1	r_1	$\mathsf{l}_1\mathsf{l}_2$	$\mathsf{l}_1\mathsf{r}_2$	$\mathsf{r}_1\mathsf{l}_3$	$\mathsf{r}_1\mathsf{r}_3$
agent 1	$\mathsf{q}_\varnothing$							
	L_1	$8,0$						
	M_1							
	R_1							
	$\mathsf{M}_1\mathsf{L}_2$				$8,6$	$7,1$	$0,0$	$6,1$
	$\mathsf{M}_1\mathsf{R}_2$				$5,8$	$4,5$	$7,1$	$2,0$
	$\mathsf{R}_1\mathsf{L}_3$				$0,1$	$4,5$	$9,1$	$0,6$
	$\mathsf{R}_1\mathsf{R}_3$				$3,1$	$5,8$	$6,4$	$1,1$

For the sake of presentation, we omit matrices F_1 and F_2. We apply Algorithm 6 to find $\mathbf{x}_1(\epsilon)$:

- iteration 0*: the strategy prescribes:* $x_1(\mathsf{M}_1, \epsilon^0) = x_1(\mathsf{M}_1\mathsf{L}_2, \epsilon^0) = 1$*, while the other sequences are played with a probability of zero; the strategy satisfies constraints (4.18)-(4.19);*

- iteration 1*: in constraints (4.24) the only sequence with a probability of zero is* L_3*; the maximization is over all the sequences except* M_1 *and* $\mathsf{M}_1\mathsf{L}_2$*; the resulting strategy prescribes:* $x_1(\mathsf{L}_1, \epsilon^1) = x_1(\mathsf{R}_1, \epsilon^1) = x_1(\mathsf{M}_1\mathsf{R}_2, \epsilon^1) = x_1(\mathsf{R}_1\mathsf{R}_3, \epsilon^1) = 1$ *and* $x_1(\mathsf{M}_1, \epsilon^1) = -2$ *and* $x_1(\mathsf{M}_1\mathsf{L}_2, \epsilon^1) = -3$*, while the other sequences are played with a probability of zero;*

- iteration 2*: no sequence appears in constraints (4.24); the maximization is only over* L_3*; the resulting strategy prescribes:* $x_1(\mathsf{L}_3, \epsilon^2) = 1$ *and* $x_1(\mathsf{R}_3, \epsilon^2) = -1$.

Summarily the resulting $\mathbf{x}_1(\epsilon)$ *is (*$\mathsf{q}_\varnothing$ *is omitted for simplicity):*

	L_1	M_1	R_1	$\mathsf{M}_1\mathsf{L}_2$	$\mathsf{M}_1\mathsf{R}_2$	$\mathsf{R}_1\mathsf{L}_3$	$\mathsf{R}_1\mathsf{R}_3$
ϵ^0	0	1	0	1	0	0	0
ϵ^1	1	−2	1	−3	1	0	1
ϵ^2	0	0	0	0	0	1	−1

Therefore, there exists a fully mixed strategy $\lim_{\epsilon \to 0} \boldsymbol{\sigma}_1(\epsilon) \to \boldsymbol{\sigma}_1$ *such that* $\boldsymbol{\sigma}_2$ *is best response to* $\boldsymbol{\sigma}_1(\epsilon)$.

Now, we apply Algorithm 6 to $\mathbf{x}_2(\epsilon)$*. In this case, the algorithm stops with the resulting strategy (*$\mathsf{q}_\varnothing$ *is omitted for simplicity):*

	l_1	r_1	$\mathsf{l}_1\mathsf{l}_2$	$\mathsf{l}_1\mathsf{r}_2$	$\mathsf{r}_1\mathsf{l}_3$	$\mathsf{r}_1\mathsf{r}_3$
ϵ^0	1	0	1	0	0	0
ϵ^1	0	0	0	0	0	0

At iteration 1, the algorithm stops because the objective function is zero. The reason is that any possible perturbation over $\mathbf{x}_2(\epsilon)$ *would make the probability with which agent 2 plays* $\mathsf{l}_1\mathsf{l}_2$ *strictly smaller than one and therefore the expected utility of agent 1 from playing* M_1 *would be strictly smaller than 8. Thus, agent 1 would play* L_1 *gaining exactly 8. In practice, the algorithm cannot assign a strictly positive perturbation over* $\mathsf{l}_1\mathsf{r}_2$ *without violating the constraints of best response of agent 1. As a result, there is no fully mixed* $\boldsymbol{\sigma}_2(\epsilon)$ *such that* $\boldsymbol{\sigma}_1$ *is a best response and, therefore,* $\boldsymbol{\sigma}$ *is not a QPE.*

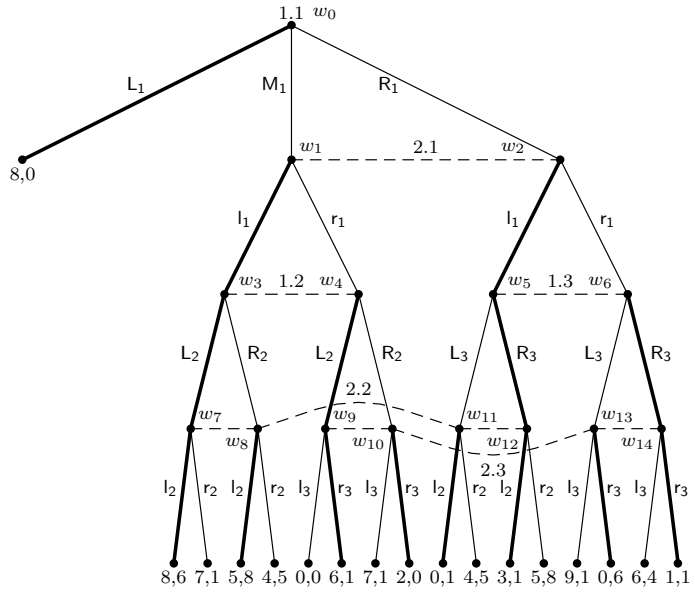


Figure 4.3: Example of strategy profile $\boldsymbol{\sigma}$ expressed in behavioral strategies that is a QPE ($\boldsymbol{\sigma}$ is represented by using bold lines to denote actions played with positive probability).

Example 4.4 *Consider the game depicted in Fig. 4.3, where* $\boldsymbol{\sigma} = (\sigma_1(\mathsf{L}_1) = 1, \sigma_1(\mathsf{L}_2) = 1, \sigma_1(\mathsf{R}_3) = 1, \sigma_2(\mathsf{l}_1) = 1, \sigma_2(\mathsf{l}_2) = 1, \sigma_2(\mathsf{r}_3) = 1)$. *We verify whether or not it is a QPE. From the application of Algorithm 6, we obtain the following fully mixed* $\mathbf{x}_1(\epsilon)$ *($\mathsf{q}_\varnothing$ is omitted for simplicity):*

	L_1	M_1	R_1	$\mathsf{M}_1\mathsf{L}_2$	$\mathsf{M}_1\mathsf{R}_2$	$\mathsf{R}_1\mathsf{L}_3$	$\mathsf{R}_1\mathsf{R}_3$
ϵ^0	1	0	0	0	0	0	0
ϵ^1	$-\frac{5}{4}$	1	$\frac{1}{4}$	1	0	0	$\frac{1}{4}$
ϵ^2	0	0	0	$-\frac{1}{2}$	$\frac{1}{2}$	1	-1

and the following fully mixed $\mathbf{x}_2(\epsilon)$ *($\mathsf{q}_\varnothing$ is omitted for simplicity):*

	l_1	r_1	$\mathsf{l}_1\mathsf{l}_2$	$\mathsf{l}_1\mathsf{r}_2$	$\mathsf{r}_1\mathsf{l}_3$	$\mathsf{r}_1\mathsf{r}_3$
ϵ^0	1	0	1	0	0	0
ϵ^1	-1	1	-2	1	0	1
ϵ^2	0	0	0	0	1	-1

therefore $\boldsymbol{\sigma}$ *is a QPE.*

On the basis of the above results, we can state the following corollary.

Corollary 4.2 *The problem of searching for an exact QPE of a game with with two agents is in $\mathcal{TFNP}$.*

Proof. The membership of the above problem to $\mathcal{TFNP}$ follows from:

- the functional problem is total since every game admits at least one QPE,
- there exists an efficient algorithm for the verification problem of QPE as shown by Theorem 4.3.

This completes the proof of the corollary. □

Now, we show how a simple variation of Algorithm 4 can be applied to the verification problem of SE with two agents.

Initially, we define the binary relation $\geq_{\text{lex-weak}}$ as follows.

Definition 4.7 ($\geq_{\text{lex-weak}}$) *Given vectors $\mathbf{y}_1, \mathbf{y}_2$, we have $\mathbf{y}_1 \geq_{\text{lex-weak}} \mathbf{y}_2$ if and only if $\mathbf{y}_1 \geq_{\text{lex}} \mathbf{y}_2$ or, let*

- *k_1 the minimum k such that $y_{1,k} \neq 0$ and*
- *k_2 the minimum k such that $y_{2,k} \neq 0$,*

$k_1 = k_2 = k$ and $y_{1,k} = y_{2,k}$.

Notice that the above relation may be true even when $\mathbf{y}_2 \geq_{\text{lex}} \mathbf{y}_1$.

On the basis of the above relation, we can formulate the verification problem for SE with two agents when only strategies are provided as a non-linear mathematical program. We denote by $q \to h'$ that there exists some sequence of the opponent such that, combined with sequence q, leads to information set h'.

Proposition 4.11 *Strategy profile $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ given as input is part of an SE if and only if program*

$$\text{Constraints } (4.13),\ (4.14),\ (4.16),\ (4.17)$$

$$v_{-i,h}(\epsilon) \geq_{\text{lex-weak}} \sum_{h':q\to h'} v_{-i,h'}(\epsilon) + (U_{-i}^T \cdot \mathbf{x}_i(\epsilon))_q$$
$$\forall h \in H_{-i}, q \in Q_{-i} : a(q) \in \rho(h) \quad (4.28)$$

admits solution for every $i \in N$.

Proof. The above program differs from program (4.13)-(4.17) presenting constraints (4.28) in place of constraints (4.15). The differences between these two groups of constraints lay only in the use of $\geq_{\text{lex-weak}}$ in constraints (4.28) in place of $\geq_{\text{lex}}$ used in constraints (4.15). In addition, in constraints (4.28) we explicitly detail all the terms instead of using the matrix-like representation used in constraints (4.15) to specify the terms on the left hand of $\geq_{\text{lex-weak}}$ and those on the right hand. By definition of $\geq_{\text{lex-weak}}$, the above program is satisfied by all the solutions of program (4.13)-(4.17). Furthermore, the above program, thanks to $\geq_{\text{lex-weak}}$, can be satisfied by additional solutions in which the value $v_{-i,h}$ is equal to the expected utility provided by the best sequences when perturbations are not present, but it may be smaller when perturbations are present. This is exactly the condition required by SE definition. □

We focus on the complexity of solving the above mathematical program.

Proposition 4.12 *Given* $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ *as input, there exists an efficient algorithm finding a solution of program (4.13), (4.14), (4.16), (4.17), (4.28).*

Proof. Program (4.13), (4.14), (4.16), (4.17), (4.28) can be solved with a simple variation of Algorithm 6. More precisely, constraints (4.26) must be substituted with the following constraints:

$$v_{-i,h}(\epsilon^k) \geq \sum_{h':q\to h'} v_{-i,h'}(\epsilon^k) + (U_{-i}^T \cdot \mathbf{x}_i(\epsilon^k))_q$$
$$\forall h \in H_{-i}, q \in Q_{-i} : a(q) \in \rho(h),$$
$$v_{-i,h}(\epsilon^{k'}) = \sum_{h':q\to h'} v_{-i,h'}(\epsilon^{k'}) + (U_{-i}^T \cdot \mathbf{x}_i(\epsilon^{k'}))_q = 0,$$
$$\forall k' < k \quad (4.29)$$

The above constraints relax constraints (4.26), requiring $\geq_{\text{lex-weak}}$ in place of $\geq_{\text{lex}}$. Therefore, a solution can be found in polynomial time. □

Finally, we can prove the following theorem.

Theorem 4.4 *There exists an efficient algorithm certifying that a behavioral strategy profile $\boldsymbol{\sigma}$ given as input is part of an SE with two-agent games.*

Proof. The problem of certifying that a behavioral strategy profile $\boldsymbol{\sigma}$ is an SE with two-agent games can be formulated, by Proposition 4.11, as a pair of programs (4.13), (4.14), (4.16), (4.17), (4.28) and, by Proposition 4.12, these programs can be solved in polynomial time. □

4.3 Related works

We summarize in Tab. 4.1 the main computational complexity results on the problem of verifying whether a given solution is an equilibrium according to some given NE refinement solution concept for extensive-form games.

NE and SPE. The verification problem for NE is known to be easy, even when the input is given in sequence form, just requiring the verification that a finite number of constraints are satisfied. The verification problem for SPE is easy when the input is completely specified with behavioral strategies, while it is $\mathcal{NP}$-hard when the input is given in sequence form [21].

SE. The only known result on the verification for SE is [38]. The authors provided a finite-step algorithm to verify whether an assessment is an SE, but, as they state it, the algorithm is exponential in the worst case. In the present chapter, we show that there is an efficient algorithm with an arbitrary number of agents. Furthermore, it is known that, when the input is partially specified, the verification problem for an SE can be harder. More precisely, the authors show in [32] that with three or more agents, verifying whether there is an SE with a given strategy is $\mathcal{NP}$-hard. In the present chapter, we show instead that with two agents there is an efficient algorithm for the verification problem.

Perfection based equilibria. The verification problem for normal-form perfect equilibrium (NFPE) is in $\mathcal{P}$ with two agents and can be accomplished by checking whether or not the actions played with strictly positive probability are weakly dominated. In the former case, the solution is not an NFPE. With more than two agents it is shown

Solution concept	Input	Agents	
		Two	Three or more
NE	strategies in sequence form	$\mathcal{P}$	$\mathcal{P}$
SPE	strategies	$\mathcal{P}$	$\mathcal{P}$
	strategies in sequence form	$\mathcal{NP}$-hard [21]	$\mathcal{NP}$-hard [21]
SE	assessment	$\mathcal{P}$ (*)	$\mathcal{P}$ (*)
	strategies	$\mathcal{P}$ (*)	$\mathcal{NP}$-hard [32]
NFPE	strategies	$\mathcal{P}$	$\mathcal{NP}$-hard [32]
QPE	strategies	$\mathcal{P}$ (*)	$\mathcal{NP}$-hard [32]
EFPE	strategies		$\mathcal{NP}$-hard [32]
-PE	strategies and perturbation	$\mathcal{P}$	$\mathcal{P}$
NFPrE	strategies		$\mathcal{NP}$-hard [32]
EFPrE	strategies		$\mathcal{NP}$-hard [32]
-PrE	strategies and perturbation	$\mathcal{P}$	$\mathcal{P}$

Table 4.1: Computational complexity results for equilibrium verification; the results with '(*)' are originally provided in the present thesis.

to be $\mathcal{NP}$-hard in [32]. The verification problem for QPE is $\mathcal{NP}$-hard with three or more agents [32]. In the present chapter, we show instead that with two agents there is an efficient algorithm. The verification problem for EFPE is open with two agents. With three or more agents the problem is shown to be $\mathcal{NP}$-hard [32]. The problem of verifying whether a strategy profile is a perfect equilibrium for a given perturbation is easy for all the perfect equilibrium solution concepts (i.e., NFPE, QPE, EFPE). Indeed, it requires one only to check whether a finite number of constraints are satisfied.

Properness based equilibria. The computational complexity of verifying a normal-form proper equilibrium (NFPrE) with two agents is open and the only available algorithm is [6] that solves a number of mixed quadratic programs, while with three or more agents the problem is shown to be $\mathcal{NP}$-hard [32]. The recent result presented in [62], showing that computing a NFPrE with two agents is

in $\mathcal{PPAD}$ and can be performed by using Lemke's algorithm after a specific transformation, could represent an interesting tool to assess the complexity of verifying an NFPrE with two agents. Also the problem of verifying an extensive-form proper equilibrium (EFPrE) with two agents is open. The problem of verifying whether a strategy profile is a proper equilibrium for a given perturbation is easy for all the proper equilibrium solution concepts (i.e., NFPrE, EFPrE). Indeed, it requires one only to check whether a finite number of constraints are satisfied.

Extensive-form perfect equilibrium computation with two-player games 5

In this chapter we focus on the problem of designing an algorithm that, given a specific perturbation over the sequence-form strategies of a two-player extensive-form game, computes an EFPE.

In this chapter, we show that

- given a perturbation over the behavioral strategies of the agents, an EFPE with two-agent general-sum games can be computed exactly by means of a variation of the Lemke's algorithm in which the perturbation is over the coefficients of the variables. We show that the aforementioned variation keeps the the problem is in $\mathcal{PPAD}$.

This chapter is structured as follows. In Section 5.1, we propose a specific perturbation over the sequence form leading to an EFPE. In Section 5.2, we derive the equilibrium constraints with perturbed sequence strategies. In section 5.3, we describe a standard linear complementarity programming (LCP) formulation to solve our problem. In Section 5.4, we show how the Lemke's algorithm must be modified to solve the perturbed problem and we show that our algorithm is complete. In Section 5.5 we describe the implementation of our algorithm and we experimental evaluate it. Finally, in Section 5.6, we survey related works.

5.1 EFPE and perturbed games

We introduce a *multiplicative* symbolic perturbation $\epsilon \geq 0$ over all the sequences q such that $x_i(q|a) \geq \epsilon \cdot x_i(q)$ for every $q \in Q_i$ and $a \in A_i$ such that $q|a \in Q_i$. Perturbation ϵ can be interpreted as an arbitrarily small value.

Theorem 5.1 *A NE of the perturbed game in which $x_i(q|a) \geq \epsilon \cdot x_i(q)$ converges to an EFPE when $\epsilon \to 0$.*

Proof. The proof is based on the fact that the perturbation $x_i(q|a) \geq \epsilon \cdot x_i(q)$ over the sequences induces a perturbation of ϵ over all the behavioral strategies satisfying the definition of EFPE. At first, we observe that all the sequences are played with strictly positive probability. This is because the empty sequence $\mathsf{q}_\varnothing$ is played with a probability of one and, by definition of the above perturbation, every sequence $q|a$ is played with strictly positive probability if sequence q is played with strictly positive probability. Thus, by induction, all the sequences are played with strictly positive probability. At second, we observe that the perturbation induced over the behavior strategies is $\sigma_i(a) = \frac{x_i(q|a)}{x_i(q)} \geq \epsilon$ independently of the strategy played in the previous information set. As a result each action a is subject to an independent perturbation as required by the definition of EFPE. This completes the proof. □

5.2 Equilibrium constraints derivation

We initially consider the unperturbed best response problem of agent i in sequence form given the strategy of opponent agent $-i$:

$$\begin{aligned} \max_{\mathbf{x}_i} \quad & \mathbf{x}_i^T \cdot U_i \cdot \mathbf{x}_{-i} \\ & F_i \cdot \mathbf{x}_i \quad = \mathbf{f}_i \\ & \mathbf{x}_i \quad \geq \mathbf{0} \end{aligned}$$

Now we consider the perturbation introduced in the previous section. Writing $x_i(q|a) \geq \epsilon \cdot x_i(q)$ as $x_i(q|a) - \epsilon \cdot x_i(q) \geq 0$ for every q and a, we obtain:

$$R_i(\epsilon) \cdot \mathbf{x}_i \geq \mathbf{0}$$

where $R_i(\epsilon)$ is a matrix whose entries depend on ϵ. Matrix $R_i(\epsilon)$ is *lower unitriangular*, i.e., a triangular matrix where all the entries in the main diagonal are ones.

Example 5.1 *Matrices $R_1(\epsilon)$ and $R_2(\epsilon)$ in the game depicted in Fig. 2.1 are the following:*

$$R_1(\epsilon) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\epsilon & 1 & 0 & 0 & 0 \\ -\epsilon & 0 & 1 & 0 & 0 \\ 0 & 0 & -\epsilon & 1 & 0 \\ 0 & 0 & -\epsilon & 0 & 1 \end{bmatrix}, \qquad R_2(\epsilon) = \begin{bmatrix} 1 & 0 & 0 \\ -\epsilon & 1 & 0 \\ -\epsilon & 0 & 1 \end{bmatrix}$$

We introduce the perturbation in the best response problem:

$$\begin{aligned} \max_{\mathbf{x}_i} \quad & \mathbf{x}_i^T \cdot U_i \cdot \mathbf{x}_{-i} \\ & F_i \cdot \mathbf{x}_i \quad = \mathbf{f}_i \\ & R_i(\epsilon) \cdot \mathbf{x}_i \quad \geq \mathbf{0} \end{aligned}$$

We do a change of variable, defining $\tilde{\mathbf{x}}_i = R_i(\epsilon) \cdot \mathbf{x}_i$. In this way, we have $\tilde{\mathbf{x}}_i \geq \mathbf{0}$. In order to use $\tilde{\mathbf{x}}_i$ in the best response problem, we need that matrix $R_i(\epsilon)$ is invertible. This property is satisfied for every $\epsilon \geq 0$ because $R_i(\epsilon)$ is triangular. In addition, $R_i^{-1}(\epsilon)$ is lower unitriangular and all its entries are positive.

Example 5.2 *Matrices $R_1^{-1}(\epsilon)$ and $R_2^{-1}(\epsilon)$ in the game depicted in Fig. 2.1 are the following:*

$$R_1^{-1}(\epsilon) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \epsilon & 1 & 0 & 0 & 0 \\ \epsilon & 0 & 1 & 0 & 0 \\ \epsilon^2 & 0 & \epsilon & 1 & 0 \\ \epsilon^2 & 0 & \epsilon & 0 & 1 \end{bmatrix}, \qquad R_2^{-1}(\epsilon) = \begin{bmatrix} 1 & 0 & 0 \\ \epsilon & 1 & 0 \\ \epsilon & 0 & 1 \end{bmatrix}$$

Now we substitute $\mathbf{x}_i$ with $\tilde{\mathbf{x}}_i$ in the best response problem of agent i:

$$\begin{aligned} \max_{\tilde{\mathbf{x}}_i} \quad & \tilde{\mathbf{x}}_i^T \cdot R_i^{-T}(\epsilon) \cdot U_i \cdot R_{-i}^{-1} \tilde{\mathbf{x}}_{-i} \\ & F_i \cdot R_i^{-1}(\epsilon) \cdot \tilde{\mathbf{x}}_i \quad = \mathbf{f}_i \\ & \tilde{\mathbf{x}}_i \quad \geq \mathbf{0} \end{aligned}$$

where the apex "$-T$" denotes the transpose of the inverse matrix. We derive the dual problem:

$$\begin{aligned} \min_{\mathbf{v}_i} \quad & \mathbf{f}_i^T \cdot \mathbf{v}_i \\ & R_i^{-T}(\epsilon) \cdot F_i^T \cdot \mathbf{v}_i \quad \geq R_i^{-T}(\epsilon) \cdot U_i \cdot R_{-i}^{-1} \tilde{\mathbf{x}}_{-i} \\ & \mathbf{v}_i \qquad\qquad \text{free in sign} \end{aligned}$$

Given that the primal and the dual best response problems are linear, strong duality holds. Thus, we can derive the complementary slackness conditions as follows:

$$F_i \cdot R_i^{-1}(\epsilon) \cdot \tilde{\mathbf{x}}_i = \mathbf{f}_i \quad \forall i \in \{1,2\} \tag{5.1}$$

$$R_i^{-T}(\epsilon) \cdot F_i^T \cdot \mathbf{v}_i - R_i^{-T}(\epsilon) \cdot U_i \cdot R_{-i}^{-1}(\epsilon) \cdot \tilde{\mathbf{x}}_{-i} \geq \mathbf{0} \quad \forall i \in \{1,2\} \tag{5.2}$$

$$\tilde{\mathbf{x}}_i \geq 0 \quad \forall i \in \{1,2\} \tag{5.3}$$

$$\tilde{\mathbf{x}}_i^T \cdot (R_i^{-T}(\epsilon) \cdot F_i^T \mathbf{v}_i - R_i^{-T}(\epsilon) \cdot U_i \cdot R_{-i}^{-1}(\epsilon) \cdot \tilde{\mathbf{x}}_{-i}) = 0 \quad \forall i \in \{1,2\} \tag{5.4}$$

Let us note that we cannot simplify constraints (5.2) by multiplying by $R_i^T(\epsilon)$ (at left). This is mainly because $R_i^T(\epsilon)$ contains negative entries that would change, for some rows of the constraints, the direction of the inequality.

5.3 Standard LCP formulation

A standard linear complementarity programming (LCP) problem is formulated as:

$$\begin{aligned} \mathbf{z} &\geq \mathbf{0} \\ \mathbf{w} &\geq \mathbf{0} \\ \mathbf{w} &= M\mathbf{z} + \mathbf{b} \\ \mathbf{w}^T\mathbf{z} &= 0 \end{aligned}$$

where the variables are $\mathbf{z}, \mathbf{w} \in \mathbb{R}^n$ and the parameters are M (square matrix $n \times n$) and $\mathbf{b} \in \mathbb{R}^n$. Each pair of variables (w_i, z_i) are said *complementary* because the constraint $\mathbf{w}^T\mathbf{z} = 0$ forces at least one variable

of the pair (w_i, z_i) to be equal to zero. In non-trivial LCPs, at least one component of $\mathbf{b}$ is strictly negative, otherwise the problem would admit the trivial solution $\mathbf{z} = \mathbf{0}, \mathbf{w} = \mathbf{b}$.

The Lemke's algorithm [43], a generalization of the Lemke-Howson's algorithm [44], is one of the algorithms that can be employed to solve an LCP and works as follows.

1. Introduce an auxiliary variable $z_0 \in \mathbb{R}^+$ such that $\mathbf{w} = M\mathbf{z} + \mathbf{d}z_0 + \mathbf{b}$ with $\mathbf{d} = \mathbf{1}$. Build the associated *augmented tableau*:

	$\mathbf{w}^T$	$\mathbf{z}^T$	z_0	
$\mathbf{w}$	I	$-M$	$-\mathbf{d}$	$\mathbf{b}$

   The current basis, composed of the variables $\mathbf{w}$, is not feasible, some component of $\mathbf{b}$ being strictly negative.

2. Make a pivoting step in which the entering variable is z_0 and the leaving variable is the $w_{\overline{k}}$ such that $b_{\overline{k}}$ is the most negative value. This pivoting step leads to a feasible basis composed of z_0 and of all the variables w_j except $w_{\overline{k}}$. The current basis is said *almost complementary* since there is only one complementary pair $(w_{\overline{k}}, z_{\overline{k}})$ such that both variables are not in the basis. The tableau in the current state is said the *initial tableau*.

3. Make repeatedly a pivoting step until z_0 leaves the basis, in which the entering variable is the complementary variable of the leaving variable in the previous pivoting step and the leaving variable is chosen by the minimum ratio test: the variable in basis at the position $\arg\min_k \{b_k/t_{k,j} : t_{k,j} > 0\}$ where $t_{k,j}$ is an entry of the tableau and j is the column of the entering variable. At each pivoting step there is a unique possible leaving variable (excluded degenerate cases discussed below) and a unique entering variable. This leads the algorithm to follow a path of almost complementary bases.

The Lemke's algorithm may return a failure, said *ray termination*, even if the problem is feasible. A ray termination happens when all the entries $t_{k,j}$ in the column j of the entering variables are non-positive. The literature provides some sufficient conditions defined on M and $\mathbf{b}$ under which the algorithm cannot reach any ray termination (see, e.g., [70]). We report the conditions we use in our paper:

Theorem 5.2 *The Lemke's algorithm is complete if the following two properties are satisfied:*

- $\forall \mathbf{z} \geq \mathbf{0}$ *we have* $\mathbf{z}^T M \mathbf{z} \geq 0$,
- $\forall \mathbf{z} \geq \mathbf{0}$ *such that* $\mathbf{z}^T M \mathbf{z} = 0$ *and* $M\mathbf{z} = \mathbf{0}$ *we have* $\mathbf{z}^T \mathbf{b} \geq \mathbf{0}$.

We can write constraints (5.1)-(5.4) as a standard LCP problem where

$$\mathbf{z} = \begin{bmatrix} \tilde{\mathbf{x}}_1 \\ \tilde{\mathbf{x}}_2 \\ \mathbf{v}_1^+ \\ \mathbf{v}_1^- \\ \mathbf{v}_2^+ \\ \mathbf{v}_2^- \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_1 \\ -\mathbf{f}_1 \\ \mathbf{f}_2 \\ -\mathbf{f}_2 \end{bmatrix}$$

and

$$M = \begin{bmatrix} 0 & -R_1^{-T}(\epsilon) U_1 R_2^{-1}(\epsilon) & R_1^{-T}(\epsilon) F_1^T \\ -R_2^{-T}(\epsilon) U_2^T R_1^{-1}(\epsilon) & 0 & 0 \\ -F_1 R_1^{-1}(\epsilon) & 0 & 0 \\ F_1 R_1^{-1}(\epsilon) & 0 & 0 \\ 0 & -F_2 R_2^{-1}(\epsilon) & 0 \\ 0 & F_2 R_2^{-1}(\epsilon) & 0 \end{bmatrix}$$

$$\begin{bmatrix} -R_1^{-T}(\epsilon) F_1^T & 0 & 0 \\ 0 & R_2^{-T}(\epsilon) F_2^T & -R_2^{-T}(\epsilon) F_2^T \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

A standard LCP can be solved by the Lemke's algorithm. However, the Lemke's algorithm does not deal with perturbation over the entries of matrix M. In the following sections, we show how the Lemke's algorithm can be modified to find an EFPE.

5.4 Revised Lemke's algorithm

The presence of symbolic perturbation ϵ over the entries of matrix M requires two modifications of the Lemke's algorithm. Specifically,

the Lemke's algorithm keeps to be same of that described in Section 5.3 except that we need to modify the minimum ratio test and how a pivoting step is conducted.

Minimum ratio test

Initially, we focus on the minimum ratio test. According to its definition, the leaving variable is the variable in basis at the position:

$$\arg\min_k \left\{ \frac{b_k(\epsilon)}{t_{k,j}(\epsilon)} : t_{k,j}(\epsilon) > 0 \right\}$$

where $t_{k,j}(\epsilon)$ is an entry of the tableau and j is the column of the entering variable. Given that ϵ is a symbolic parameter and we deal with it in lexicographic fashion, we need to redefine the minimum ratio test. We observe that the peculiarity lays in $t_{k,j}(\epsilon)$: it is a polynomial in ϵ. The division $\frac{b_k(\epsilon)}{t_{k,j}(\epsilon)}$ keeping ϵ as a symbol is not generally allowed, $b_k(\epsilon)$ not being necessarily a multiple of $t_{k,j}(\epsilon)$. Thus, we say that $\frac{b_k(\epsilon)}{t_{k,j}(\epsilon)}$ is the minimum between $\frac{b_k(\epsilon)}{t_{k,j}(\epsilon)}$ and $\frac{b'_k(\epsilon)}{t_{k',j}(\epsilon)}$ if

$$b_{k'}(\epsilon) \cdot t_{k,j}(\epsilon) >_{\text{lex}} b_k(\epsilon) \cdot t_{k',j}(\epsilon)$$

The direction of the inequality does not change, given that $t_{k,j}(\epsilon)$, $t_{k',j}(\epsilon) >_{\text{lex}} 0$ by definition of minimum ratio test. On the basis of this definition of minimum, we have that the (lexico) minimum ratio test is:

$$\arg\operatorname{lex}\min_k \left\{ \frac{b_k(\epsilon)}{t_{k,j}(\epsilon)} : t_{k,j}(\epsilon) >_{\text{lex}} 0 \right\}$$

Pivoting step

Now, we focus on the pivoting step. Here, the same above problem due to the division by $t_{k,j}(\epsilon)$ is present. Indeed, call $\bar{t}_{k,j}(\epsilon)$ the pivot, during each pivoting step it would be necessary to divide all the entries of row k by the pivot, but some of these entries may be not multiples of the pivot. A simple solution to this problem would be to avoid the division, but with this solution the maximum degree of ϵ in the entries $t_{k,j}(\epsilon)$ of the tableau would double at each pivoting step. As a result, the memory occupancy would rise exponentially and the computational time would do the same, requiring in the lexico minimum ratio test to compare an exponential number of coefficients. In

order to get around this problem, integer pivoting can be adopted. Integer pivoting does not perform any division during the pivoting step, but at the end of the pivoting step, all the entries of the tableau except those in the row of the pivot are divided by the pivot of the previous step. This division is theoretically assured to be integer and can be performed in polynomial time by means of *polynomial long division* algorithm [4]. In addition, this division allows the maximum degree of ϵ in the entries $t_{k,j}(\epsilon)$ of the tableau to rise linearly at each pivoting step.

Hence, dealing with perturbations over matrix M we have that

- the memory occupancy rises linearly in the number of pivoting steps,
- the time needed for each pivoting step rises polynomially in the number of pivoting steps.

Therefore, the worst case complexity of solving an LCP with perturbation over M is linear w.r.t. the worst case complexity of solving an LCP without perturbation over M.

Degeneracy and lexicographic perturbation

In presence of degeneracy, more vertices of the feasible region overlap and, practically, the minimum ratio test can return more than one possible leaving variables. In this case, it is necessary to break the ties, otherwise the algorithm could cycle. A way to address degeneracy is based on *lexicographic perturbation*. Given two vectors $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$, $\mathbf{x}_1$ is the *lexico-minimum* (lex min) if $\mathbf{x}_2 - \mathbf{x}_1 >_{\text{lex}} 0$. Lexicographic perturbation is used as follows.

1. Define the vector $\boldsymbol{\delta} = [\delta^1 \ \ \delta^2 \ \ \dots \ \ \delta^n]$ where δ is a symbolic value that goes to zero.
2. Define a full-rank matrix Y (e.g., the identity matrix I) and redefine the constraint $\mathbf{w} = M\mathbf{z} + \mathbf{b}$ as $\mathbf{w} = M\mathbf{z} + \mathbf{b} + Y\boldsymbol{\delta}$. As $\delta \to 0$, the new constraint coincides with the original one, and therefore all the solutions fulfilling the perturbed constraints fulfill also the original constraints. The term $Y\boldsymbol{\delta}$ is considered a symbolic perturbation of $\mathbf{b}$. The new initial tableau is:

	$\mathbf{w}^T$	$\mathbf{z}^T$	z_0		$\boldsymbol{\delta}^T$
$\mathbf{w}$	I	$-M$	$-\mathbf{d}$	$\mathbf{b}$	Y

3. The symbolic perturbation is used only for the minimum ratio test. In this case, the leaving variable is the one at the position:

$$\arg\operatorname{lex}\min_k \left\{ \frac{[b_k \;\; y_{k,1} \;\; \dots \;\; y_{k,n}]}{t_{k,j}} : t_{k,j} > 0 \right\}$$

where $y_{k,j}$ is an entry of Y and j is the column of the entering variable.

In the problem of computing an EFPE the presence of perturbation over M does not provide any guarantee over the non-existence of degeneracy. Thus, we need to introduce a perturbation over the constants $\mathbf{b}$ also in the LCP described in Section 5.3.

We define an alphabet of symbols $\Sigma = (\delta, \epsilon)$ with a total order such that $\delta < \epsilon^k$ for every k. We use $\delta > 0$ as a new perturbation and this perturbation is by definition smaller than any power of ϵ. In this way, in breaking ties in the minimum ratio test, we consider perturbation δ only after having considered all the degrees of ϵ^k.

Without perturbation $\boldsymbol{\epsilon}$, the constants can be seen as a vector of univariate polynomials in δ. With perturbation δ, the polynomials are bivariate in δ and ϵ. We denote by $c_k(\delta, \epsilon)$ the bivariate polynomial expressing the constant in row k. We have: $c_k(\delta, \epsilon) = \sum_i \sum_j \overline{c}^k_{i,j} \delta^i \epsilon^j$ where $\overline{c}^k_{i,j}$ are coefficients. Given the above alphabet Σ, we have that $c_k(\epsilon, \delta) <_{\text{lex}} 0 \leftrightarrow \exists m, m' : (\forall i < m, \forall j, \overline{c}^k_{i,j} = 0) \wedge (i = m, \forall j < m', \overline{c}^k_{i,j} = 0) \wedge (i = m, j = m', \overline{c}^k_{i,j} < 0)$. In words, we scour the coefficients lexicographically to look for the first non-zero coefficient. Therefore, we can write the lexico minimum ratio test as

$$\arg\operatorname{lex}\min_k \left\{ \frac{[b_k(\epsilon^0) \; \dots \; b_k(\epsilon^\gamma) \; y_{k,1}(\epsilon^0) \; \dots \; y_{k,1}(\epsilon^\gamma) \; y_{k,2}(\epsilon^0) \; \dots \; y_{k,n}(\epsilon^\gamma)]}{t_{j,k}(\epsilon)} : t_{j,k}(\epsilon) >_{\text{lex}} 0 \right\}$$

where γ denotes the highest degree of ϵ (notice that this degree changes along the pivoting steps, increasing monotonically).

Ray termination

The Lemke's algorithm can go in ray termination without returning any feasible solution even if there is a feasible solution. In our case, we can show that the Lemke's algorithm is complete.

Theorem 5.3 *For every possible value of ϵ, the Lemke's algorithm always finds a feasible solution of the above standard LCP when U_1 and U_2 are negative.*

Proof. It is possible to show that the Lemke's algorithm cannot reach any ray termination under some conditions over M and $\mathbf{b}$, always finding a feasible solution if the LCP admits one. The above LCP always admits a feasible solution given that every perturbed game admits an NE. Thus, we just need to show that in our case the Lemke's algorithm is complete. To show this property we use Theorem 5.2. The first condition of Theorem 5.2 requires:

$$\tilde{\mathbf{x}}_1^T(-R_1^{-T}(\epsilon)U_1R_2^{-1}(\epsilon) - R_1^{-T}(\epsilon)U_2R_2^{-1}(\epsilon))\tilde{\mathbf{x}}_2 \geq 0 \quad \forall \tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2 \geq \mathbf{0}$$

We can rewrite the condition as $\tilde{\mathbf{x}}_1^T R_1^{-T}(\epsilon)(-U_1 - U_2)R_2^{-1}(\epsilon)\tilde{\mathbf{x}}_2 \geq 0$ and we note that $\tilde{\mathbf{x}}_1^T R_1^{-T}(\epsilon)$ and $R_2^{-1}(\epsilon)\tilde{\mathbf{x}}_2$ are positive by construction (a positive vector multiply by a positive matrix), so when U_1 and U_2 are negative the first condition of Theorem 5.2 is satisfied.

The second condition of Theorem 5.2 requires that when $\tilde{\mathbf{x}}_i = \mathbf{0}$ and $\mathbf{v}_i^+ = \mathbf{v}_i^-$, $M\mathbf{z} \geq \mathbf{0}$. It can be easily seen that such a condition is always satisfied. This completes the proof. □

Given that any linear transformation of the utility matrix of the agents returns a game that is equivalent to the original one, we can always subtract a constant from all the payoffs of an agent to obtain a negative utility matrix. Therefore, after an opportune transformation, we can apply the Lemke's algorithm to any two-player game.

Example 5.3 *Consider the game depicted in Fig. 2.1. We subtract a constant to matrices U_1 and U_2 to make them negative:*

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & -2 \end{bmatrix}$$

Now, we report matrices $R_1^{-T}(\epsilon)U_1R_2^{-1}(\epsilon)$ *and* $R_2^{-T}(\epsilon)U_2R_1^{-1}(\epsilon)$*:*

$$U_1 = \begin{bmatrix} -\epsilon - 4\epsilon^2 & -\epsilon^2 & -2\epsilon^2 \\ -1 & 0 & 0 \\ -4\epsilon & -\epsilon & -2\epsilon \\ -1 & 0 & 0 \\ -3\epsilon & -1 & -2 \end{bmatrix}$$

$$U_2 = \begin{bmatrix} -\epsilon - \epsilon^2 - 3\epsilon^3 & -1 & -\epsilon - 3\epsilon^2 & -1 & -3\epsilon \\ -\epsilon^2 & 0 & -\epsilon & 0 & -1 \\ -2\epsilon^2 & 0 & -2\epsilon & 0 & -2 \end{bmatrix}$$

From these results, we can state the following theorem.

Theorem 5.4 *Given a specific perturbation over the behavioral strategies, the computation of the EFPE with two agents is in* $\mathcal{PPAD}$*.*

Corollary 5.1 *Given a specific perturbation over the behavioral strategies, the problem of computing an EFPE with two agents is* $\mathcal{PPAD}$*-complete.*

Proof. Being an EFPE a refinement of NE admitted by every extensive-form game, the problem of computing an EFPE cannot be easier than the problem of computing a NE. The problem of computing a NE is $\mathcal{PPAD}$-complete [16]. Thus the problem of computing an EFPE is at least $\mathcal{PPAD}$-hard. Thus, it is $\mathcal{PPAD}$-hard and it is in $\mathcal{PPAD}$ (Th. 5.4), consequently it is $\mathcal{PPAD}$-complete. □

The algorithm presented in the previous section finds an EFPE given a uniform perturbation ϵ over the behavioral strategy $\sigma_i(a)$ for all the actions a. In order to find an EFPE with a generic perturbation over the behavioral strategy, it is sufficient to change the multiplicative perturbation as follows. Given $l_i(a, \epsilon)$, the perturbation over $q' = q|a$ is $x_i(q') \geq l_i(a, \epsilon) \cdot x_i(q)$.

Example 5.4 *Given the perturbation*

$$\mathbf{l}_1(\epsilon) = \begin{bmatrix} 4\epsilon^2 \\ 2\epsilon \\ \epsilon \\ 3\epsilon^3 \end{bmatrix}, \qquad \mathbf{l}_2(\epsilon) = \begin{bmatrix} \epsilon^2 \\ 3\epsilon \end{bmatrix}$$

over the actions, matrices $R_1(\epsilon)$ and $R_2(\epsilon)$ in the game depicted in Fig. 2.1 are the following:

$$R_1(\epsilon) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -4\epsilon^2 & 1 & 0 & 0 & 0 \\ -2\epsilon & 0 & 1 & 0 & 0 \\ 0 & 0 & -\epsilon & 1 & 0 \\ 0 & 0 & -3\epsilon^3 & 0 & 1 \end{bmatrix}, \qquad R_2(\epsilon) = \begin{bmatrix} 1 & 0 & 0 \\ -\epsilon^2 & 1 & 0 \\ -3\epsilon & 0 & 1 \end{bmatrix}$$

Finally, we remark that, once applied a generic perturbation over the behavioral strategies in formulation (5.1)-(5.4), our result can be exploited to design an algorithm to verify whether a given solution is an EFPE with two-player game. However, its exploitation does not seem straightforward and it remains open whether or not the verification problem is tractable.

5.5 Experimental evalutation

In this section we describe the implementation of the presented algorithm, the experimental setting over we test our algorithm and the results of the tests.

Algorithm implementation

The implementation of the EFPE algorithm is not trivial and requires to drive a careful software design analysis. The choice of the programming language and the underlying structures affects the overall performances of the algorithm.

The choice of the programming language has been mainly guided by the need for efficiency. Despite an higher prototyping time compared to script languages (e.g., MATLAB®or Python), C++ language offers in general an overall speed, either real or perceived, greater than the other programming languages. In addition, we can exploit the Object Oriented features to model the required elements: tableau and polynomials.

Recall that the tableau is not planar but it has a three dimensional structure because each cell represents a polynomial in ϵ. The simplest implementation corresponds to model the tableau has a simple three dimensional matrix. However, this choice is very inefficient. First, cells may contains polynomial of different degree, resulting in

a very sparse matrix. Second, at every pivoting step the degree of the polynomial may increase. Practically this corresponds to resize the matrix introducing new additional components in the direction corresponding to the degree of the polynomial. This operation is usually inefficient both in time and space because increments the dimension of every cell while only few polynomial are really involved in the operation. While the first problem can be overcame using sparse matrix, the second one is related to the programming language or, even worse, to the library used.

For these reasons, we have chosen to model the third dimension (polynomial degree) of the tableau independently for each cell. This goal has been achieved using a sparse matrix for the planar structure of the tableau and a class representing a polynomial for the third dimension. This means that each cell of the two dimensional matrix stores an instance of the class that represents the polynomial.

For the implementation of the sparse matrix we have relied on the Eigen library [31]. Eigen is fast, efficient and versatile. In particular, the main advantage of Eigen library is its template pattern design, that allows to use standard library functions is combination with user defined numerical types. Only few constraints on the interface between user numerical type and Eigen library must be satisfied, involving operators overloading[1] in the user designed numerical type. Eigen sparse matrix implementation support both row and column major indexing, a wide set of decomposition techniques and matrix operations, while maintaining a user friendly design.

The other component we are going to describe is the class representing the polynomial. The best way in order to represent a polynomial is to store its coefficients. The order in which polynomial are stored affects the performance of the algorithm. Let $p(\epsilon)$ be a polynomial of degree d, such that $p(\epsilon) = \sum_{i=0}^{d} a_i \epsilon^{d-i}$. We store the coefficient in an list according to increasing degree:

$$p = \begin{bmatrix} a_0 & a_1 & \dots & a_{d-1} & a_d \end{bmatrix}$$

This representation allows efficient operations between polynomials because changes of the degree of the polynomial corresponds to the addition or remove of a component to the tail of the list, without any copy of the elements.

[1] Operator overloading is a C++ feature, not all the programming languages support it. Reader may refer to [17] for details.

From a practical perspective, the Lemke's algorithm suffers of numerical instability. To avoid this problem we have exploited a library for arbitrary precision arithmetic. Several alternatives are available for C++ language, among them we have decided to use the standard GNU Multiple Precision (GMP) arithmetic library. The main advantages of GMP library are the rich set of functions and the interface for both C and C++ language. In addition, the C++ version of GMP have been designed using operator overloading, that allows a simpler and faster implementation, e.g., it is possible to write a single code and interchange GMP with standard types. This feature allows the development of a template interface for the polynomial class. In this way, without any change of the class, we have been able to test the algorithm with and without arbitrary numerical precision (in order to verify instability issues), and to test other arithmetic libraries.

The last step consists of an efficient implementation of the numerical operations [2] between polynomials.

Experimental setting

We evaluated our algorithm over an experimental setting of extensive-form games. We conducted the experiments on a UNIX computer with dual quad-core 2.33GHz CPUs and 16GB RAM. We generated 20 instances of game for each combination of the following parameters:

- $l \in \{2, \ldots, 9\}$: the number of levels of the tree,
- $b \in \{2, 3\}$: the branching factor, i.e. the number of actions available to each node,
- $\beta \in \{0, 0.5, 1\}$: the unbalance factor. If $\beta = 0$ the tree is balanced, i.e. all the terminal sequences have the same length, if $\beta = 1$ the length of a sequence is arbitrary,
- $\nu \in \{0, 0.5, 0.8\}$: the uncertainty factor. If $\nu = 0$ the game is with perfect information, if $\nu = 1$ the game is with imperfect information and it has the smaller number of information sets compatibly with the perfect recall constraints,

[2] All the standard operation $\{+, -, *, /\}$ are required by the algorithm.

- $\lambda \in \{0, 0.5, 1\}$: the alternating factor. If $\lambda = 0$ the player who plays at each node is randomly chosen, if $\lambda = 1$ it is an alternating game, i.e. after an action undertaken by agent 1, there is one undertaken by agent 2.

Experimental results

In Tab. 5.1-5.9 we summarize the results about the computation of EFPE. For each combination of parameter (l, b and λ) we report the average value of dimension of game (number of sequences and information sets), the average number of pivoting steps (p.s.), the average pivoting time (p.t.) and the number of instances such that the algorithm is not able to solve in an interval of 30 minutes. For the sake of clarity, we organise the tables of results as follows:

5.1: $\beta = 0.0, \nu = 0.0$	5.2: $\beta = 0.0, \nu = 0.5$	5.3: $\beta = 0.0, \nu = 0.8$
5.4: $\beta = 0.5, \nu = 0.0$	5.5: $\beta = 0.5, \nu = 0.5$	5.6: $\beta = 0.5, \nu = 0.8$
5.7: $\beta = 1.0, \nu = 0.0$	5.8: $\beta = 1.0, \nu = 0.5$	5.9: $\beta = 1.0, \nu = 0.8$

The rows corresponding to combinations of parameters such that the algorithm did not solve any instance are not present in the tables.

As we can see in the following tables, the most influencing parameter is β because a tree, the greater is unbalanced, the smaller is the number of information sets and sequences. With $\beta = 1$ the algorithm can solve games with $l = 9$ and $b = 3$ in about 1 second against $l = 8$ and $b = 2$ in more than 1000 second with $\beta = 0$. The parameter ν is more influential in games with small λ because if a tree is an alternating game the number of the information sets is greater than a game where the agents play some actions consecutively. In Tab. 5.3 we can see that in rows with same values of l and b, when $\lambda = 1$ the games are smaller than games where $\lambda = 0$ and the number of instances solved is greater. Finally we underline that the algorithm solve in 30 minutes only small instances of game, with branching factor in $\{2, 3\}$ and level less than 10 and we can see in all the tables that the pivoting steps and pivoting time increase exponentially in the size of game.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	4	4	2	2	11	0.024	0
2	2	0.5	3	4	2	2	11	0.022	0
2	2	1.0	3	5	2	3	10	0.020	0
2	3	0.0	6	7	2	3	15	0.062	0
2	3	0.5	4	9	2	3	14	0.068	0
2	3	1.0	4	10	2	4	12	0.049	0
3	2	0.0	8	7	4	4	22	0.189	0
3	2	0.5	9	6	5	3	23	0.209	0
3	2	1.0	11	5	6	3	21	0.154	0
3	3	0.0	21	19	7	7	45	2.085	0
4	2	0.0	16	15	8	8	49	2.364	0
4	2	0.5	12	19	6	10	47	2.236	0
4	2	1.0	11	21	6	11	49	1.815	0
4	3	0.0	61	60	21	20	163	216.387	1
4	3	0.5	43	78	15	26	177	220.908	0
4	3	1.0	31	91	11	31	147	77.001	0
5	2	0.0	34	29	17	15	107	43.204	2
5	2	0.5	35	28	18	14	116	41.651	4
5	2	1.0	43	21	22	11	96	18.934	0
6	2	0.0	63	64	32	32	249	959.289	4
6	2	0.5	54	73	27	37	232	801.457	4
6	2	1.0	43	85	22	43	234	475.285	1

Table 5.1: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 0$ and $\nu = 0$. The values are the average over the solved instances.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	4	3	2	2	11	0.026	0
2	2	0.5	3	3	2	2	11	0.047	0
2	2	1.0	3	3	2	2	9	0.013	0
2	3	0.0	5	6	2	2	14	0.053	0
2	3	0.5	5	7	2	3	14	0.052	0
2	3	1.0	4	6	2	2	11	0.026	0
3	2	0.0	6	7	3	4	20	0.152	0
3	2	0.5	8	5	4	3	20	0.149	0
3	2	1.0	8	3	4	2	17	0.092	0
3	3	0.0	16	19	6	7	39	1.672	0
3	3	0.5	22	10	8	4	36	1.259	0
4	2	0.0	12	13	6	7	42	1.281	2
4	2	0.5	12	14	6	7	43	1.392	0
4	2	1.0	8	14	4	7	36	0.723	0
4	3	0.0	46	52	16	18	129	83.261	2
4	3	1.0	20	56	7	19	106	21.244	1
5	2	0.0	24	25	12	13	90	23.624	1
5	2	0.5	29	21	15	11	94	20.515	0
5	2	1.0	28	13	14	7	65	5.603	1
5	3	0.0	132	156	44	52	176	1097.451	17
5	3	0.5	140	107	47	36	209	771.270	18
6	2	0.0	51	49	26	25	203	294.108	5
6	2	0.5	41	55	21	28	224	395.527	3
6	2	1.0	26	51	13	26	159	98.257	1
7	2	0.0	91	101	46	51	248	1181.520	19
7	2	0.5	91	87	46	44	237	1003.251	17
7	2	1.0	104	53	52	27	306	1044.155	7

Table 5.2: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 0$ and $\nu = 0.5$. The values are the average over the solved instances.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	4	3	2	2	11	0.023	0
2	2	0.5	3	3	2	2	10	0.017	0
2	2	1.0	3	3	2	2	9	0.011	0
2	3	0.0	5	5	2	2	12	0.046	0
2	3	0.5	5	6	2	2	13	0.043	0
2	3	1.0	4	5	2	2	10	0.018	0
3	2	0.0	6	7	3	4	20	0.197	0
3	2	0.5	7	5	4	3	18	0.122	0
3	2	1.0	7	3	4	2	14	0.045	0
3	3	1.0	16	4	6	2	23	0.316	0
4	2	0.0	14	9	7	5	41	1.233	3
4	2	0.5	9	12	5	6	38	0.909	0
4	2	1.0	7	10	4	5	27	0.351	0
4	3	0.0	36	44	12	15	120	62.890	2
5	2	0.0	22	23	11	12	79	14.480	2
5	2	0.5	23	19	12	10	83	15.872	1
5	2	1.0	20	11	10	6	49	2.541	0
5	3	0.0	109	106	37	36	252	819.849	19
5	3	0.5	115	67	39	23	370	786.322	19
5	3	1.0	104	33	35	11	190	396.780	1
6	2	0.0	40	40	20	20	193	231.984	4
6	2	0.5	34	40	17	20	192	229.734	1
6	2	1.0	19	36	10	18	110	36.793	0
7	2	0.5	72	60	36	30	295	988.529	12
7	2	1.0	68	34	34	17	179	332.729	1
8	2	0.5	135	151	68	76	178	1266.980	19

Table 5.3: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 0$ and $\nu = 0.8$. The values are the average over the solved instances.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	3	3	2	2	9	0.012	0
2	2	0.5	2	3	1	2	9	0.012	0
2	2	1.0	3	3	2	2	9	0.013	0
2	3	0.0	4	4	2	2	10	0.020	0
2	3	0.5	4	5	2	2	10	0.023	0
2	3	1.0	4	6	2	2	10	0.026	0
3	2	0.0	6	5	3	3	19	0.118	0
3	2	0.5	6	5	3	3	17	0.105	0
3	2	1.0	7	4	4	2	17	0.077	0
3	3	0.5	16	12	6	4	30	0.653	0
4	2	0.0	11	12	6	6	39	1.273	1
4	2	0.5	12	11	6	6	37	0.955	0
4	2	1.0	9	13	5	7	43	1.014	1
4	3	0.0	43	33	15	11	104	38.478	0
4	3	0.5	34	46	12	16	105	32.749	1
5	2	0.0	21	22	11	11	79	11.585	1
5	2	0.5	27	19	14	10	98	21.204	2
5	2	1.0	28	19	14	10	89	11.823	1
6	2	0.0	43	46	22	23	208	382.072	1
6	2	0.5	42	51	21	26	230	450.788	2
6	2	1.0	38	49	19	25	182	173.494	7

Table 5.4: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 0.5$ and $\nu = 0$. The values are the average over the solved instances.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	3	3	2	2	9	0.014	0
2	2	0.5	3	3	2	2	9	0.013	0
2	2	1.0	3	3	2	2	9	0.012	0
2	3	0.0	4	5	2	2	11	0.033	0
2	3	0.5	4	5	2	2	10	0.026	0
2	3	1.0	4	4	2	2	10	0.016	0
3	2	0.0	6	5	3	3	17	0.091	1
3	2	0.5	6	4	3	2	16	0.092	1
3	2	1.0	6	3	3	2	14	0.048	1
3	3	0.0	12	10	4	4	33	0.706	1
4	2	0.0	8	10	4	5	36	0.666	0
4	2	0.5	10	10	5	5	33	0.652	1
4	2	1.0	7	9	4	5	28	0.884	2
4	3	0.5	25	40	9	14	104	26.750	2
5	2	0.0	20	19	10	10	92	14.787	3
5	2	0.5	20	17	10	9	75	8.010	2
5	2	1.0	19	11	10	6	54	2.712	3
5	3	0.0	83	95	28	32	260	747.052	15
5	3	0.5	94	77	32	26	326	1015.636	11
6	2	0.0	36	40	18	20	195	210.224	4
6	2	0.5	33	35	17	18	169	135.030	3
6	2	1.0	24	36	12	18	143	62.223	2
7	2	0.0	75	66	38	33	397	1464.860	18
7	2	0.5	80	63	40	32	339	1301.844	17
7	2	1.0	73	46	37	23	357	881.540	11

Table 5.5: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 0.5$ and $\nu = 0.5$. The values are the average over the solved instances.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	3	3	2	2	9	0.017	0
2	2	0.5	2	3	1	2	9	0.013	0
2	2	1.0	3	3	2	2	9	0.011	0
2	3	0.0	5	4	2	2	11	0.031	0
2	3	0.5	4	4	2	2	10	0.025	0
2	3	1.0	4	4	2	2	10	0.016	0
3	2	0.0	4	5	2	3	14	0.065	1
3	2	0.5	5	4	3	2	14	0.051	0
3	2	1.0	6	3	3	2	14	0.050	0
3	3	0.5	12	9	4	3	25	0.323	0
3	3	1.0	12	5	4	2	19	0.154	2
4	2	0.0	9	9	5	5	34	0.696	0
4	2	0.5	8	7	4	4	28	0.358	1
4	2	1.0	6	7	3	4	23	0.210	0
4	3	1.0	16	24	6	8	61	4.261	2
5	2	0.0	19	14	10	7	68	7.945	4
5	2	0.5	17	15	9	8	66	4.850	4
5	2	1.0	17	8	9	4	46	1.837	1
5	3	0.5	89	51	30	17	293	834.075	9
6	2	0.0	33	30	17	15	193	145.079	2
6	2	0.5	29	31	15	16	192	132.610	5
6	2	1.0	20	26	10	13	127	33.098	1
7	2	0.0	51	58	26	29	238	688.586	11
7	2	0.5	51	54	26	27	291	794.964	12
7	2	1.0	53	29	27	15	284	403.570	9
8	2	0.0	121	91	61	46	194	1129.010	19
8	2	0.5	101	81	51	41	230	1765.240	19

Table 5.6: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 0.5$ and $\nu = 0.8$. The values are the average over the solved instances.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	3	2	2	1	8	0.010	0
2	2	0.5	2	3	1	2	8	0.010	0
2	2	1.0	3	3	2	2	9	0.011	0
2	3	0.0	3	4	1	2	9	0.014	0
2	3	0.5	3	4	1	2	9	0.014	0
2	3	1.0	4	4	2	2	8	0.013	0
3	2	0.0	3	4	2	2	11	0.024	0
3	2	0.5	4	3	2	2	11	0.026	0
3	2	1.0	5	3	3	2	13	0.040	0
3	3	0.0	3	7	1	3	12	0.032	0
3	3	0.5	6	4	2	2	12	0.034	0
4	2	0.0	5	4	3	2	14	0.052	0
4	2	0.5	4	5	2	3	15	0.050	0
4	2	1.0	5	5	3	3	15	0.056	0
4	3	0.5	6	7	2	3	17	0.080	0
5	2	0.0	6	5	3	3	18	0.107	0
5	2	0.5	5	6	3	3	18	0.119	0
5	2	1.0	7	5	4	3	19	0.112	0
5	3	0.5	8	8	3	3	20	0.166	0
6	2	0.0	6	7	3	4	20	0.178	0
6	2	0.5	6	7	3	4	24	0.216	0
6	2	1.0	7	7	4	4	23	0.184	0
6	3	1.0	10	10	4	4	24	0.262	0
7	2	0.0	8	8	4	4	26	0.304	0
7	2	0.5	8	7	4	4	24	0.270	0
7	2	1.0	9	7	5	4	25	0.246	0
7	3	0.0	10	12	4	4	26	0.427	0
8	2	0.0	8	9	4	5	30	0.496	0
8	2	0.5	9	8	5	4	33	0.519	0
8	2	1.0	9	9	5	5	28	0.422	0
8	3	0.0	10	15	4	5	34	0.729	0
8	3	1.0	13	13	5	5	33	0.769	0
9	2	0.0	9	10	5	5	36	0.876	0
9	2	0.5	9	10	5	5	34	0.743	0
9	2	1.0	11	9	6	5	31	0.651	0
9	3	0.0	13	15	5	5	33	0.932	0
9	3	0.5	15	13	5	5	35	1.011	1
9	3	1.0	16	13	6	5	37	1.142	0

Table 5.7: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 1$ and $\nu = 0$. The values are the average over the solved instances.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	3	2	2	1	8	0.010	0
2	2	0.5	2	3	1	2	8	0.011	0
2	2	1.0	3	3	2	2	9	0.011	0
2	3	0.0	4	3	2	1	9	0.014	0
2	3	0.5	4	3	2	1	8	0.013	0
2	3	1.0	4	4	2	2	9	0.014	0
3	2	0.0	4	3	2	2	11	0.028	0
3	2	0.5	4	3	2	2	11	0.025	0
3	2	1.0	5	3	3	2	12	0.028	0
3	3	1.0	7	4	3	2	13	0.035	0
4	2	0.0	5	5	3	3	15	0.053	0
4	2	0.5	5	4	3	2	16	0.073	0
4	2	1.0	5	5	3	3	15	0.053	0
4	3	0.5	6	7	2	3	16	0.098	0
4	3	1.0	7	7	3	3	17	0.091	0
5	2	0.0	6	5	3	3	21	0.122	0
5	2	0.5	6	5	3	3	20	0.115	0
5	2	1.0	7	5	4	3	18	0.118	0
6	2	0.0	7	6	4	3	23	0.226	0
6	2	0.5	7	7	4	4	23	0.172	0
6	2	1.0	7	7	4	4	23	0.188	0
6	3	0.0	9	10	3	4	24	0.306	0
6	3	0.5	9	10	3	4	25	0.270	0
6	3	1.0	10	10	4	4	24	0.247	1
7	2	0.0	8	7	4	4	25	0.268	0
7	2	0.5	8	7	4	4	24	0.280	0
7	2	1.0	9	7	5	4	26	0.306	0
7	3	0.0	11	11	4	4	29	0.448	0
7	3	0.5	11	11	4	4	25	0.414	0
8	2	0.0	8	9	4	5	29	0.485	0
8	2	0.5	8	9	4	5	30	0.462	0
8	2	1.0	9	9	5	5	29	0.440	0
8	3	0.0	12	13	4	5	32	0.682	0
8	3	0.5	11	14	4	5	32	0.730	0
9	2	0.0	9	10	5	5	33	0.679	0
9	2	0.5	10	9	5	5	34	0.724	0
9	2	1.0	11	9	6	5	30	0.575	0
9	3	0.0	15	13	5	5	37	1.166	0

Table 5.8: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 1$ and $\nu = 0.5$. The values are the average over the solved instances.

l	b	λ	$\lvert S_1\rvert$	$\lvert S_2\rvert$	$\lvert H_1\rvert$	$\lvert H_2\rvert$	p.s.	p.t.	$> 30'$
2	2	0.0	2	3	1	2	9	0.011	0
2	2	0.5	3	3	2	2	8	0.010	0
2	2	1.0	3	3	2	2	9	0.010	0
2	3	0.0	3	4	1	2	9	0.014	0
2	3	0.5	3	4	1	2	9	0.014	0
2	3	1.0	4	4	2	2	9	0.013	0
3	2	0.0	3	4	2	2	12	0.029	0
3	2	0.5	4	3	2	2	11	0.022	0
3	2	1.0	5	3	3	2	12	0.029	0
3	3	0.0	5	5	2	2	12	0.031	0
3	3	1.0	7	4	3	2	13	0.036	0
4	2	0.0	5	4	3	2	14	0.050	0
4	2	0.5	4	5	2	3	14	0.044	0
4	2	1.0	5	5	3	3	17	0.062	0
4	3	0.5	7	6	3	2	17	0.086	0
5	2	0.0	6	5	3	3	19	0.118	0
5	2	0.5	5	6	3	3	18	0.104	0
5	2	1.0	7	5	4	3	18	0.100	0
5	3	0.5	9	7	3	3	21	0.172	0
6	2	0.0	8	5	4	3	23	0.202	0
6	2	0.5	7	6	4	3	20	0.161	0
6	2	1.0	7	7	4	4	21	0.169	0
6	3	1.0	10	10	4	4	27	0.329	0
7	2	0.0	9	7	5	4	26	0.312	0
7	2	0.5	7	8	4	4	25	0.296	0
7	2	1.0	9	7	5	4	23	0.236	0
7	3	0.0	11	11	4	4	25	0.364	1
7	3	0.5	11	11	4	4	27	0.407	0
8	2	0.0	8	9	4	5	30	0.524	0
8	2	0.5	8	9	4	5	26	0.350	0
8	2	1.0	9	9	5	5	28	0.424	0
9	2	0.0	10	9	5	5	35	0.828	0
9	2	0.5	10	9	5	5	31	0.607	0
9	2	1.0	11	9	6	5	33	0.618	0

Table 5.9: Experimental results about the computation of an EFPE in extensive-form games with parameters $\beta = 1$ and $\nu = 0.8$. The values are the average over the solved instances.

5.6 Related works

We summarize in Tab. 5.10 the main computational complexity results on the problem of computing a strategy profile that is an equilibrium according to some given NE refinement solution concept for extensive-form games.

Solution concept	Input	Two agents
NE	strategies	$\mathcal{PPAD}$ -complete [43]
SPE	strategies	$\mathcal{PPAD}$ -complete
SE	strategies	
NFPE	strategies	$\mathcal{PPAD}$-complete [70]
QPE	strategies and pertubation	$\mathcal{PPAD}$-complete [48]
EFPE	strategies and pertubation	$\mathcal{PPAD}$-complete (*)
NFPrE	strategies and pertubation	$\mathcal{PPAD}$-complete [62]
EFPrE	strategies and pertubation	

Table 5.10: Computational complexity results for equilibrium computation; all the strategies are in sequence form; the results with '(*)' are originally provided in the present thesis.

NE. The sequence form is the most efficient representation to compute an NE (normal form is exponentially larger, while agent form poses highly non-linear constraints over the best response optimization problem). The computation of an NE with two-agent zero-sum games can be easily tackled by resorting to linear programming, computing a maxmin/minmax strategy. Instead, solving general-sum games is harder. The main results on the computation of an NE are with two agents. The problem to search for an NE is formulated as a LCP and solved by employing the Lemke's algorithm.[3] The LCP formulation is:

[3]While the Lemke-Howson algorithm cannot be used with the sequence form of an extensive-form game, it can be used with the normal form.

$$F_i \mathbf{x}_i \geq \mathbf{f}_i \qquad \forall i \in \{1,2\} \tag{5.5}$$
$$-F_i \mathbf{x}_i \geq -\mathbf{f}_i \qquad \forall i \in \{1,2\} \tag{5.6}$$
$$F_i^T \mathbf{v}_i^+ - F_i^T \mathbf{v}_i^- - U_i \mathbf{x}_{-i} \geq \mathbf{0} \qquad \forall i \in \{1,2\} \tag{5.7}$$
$$\mathbf{x}_i \geq \mathbf{0} \qquad \forall i \in \{1,2\} \tag{5.8}$$
$$\mathbf{v}_i^+ \geq \mathbf{0} \qquad \forall i \in \{1,2\} \tag{5.9}$$
$$\mathbf{v}_i^- \geq \mathbf{0} \qquad \forall i \in \{1,2\} \tag{5.10}$$
$$\mathbf{x}_i^T \cdot (F_i^T \mathbf{v}_i^+ - F_i^T \mathbf{v}^- - U_i \mathbf{x}_{-i}) = 0 \qquad \forall i \in \{1,2\} \tag{5.11}$$

where $\mathbf{v}_i$ is the vector of the variables associated to the dual best response problem of agent i, $\mathbf{v}_i^+$ and $\mathbf{v}_i^-$ are defined as $\mathbf{v}_i = \mathbf{v}_i^+ - \mathbf{v}_i^-$, and matrices U_is are non-positive. The LCP formulation described in constraints (5.5)-(5.11) can be solved with the Lemke's algorithm by using:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{v}_1^+ \\ \mathbf{v}_1^- \\ \mathbf{v}_2^+ \\ \mathbf{v}_2^- \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_1 \\ -\mathbf{f}_1 \\ \mathbf{f}_2 \\ -\mathbf{f}_2 \end{bmatrix}$$

$$M = \begin{bmatrix} 0 & -U_1 & F_1^T & -F_1^T & 0 & 0 \\ -U_2^T & 0 & 0 & 0 & F_2^T & -F_2^T \\ -F_1 & 0 & 0 & 0 & 0 & 0 \\ F_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -F_2 & 0 & 0 & 0 & 0 \\ 0 & F_2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and forcing U_1 and U_2 to be non-strictly negative by subtracting a constant to each entry of the matrices associated with a pair of terminal sequences. An alternative LCP approach to compute an NE is proposed in [70].

Searching for an NE is not $\mathcal{NP}$-complete unless co-$\mathcal{NP}$ = $\mathcal{NP}$ [16]. The appropriate class is $\mathcal{PPAD}$-complete. It is not known whether or not there is a polynomial time algorithm to compute an NE, but it is commonly believed it is not.

Two approaches, alternative to LCP, can be found for two-agent strategic-form games. The first approach is based on support enumeration (the support is the set of actions played with strictly pos-

itive probability) and heuristics. Although this approach performs well with strategic-form games, its application with extensive-form games appears intractable (as preliminary shown in [11] for Bayesian games). The second approach is based on mixed-integer linear programming (MILP). This approach is unexplored with extensive-form games. In the case of strategic-form games, the MILP approach performs worse than the other algorithms (i.e., support enumeration based and LCP) for the search for an NE, but better for the search for an optimal NE.[4] With more than two agents, no result is known except those on the computation of an NE for strategic-form games (see [59], for a detailed survey).

SPE. The computation of an SPE can be done by combining backward induction together with the computation of NEs. As a result, the complexity is $\mathcal{PPAD}$-complete.

SE. The computation of an SE is open [32]. The computation of an SE with two agents can be done by computing a QPE (see below), but there is no algorithm to find a generic SE.

QPE. In order to find a QPE, we can impose a perturbation $\mathbf{x}_i(\epsilon) \geq \mathbf{l}_i(\epsilon)$ where $\mathbf{l}_i(\epsilon)$ is defined in Section 2.3. Practically, we substitute $\mathbf{x}_i$ with $\tilde{\mathbf{x}}_i(\epsilon) + \mathbf{l}_i(\epsilon)$ posing the constraint $\tilde{\mathbf{x}}_i(\epsilon) \geq 0$. Define $\boldsymbol{\epsilon} = [\epsilon^1 \quad \epsilon^2 \quad \dots \quad \epsilon^{|\overline{q}|}]$ where $\overline{q}$ is the longest sequence over all the sequences (where the length of a sequence q is the number of actions a composing the sequence). Vectors $\mathbf{l}_i(\epsilon)$ can be written as $L_i\boldsymbol{\epsilon}$ where L_i is matrix $|Q_i| \times |\overline{q}|$. By substituting $\mathbf{x}_i$ with $\tilde{\mathbf{x}}_i + L_i\boldsymbol{\epsilon}$ we obtain the following tableau:

	$\mathbf{w}^T$	$\mathbf{z}^T$	z_0		$\boldsymbol{\epsilon}^T$	$\boldsymbol{\delta}^T$
$\mathbf{w}$	I	$-M$	$-\mathbf{d}$	$\mathbf{b}$	E	Y

where

$$E = \begin{bmatrix} U_1 L_1 \\ U_2^T L_2 \\ -F_1 L_1 \\ F_1 L_1 \\ -F_2 L_2 \\ F_2 L_2 \end{bmatrix}$$

[4] With LCP algorithms, an optimal equilibrium can be found enumerating all the equilibria, while with support enumeration, the enumeration of all the supports is needed.

The lexico minimum ratio test will consider at first the entries of E and subsequently those of Y, as follows:

$$\arg \operatorname{lex} \min_k \left\{ \frac{[b_k \;\; e_{k,1} \;\; \dots \;\; e_{k,n'} \;\; y_{k,1} \;\; \dots \;\; y_{k,n}]}{t_{k,j}} : t_{k,j} > 0 \right\}$$

where $e_{k,j}$ is an entry of E and j is the column of the entering variable. In [48] the authors use the Lemke's algorithm with perturbations $\mathbf{l}_1(\epsilon)$, $\mathbf{l}_2(\epsilon)$ with $l_i(q, \epsilon) = \epsilon^{|q|}$ where $|q|$ is the length of sequence q to compute a QPE when agents are two. Due to numerical stability problems, ϵ must be a symbolic value and it is used to break lexicographically the ties in the minimum-ratio test. This shows that the problem to compute a QPE with two agents is in $\mathcal{PPAD}$ when a specific perturbation is given.

Perfection based equilibria. With two agents, an NFPE can be computed either by employing the algorithm presented in [70] with a starting point that is a fully mixed strategy or by searching for an NE once weakly dominated strategies are removed (this last task can be accomplished in polynomial time). Therefore, the problem of searching for an NFPE is $\mathcal{PPAD}$-complete. The problem of computing an EFPE without an input perturbation over the strategies is open. In particular, it is not known whether this problem can or cannot be formulated with the sequence form. In the latter case, the problem to compute an EFPE would require non-linear optimization tools and, the problem not being convex, no exact solution could be computed.

Properness based equilibria. The problem of computing a NFPrE with two agents is in $\mathcal{PPAD}$ and can be performed by using the Lemke's algorithm after a specific transformation [62]. However, it is not known whether such a transformation can be opportunely modified to be applied to the sequence form and/or to find an EFPrE.

Computing equilibria in two-player zero-sum continuous stochastic games with switching controller 6

In this chapter, we focus on continuous stochastic games with switching control. The literature only provides results for finite stochastic games with switching control [71] and for polynomial continuous stochastic games with single controller [57]. In this chapter we provide the following original contributions

- we provide an iterative SDP based algorithm that converges to an MPE - the appropriate solution concept for stochastic games - when both the reward and the state transitions are polynomial functions and returns an ϵ-approximate MPE (ϵ-MPE) with $\epsilon \leq \bar{\epsilon}$, where $\bar{\epsilon}$ is given as input,
- we use our algorithm to approximate solutions of non-polynomial games: approximating the reward and state transitions given as input with polynomials, solving the approximated game, and providing theoretical upper bounds on the quality of the solutions as functions of the approximation error with infinity norm,
- we experimentally evaluate the performance of our algorithm in terms of iterations and computational time.

In Section 6.1 we describe the algorithm converging to an MPE with polynomial games. In Section 6.2 we give theoretical bounds of approximation errors of our algorithm with non-polynomial games. Finally, in Section 6.3 we experimental evaluate our algorithm.

6.1 Equilibrium computation with polynomial games

Here, we describe the algorithm converging to an MPE with polynomial games. For the sake of clarity, we initially describe the algorithm omitting details on the SDPs the algorithm uses. Details are provided later.

Algorithm

The procedure is summarized in Algorithm 5. The algorithm uses auxiliary utilities $\hat{\mathbf{u}}$. As shown below, these utilities converge to the utilities at the equilibrium, denoted by $\mathbf{u}^*$.

Initially, (Steps 1 and 2) the algorithm initializes $\hat{u}_s = 0$ for every $s \in S$. Then, the algorithm repeats Steps 3-11 until an $\overline{\epsilon}$-MPE has been found where $\overline{\epsilon}$ is given as input.

Algorithm 5 Iterative Nash approximation

1: assign $\hat{u}_s = 0$ for every $s \in S$
2: **repeat**
3: $[\hat{\mathbf{u}}, \boldsymbol{\sigma}_2^{S_1}]$ = solve $\mathrm{PS}_1(\hat{\mathbf{u}})$
4: $[\ \ , \boldsymbol{\sigma}_1^{S_1}]$ = solve $\mathrm{DS}_1(\hat{\mathbf{u}})$
5: $[\hat{\mathbf{u}}, \boldsymbol{\sigma}_1^{S_2}]$ = solve $\mathrm{PS}_2(\hat{\mathbf{u}})$
6: $[\ \ , \boldsymbol{\sigma}_2^{S_2}]$ = solve $\mathrm{DS}_2(\hat{\mathbf{u}})$
7: assign $\overline{\boldsymbol{\sigma}}_1 = (\boldsymbol{\sigma}_1^{S_1}, \boldsymbol{\sigma}_1^{S_2})$ and $\overline{\boldsymbol{\sigma}}_2 = (\boldsymbol{\sigma}_2^{S_1}, \boldsymbol{\sigma}_2^{S_2})$
8: calculate $\overline{\mathbf{u}}$ with $(\overline{\boldsymbol{\sigma}}_1, \overline{\boldsymbol{\sigma}}_2)$
9: $\mathbf{u}_1^*$ = solve BR_1 with $\overline{\boldsymbol{\sigma}}_2 = (\boldsymbol{\sigma}_2^{S_1}, \boldsymbol{\sigma}_2^{S_2})$
10: $\mathbf{u}_2^*$ = solve BR_2 with $\overline{\boldsymbol{\sigma}}_1 = (\boldsymbol{\sigma}_1^{S_1}, \boldsymbol{\sigma}_1^{S_2})$
11: **until** $\max\{\|\mathbf{u}_1^* - \overline{\mathbf{u}}\|_\infty, \|\overline{\mathbf{u}} - \mathbf{u}_2^*\|_\infty\} \leq \overline{\epsilon}$

At first, the algorithm finds the optimal strategies in the states S_1 controlled by agent 1 when the utilities of the states $s \in S_2$ are fixed to $\hat{u}_s$, and assigns the returned optimal utility values of states $s \in S_1$ to $\hat{u}_s$. This is accomplished into two steps: in Step 3, the optimal

strategy of agent 2 is computed by solving an SDP called PS_1, while in Step 4, the optimal strategy of agent 1 is computed by solving an SDP called DS_1. PS_1 is the primal problem, while DS_1 is the dual problem. (As we will discuss in the following section, strong duality holds for these two problems.) The problem PS_1:

$$\textbf{(PS}_1\textbf{)} \qquad \min \sum_{s\in S_1} u_s$$

$$\mathbb{E}_{x_2\sim\sigma_{2,s}}\left[r_s(x_1,x_2)\right] + \gamma \sum_{s'\in S_1} u_{s'}\cdot p_{s,s'}(x_1) + \gamma \sum_{s'\in S_2} \hat{u}_{s'}\cdot p_{s,s'}(x_1) \le u_s \qquad \forall s\in S_1, x_1\in X_{1,s} \quad (6.1)$$

$$\sigma_{2,s} \text{ is a probability measure on } X_{2,s} \qquad \forall s\in S_1 \qquad (6.2)$$

where $\mathbb{E}[\cdot]$ denotes the expectation of "$\cdot$". Notice that (6.1) contains infinitely many constraints, one for each value of $x_1 \in X_{1,s}$. PS_1 returns the optimal strategy $\sigma_{2,s}^{S_1}$ and the optimal utilities $\overline{u}_s$ in states $s \in S_1$ given that utilities of states S_2 are fixed and equal to $\hat{u}_s$. Utilities $\overline{u}_s$ are assigned to $\hat{u}_s$ for $s \in S_1$. Similarly, the dual problem DS_1 is:

$$\textbf{(DS}_1\textbf{)} \qquad \max\left(\sum_{s\in S_2} z_s - \gamma\cdot\sum_{s'\in S_1}\mathbb{E}_{x_2\sim\sigma_{2,s}}[p_{s,s'}(x_2)\cdot\hat{u}_{s'}]\right)$$

$$\begin{aligned} \textstyle\sum_{s\in S_2}\mathbb{E}_{x_2\sim\sigma_{2,s}}[\mathbf{1}_{s=s'} - \gamma p_{s,s'}(x_2)] &= 1 \quad \forall s\in S_2 \\ -z_s - \mathbb{E}_{x_2\sim\sigma_{2,s}}[r_s(x_1,x_2)] &\ge 0 \quad \forall s\in S_2, x_1\in X_{1,s} \end{aligned}$$

$$\sigma_{2,s} \text{ is a probability measure on } X_{2,s} \qquad \forall s\in S_2$$

where z_s are auxiliary variables, and $\mathbf{1}_{s=s'}$ is equal to 1 when $s = s'$ and 0 otherwise. As above, DS_1 contains infinite constraints one for each value of $x_2 \in X_{2,s}$. DS_1 returns the optimal strategy $\sigma_{1,s}^{S_1}$ in states S_1.

Then, in Steps 5 and 6, the algorithm repeats Steps 3 and 4 for the states S_2 by solving two SDPs, called PS_2 and DS_2 and omitted here being similar to PS_1 and DS_1, respectively. These programs return

the optimal strategies $\sigma_{1,s}^{S_2}$, $\sigma_{2,s}^{S_2}$ and the optimal utilities $\overline{u}_s$ in states S_2 given $\hat{u}_s$ at $s \in S_1$. Utilities $\overline{u}_s$ are assigned to $\hat{u}_s$ for $s \in S_2$.

In the next steps, the current solution is considered as an ϵ-MPE and the value of ϵ is estimated as follows: given the joint strategy at the current iteration, the algorithm computes the utilities and compares them w.r.t. the utilities provided by each agent's best response. The utilities provided by the current solution $(\overline{\boldsymbol{\sigma}}_1, \overline{\boldsymbol{\sigma}}_2)$ with $\overline{\boldsymbol{\sigma}}_1 = (\boldsymbol{\sigma}_1^{S_1}, \boldsymbol{\sigma}_1^{S_2})$ and $\overline{\boldsymbol{\sigma}}_2 = (\boldsymbol{\sigma}_2^{S_1}, \boldsymbol{\sigma}_2^{S_2})$ can be easily obtained by solving the following linear system (Step 8):

$$\mathbb{E}_{\substack{x_1 \sim \overline{\sigma}_{1,s} \\ x_2 \sim \overline{\sigma}_{2,s}}} \left[r_s(x_1, x_2) + \gamma \sum_{s' \in S} \overline{u}_{s'} \cdot p_{s,s'}(x_1) \right] = \overline{u}_s \quad \forall s \in S_1$$

$$\mathbb{E}_{\substack{x_1 \sim \overline{\sigma}_{1,s} \\ x_2 \sim \overline{\sigma}_{2,s}}} \left[r_s(x_1, x_2) + \gamma \sum_{s' \in S} \overline{u}_{s'} \cdot p_{s,s'}(x_2) \right] = \overline{u}_s \quad \forall s \in S_2$$

The problem of computing agent i's best response given $\overline{\boldsymbol{\sigma}}_{-i}$ is a continuous MDP with polynomial reward and transition functions (and therefore it admits an optimal pure strategy). This problem can be formulated, as shown later, as an SDP. Thus, we have two SDPs, called BR_1 and BR_2 for agent 1 and agent 2, respectively. BR_1 (Step 9) is formulated as:

$$\textbf{(BR}_1\textbf{)} \qquad \min \sum_{s \in S} u_s$$

$$\mathbb{E}_{x_2 \sim \overline{\sigma}_{2,s}} \left[r_s(x_1, x_2) \right] + \gamma \sum_{s' \in S} u_{s'} \cdot p_{s,s'}(x_1) \leq u_s \qquad \forall s \in S_1, x_1 \in X_{1,s} \quad (6.3)$$

$$\mathbb{E}_{x_2 \sim \overline{\sigma}_{2,s}} \left[r_s(x_1, x_2) + \gamma \sum_{s' \in S} u_{s'} \cdot p_{s,s'}(x_2) \right] \leq u_s \qquad \forall s \in S_2, x_1 \in X_{1,s} \quad (6.4)$$

BR_2 (Step 10) is defined similarly and then omitted. Call $\mathbf{u}_1^*$ the vector of utilities returned by BR_1 and $\mathbf{u}_2^*$ the vector of utilities returned by BR_2. Call $\overline{\mathbf{u}}$ the vector of utilities $\overline{u}_s$. The ϵ value of the current strategy profile $(\overline{\boldsymbol{\sigma}}_1, \overline{\boldsymbol{\sigma}}_2)$ is $\max_i \{ \| \mathbf{u}_i^* - \overline{\mathbf{u}} \|_\infty \}$, that is, the maximum

loss of all the agents over all the states. The algorithm terminates if $\epsilon \leq \overline{\epsilon}$.

We can state the following theorem.

Theorem 6.1 *Given polynomial reward and transition functions, Algorithm 5 returns an $\overline{\epsilon}$-MPE.*

Proof. The proposed algorithm needs non-negative reward functions. This assumption can be easily satisfied (without changing the equilibrium strategies) by adding a constant to the reward functions, so that all the rewards get strictly positive. We observe that the solution of the problems PS_1 and PS_2 are monotonically increasing in $\hat{u}_s$. That is, given $\hat{u}'_s$ and $\hat{u}''_s$ such that $\hat{u}'_s \leq \hat{u}''_s \leq u^*_s$, where u^*_s are the utilities at the equilibrium, for every $s \in S$ and called $\overline{u}'_s$ the solution of PS_i when the input is $\hat{u}'_s$ and $\overline{u}''_s$ the solution of PS_i when the input is $\hat{u}''_s$, we have that $\overline{u}'_s \leq \overline{u}''_s \leq u^*_s$. As a result, when reward functions are non-negative, starting from $\hat{u}_s = 0$ for every $s \in S$ we have a sequence of $\hat{u}_s$ that is monotonically increasing as long $\hat{u}^i_s \neq u^*_s$. Therefore, the algorithm converges to an MPE. The algorithm stops when no agent, given the strategy of the opponent, can gain more than $\overline{\epsilon}$. Therefore, the algorithm returns an $\overline{\epsilon}$-MPE. □

Differently from finite switching-control games [71], when games are continuous there is no guarantee that the algorithm converges by a finite number of steps. At each iteration, it is possible to check whether or not there is an MPE with the current support, i.e., the set of actions played with non-zero probability in $(\overline{\boldsymbol{\sigma}}_1, \overline{\boldsymbol{\sigma}}_2)$. (When games are finite, this problem can be formulated as a linear programming problem; it can be easily shown that with continuous polynomial games such problem can be formulated as an SDP.) Since with finite games the number of supports is finite, it is possible to guarantee the termination by finite time. The same approach cannot be used with continuous games, the number of supports being infinite.

Semidefinite programming formulation

We show how PS_1 can be formulated as an SDP (the formulations of DS_1, PS_2, and DS_2 are similar and therefore they will be omitted here). At first, we rewrite constraint (6.1) by considering that r_s and $p_{s,s'}$ are polynomial functions and by expanding the expected value operator $\mathbb{E}[\cdot]$:

$$u_s - \sum_{k=0}^{m}\sum_{j=0}^{m} r_{s,k,j}\cdot(x_1)^k\cdot\left(\int_{X_{2,s}}\sigma_{2,s}(x_2)\cdot(x_2)^j dx_2\right) - \gamma\sum_{s'\in S}u'_s\cdot\sum_{k=0}^{m}p_{s,s',k}\cdot(x_1)^k \geq 0 \qquad \forall s\in S_1, x_1\in X_{1,s}$$

We can substitute $\int_{X_{2,s}}\sigma_{2,s}(x_2)\cdot(x_2)^j dx_2$ with the moment $\mu_{2,s,j}$ of the j-th order of $\sigma_{2,s}$, obtaining:

$$u_s - \sum_{k=0}^{m}\sum_{j=0}^{m} r_{s,k,j}\cdot(x_1)^k\cdot\mu_{2,s,j} - \gamma\sum_{s'\in S}u_{s'}\cdot\sum_{k=0}^{m}p_{s,s',k}\cdot(x_1)^k \geq 0 \qquad \forall s\in S_1, x_1\in X_{1,s}$$

Call $\boldsymbol{\mu}_{2,s}$ the vector of $\mu_{2,s,j}$ with $j\in\{0,\ldots,m\}$ (higher order moments are not constrained). PS_1 is:

$$\textbf{(PS}_1\textbf{)} \qquad \min\sum_{s\in S_1}u_s$$

$$u_s - \sum_{k=0}^{m}\sum_{j=0}^{m} r_{s,k,j}\cdot(x_1)^k\cdot\mu_{2,s,j} - \gamma\sum_{s'\in S}u_{s'}\cdot \cdot\sum_{k=0}^{m}p_{s,s',k}\cdot(x_1)^k \in \mathcal{P}^+(x_1\in X_{1,s}) \qquad \forall s\in S_1$$

$$\boldsymbol{\mu}_{2,s}\in\mathcal{M}(X_{2,s}) \qquad \forall s\in S_1$$

where $\mathcal{P}^+(x_1\in X_{1,s})$ is the space of univariate polynomials in x_1 that are non-negative on $X_{1,s}$ and $\mathcal{M}(X_{1,s})$ is the space of the moment vectors of well defined probability measures on $X_{1,s}$. Both these constraints (i.e., non-negativeness of univariate polynomials and correspondence of moment vectors to well defined probability measures) can be coded as semidefinite programming constraints. Call:

$$\begin{aligned}
\mathbf{u} &= \begin{bmatrix} u_1 & u_2 & \dots & u_{|S|} \end{bmatrix}^T \\
[x]_m &= \begin{bmatrix} (x)^0 & (x)^1 & (x)^2 & \dots & (x)^m \end{bmatrix}^T \\
R_s^T \cdot [x]_m &= \sum_{k=0}^{m} r_{s,k} \cdot (x)^k \\
\mathbf{u}^T \cdot P_{1,s}^T \cdot [x]_m &= \sum_{s' \in S_1} u_{s'} \cdot \sum_{k=0}^{m} p_{s,s',k} \cdot (x)^k \\
\hat{\mathbf{u}}^T \cdot \hat{P}_{1,s}^T \cdot [x]_m &= \sum_{s' \in S_2} \hat{u}_{s'} \cdot \sum_{k=0}^{m} \hat{p}_{s,s',k} \cdot (x)^k
\end{aligned}$$

Call $\mathcal{H}$ the following operator returning an Hankel matrix:

$$\mathcal{H} : \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{2n-1} \end{bmatrix} \longrightarrow \begin{bmatrix} a_1 & a_2 & \dots & a_n \\ a_2 & a_3 & \dots & a_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n+1} & \dots & a_{2n-1} \end{bmatrix}$$

and $\mathcal{H}^*$ the following adjoint operator:

$$\mathcal{H}^* : \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{1,2} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,n} & a_{2,n} & \dots & a_{n,n} \end{bmatrix} \longrightarrow \begin{bmatrix} a_{1,1} \\ 2a_{1,2} \\ a_{2,2} + 2a_{1,3} \\ \vdots \\ a_{n,n} \end{bmatrix}$$

Define the following matrices as:

$$L_1 = \begin{bmatrix} I_{m \times m} \\ \mathbf{0}_{1 \times m} \end{bmatrix}, L_2 = \begin{bmatrix} \mathbf{0}_{1 \times m} \\ I_{m \times m} \end{bmatrix}$$

When $X_{1,s} = [0, 1]$ with $s \in S$, PS_1 can be written as:

$$\min \sum_{s \in S_1} u_s$$

$$\begin{aligned}
&\mathcal{H}^*(Z_s + \frac{1}{2}(L_1 W_s L_2^T + L_2 W_s L_1^T) - L_2 W_s L_2^T) \\
&\quad - u_s - R_s \boldsymbol{\mu}_{i,s} - \gamma(P_{1,s}\mathbf{u} + \hat{P}_{1,s}\hat{\mathbf{u}}) = 0 \qquad \forall s \in S_1 \quad (6.5)
\end{aligned}$$

$$\frac{1}{2}(L_1^T\mathcal{H}(\boldsymbol{\mu}_{i,s})L_2 + L_2^T\mathcal{H}(\boldsymbol{\mu}_{i,s})L_1) - L_2^T\mathcal{H}(\boldsymbol{\mu}_{i,s})L_2 \succeq 0 \quad \forall s \in S_1 \tag{6.6}$$

$$Z_s, W_s \succeq 0 \quad \forall s \in S_1 \tag{6.7}$$
$$\mathcal{H}(\boldsymbol{\mu}_{i,s}) \succeq 0 \quad \forall s \in S_1 \tag{6.8}$$
$$\mu_{i,s,0} = 1 \quad \forall s \in S_1 \tag{6.9}$$
$$\mathcal{H}(\boldsymbol{\mu}'_{i,s}) \succeq 0 \quad \forall s \in S_1 \tag{6.10}$$
$$\mathcal{H}(\boldsymbol{\mu}_{i,s}) - \mathcal{H}(\boldsymbol{\mu}'_{i,s}) \succeq 0 \quad \forall s \in S_1 \tag{6.11}$$

where $\boldsymbol{\mu}'_{i,s} = [\mu_{i,s,1} \quad \mu_{i,s,2} \quad \dots \quad \mu_{i,s,m+1}]$, W_s and Z_s are matrices of auxiliary variables, and "$\succeq 0$" means semidefinite positive. Constraints (6.5) and (6.7) translate constraint (6.1) in SDP fashion; constraints (6.8)-(6.9) translate constraint (6.2); constraints (6.10) and (6.11) are accessory for the resolution of PS_1, but necessary for finding well defined $\mu_{i,s,m+1}$ that are needed for the strategy recovery as described in the following section. The proof that strong duality holds with this SDP easily follows from the satisfaction of Slater's constraint qualification and that it is bounded from below, a similar proof can be found in [57].

Strategy recovery

The SDP programs discussed in the previous sections return strategies defined in the space of moments. In order to recovery the strategy in the space of actions we can use the methods discussed in [36, 54, 60]. Since the number of moments of $\mu_{i,s,h}$ of $\sigma_{i,s}$ is finite, we can always define a finite support $\Psi_{i,s}$, where $x_{i,s,h}$ is the h-th value of $\Psi_{i,s}$. For the sake of presentation, in the following we will omit indices i, s from $\mu_{i,s,h}$, $x_{i,s,h}$, and $\sigma_{i,s}$.

The strategy can be recovered by, at first, solving the following linear equation system:

$$\begin{bmatrix} \mu_0 & \mu_1 & \cdots & \mu_{\frac{m}{2}+1} \\ \mu_1 & \mu_2 & \cdots & \mu_m \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{\frac{m}{2}+1} & \mu_{\frac{m}{2}+2} & \cdots & \mu_m \end{bmatrix} \cdot \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{m-1} \end{bmatrix} = - \begin{bmatrix} \mu_{\frac{m}{2}+1} \\ \mu_{\frac{m}{2}+2} \\ \vdots \\ \mu_{m+1} \end{bmatrix}$$

and finding the vector of coefficients b_j. Notice that the above (Hankel) matrix of the moments is semidefinite positive due to constraint (6.8) and therefore the solution of the above linear system is unique. The actions in the support are the roots x_h of the following univariate polynomial:

$$x^m + b_{m-1} \cdot x^{m-1} + \ldots + b_1 \cdot x + b_0 = 0.$$

The probabilities $\sigma(x_h)$ associated with actions x_h can be found by solving the non-singular system of Vandermonde:

$$\sum_{h=1}^{m} \sigma(x_h) \cdot x_h^j = \mu_j \qquad\qquad 0 \leq j \leq m-1$$

Best response computation

Given strategy $\overline{\boldsymbol{\sigma}}_2$, the problem BR_1 to find the agent 1's best response is:

$$\min \sum_{s \in S} u_s$$

$$u_s - \sum_{h \in \Psi_{2,s}} \overline{\sigma}_{2,s,h} \sum_{k=0}^{m} \sum_{j=0}^{m} r_{s,k,j} \cdot (x_1)^k \cdot (\overline{x}_{2,s,h})^j - \gamma \sum_{s' \in S} u'_s \cdot \sum_{k=0}^{m} p_{s,s',k} \cdot (x_1)^k \in \mathcal{P}^+(X_{1,s}) \qquad \forall s \in S_1$$

$$u_s - \sum_{h \in \Psi_{2,s}} \overline{\sigma}_{2,s,h} \sum_{k=0}^{m} \sum_{j=0}^{m} r_{s,k,j} \cdot (x_1)^k \cdot (\overline{x}_{2,s,h})^j - \gamma \sum_{h \in \Phi_{2,s}} \sum_{s' \in S} u'_s \sum_{k=0}^{m} p_{s,s',k} \cdot (\overline{x}_{2,s,h})^k \in \mathcal{P}^+(X_{2,s}) \overline{\sigma}_{2,s,h} \qquad \forall s \in S_2$$

Call:

$$\overline{R}_s^T \cdot [x_s]_{d_s} = \sum_{h \in \Psi_{2,s}} \overline{\sigma}_{2,s,h} \sum_{k=0}^{m} \cdot \sum_{j=0}^{m} r_{s,j,k} \cdot (\overline{x}_{2,s,h})^j \cdot (x_1)^k$$

$$\mathbf{u}^T \cdot \overline{P}_s^T \cdot [x_s]_{d_s} = \sum_{h \in \Psi_{2,s}} \overline{\sigma}_{2,s,h} \sum_{s' \in S} u'_s \sum_{k=0}^{m} p_{s,s',j} \cdot (\overline{x}_{2,s,h})^k \cdot (x_1)^0$$

when $X_s = [0, 1]$ the above mathematical program can be formulated as the following SDP:

$$\min \sum_{s \in S} u_s$$

$$\mathcal{H}^*(Z_s + \frac{1}{2}(L_1 W_s L_2^T + L_2 W_s L_2^T) - L_2 W_s L_2^T) - u_s - \overline{R}_s - \gamma P_s \mathbf{u} = 0 \qquad \forall s \in S_1 \quad (6.12)$$

$$\mathcal{H}^*(Z_s + \frac{1}{2}(L_1 W_s L_2^T + L_2 W_s L_2^T) - L_2 W_s L_2^T) - u_s - \overline{R}_s - \gamma \overline{P}_s \mathbf{u} = 0 \qquad \forall s \in S_2 \quad (6.13)$$

$$W_s, Z_s \succeq 0 \qquad \forall s \in S \quad (6.14)$$

constraints (6.12) translate constraints (6.3), while constraints (6.13) translate constraints (6.4); W_s and Z_s are matrices of auxiliary variables. BR_2 can be similarly defined.

6.2 Equilibrium approximation with non-polynomial games

In case of non-polynomial games, we approximate reward and state transitions with polynomial functions and then we apply Algorithm 1. If maximum approximation errors are small we can expect to find solutions that produces nearly-optimal outcomes. Notice that, it is known from approximation theory that continuous functions defined over compact subsets of a d-dimensional Euclidean space can be approximated to arbitrary accuracy with a degree-n polynomial [14]. In this section, we provide theoretical bounds on how the equilibrium approximation accuracy is affected by using reward and state transition functions that are polynomial approximations of the original ones.

For the sake of presentation, in the following we use the contract form $f_s^{\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2}$ to denote the expected value of a function f in state s with respect to the strategy profile $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$:

$$f_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} = \mathbb{E}_{\substack{x_1 \sim \boldsymbol{\sigma}_{1,s} \\ x_2 \sim \boldsymbol{\sigma}_{2,s}}} f_s(x1, x2)$$

Given a stochastic game with reward function $r_\cdot(\cdot,\cdot)$ and state transition probabilities $p_{\cdot,\cdot}(\cdot)$, we consider their polynomial approximations $\tilde{r}_\cdot(\cdot,\cdot)$ and $\tilde{p}_{\cdot,\cdot}(\cdot)$ with the following maximum error bounds:[1]

$$\|r_s(\cdot,\cdot) - \tilde{r}_s(\cdot,\cdot)\|_\infty \leq \delta \qquad \forall s \in S \tag{6.15}$$

$$\|p_{s,s'}(\cdot) - \tilde{p}_{s,s'}(\cdot)\|_\infty \leq \rho \qquad \forall s, s' \in S \tag{6.16}$$

We define an ϵ_s-MPE as a strategy profile such that no agent can gain more than ϵ in state s by deviating from her strategy, while in the other state utilities can be arbitrary. We state the following theoretical results. The first result is similar to [13], but stronger.

Theorem 6.2 *Given game $\mathcal{G}$ and absorbing state $\underline{s}$, the MPE strategy profile $(\tilde{\boldsymbol{\sigma}}_1, \tilde{\boldsymbol{\sigma}}_2)$ when agents' utility functions are $\tilde{u}_{\underline{s}}(\cdot,\cdot)$ and $-\tilde{u}_{\underline{s}}(\cdot,\cdot)$, respectively, is an $\epsilon_{\underline{s}}$-MPE with $\epsilon_{\underline{s}} \leq 2\delta$ when agents' utility functions are $u_{\underline{s}}(\cdot,\cdot)$ and $-u_{\underline{s}}(\cdot,\cdot)$.*

Proof. Call $\boldsymbol{\sigma}_1^*$ the best response of agent 1 to the $\boldsymbol{\sigma}_2$ of agent 2. We can compute the upper bound over the expected utility loss of agent 1 (recalling that, by definition, for every $\boldsymbol{\sigma}_1$ we have $\tilde{u}_{\underline{s}}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} - \tilde{u}_{\underline{s}}^{\boldsymbol{\sigma}_1,\tilde{\boldsymbol{\sigma}}_2} \geq 0$):

$$\begin{aligned}
\epsilon_{\underline{s}} &= u_{\underline{s}}^{\sigma_1^*,\tilde{\boldsymbol{\sigma}}_2} - u_{\underline{s}}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} = r_{\underline{s}}^{\sigma_1^*,\tilde{\boldsymbol{\sigma}}_2} - r_{\underline{s}}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} \\
&\leq r_{\underline{s}}^{\sigma_1^*,\tilde{\boldsymbol{\sigma}}_2} - \tilde{r}_{\underline{s}}^{\sigma_1^*,\tilde{\boldsymbol{\sigma}}_2} + \tilde{r}_{\underline{s}}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} - r_{\underline{s}}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} \\
&\leq |r_{\underline{s}}^{\sigma_1^*,\tilde{\boldsymbol{\sigma}}_2} - \tilde{r}_{\underline{s}}^{\sigma_1^*,\tilde{\boldsymbol{\sigma}}_2}| + |\tilde{r}_{\underline{s}}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} - r_{\underline{s}}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2}| \\
&\leq \|r_{\underline{s}}(\cdot,\cdot) - \tilde{r}_{\underline{s}}(\cdot,\cdot)\|_\infty + \|\tilde{r}_{\underline{s}}(\cdot,\cdot) - r_{\underline{s}}(\cdot,\cdot)\|_\infty \\
&\leq 2\delta
\end{aligned}$$

The same reasoning can be applied to agent 2, obtaining the same upper bound. Hence, the theorem is proved. □

[1] Given an arbitrary univariate function p_s, it is possible to find the best m-degree polynomial approximation $\tilde{p}_s$ (i.e., the m-degree polynomial minimizing ρ) with [51]. The optimal approximation of r_s is harder, it being a bivariate function. In this case, the algorithm presented in [10] can be used to find a good polynomial approximation $\tilde{r}_s$ of r_s.

We generalize the above theorem for generic states when $\rho = 0$. Given state s, call $l(s)$ the largest number of actions needed to reach an absorbing state from s. We first introduce the following lemma which bounds the difference between the two utility functions $u_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}$ and $\tilde{u}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}$ for any state s and any strategy profile $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$.

Lemma 6.1 *(With $\rho = 0$.) Given a tree-based game $\mathcal{G}$, for any state s and for any strategy profile $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$, the maximum absolute difference between utility of agent 1 computed according to $u(\cdot,\cdot)$ and utility of agent 1 computed according to $\tilde{u}(\cdot,\cdot)$ is less than $\delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$: $\|u_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty \leq \delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$.*

Proof. Given the bound on the reward functions in (6.15) we can write:

$$\begin{aligned}\|u_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty &\leq \|r_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{r}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty \\ &\quad +\gamma\|\sum_{s'} p_{s,s'}^{\boldsymbol{\sigma}_c}\left(u_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\right)\|_\infty \\ &\leq \delta + \gamma\max_{s'}\|u_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty\end{aligned}$$

By the recursive application of the above formula starting from an absorbing state $\underline{s}$ up to state s, we obtain: $\|u_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty \leq \delta\sum_{i=0}^{l(s)}(\gamma)^i = \delta\frac{1-(\gamma)^{l(s)+1}}{1-\gamma}$. □

Now, we are ready to extend theorem 6.2 to the case of tree-based games.

Theorem 6.3 *(With $\rho = 0$.) Given a tree-based game $\mathcal{G}$ and a generic state s, the MPE strategy profile $(\tilde{\boldsymbol{\sigma}}_1, \tilde{\boldsymbol{\sigma}}_2)$ of the subgame whose root node is s when agents' utility functions are $\tilde{u}(\cdot,\cdot)$ and $-\tilde{u}(\cdot,\cdot)$, respectively, is an ϵ_s-MPE with $\epsilon_s \leq 2\delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$ of such subgame when agents' utility functions are $u(\cdot,\cdot)$ and $-u(\cdot,\cdot)$.*

Proof. For every state s' in the subgame whose root node s, call $\boldsymbol{\sigma}_2^*(\boldsymbol{\sigma}_2)$ the best strategy of agent 1 found by backward induction when agent 2 plays $\boldsymbol{\sigma}_2$. By following the same line of theorem 6.2's proof, we can compute the upper bound over the expected utility loss of agent 1 at s as a function of the difference of the two utility functions $u_s(\cdot,\cdot)$ and $\tilde{u}_s(\cdot,\cdot)$, which have been bounded in Lemma 6.1:

$$\begin{aligned}\epsilon_s &= u_s^{\boldsymbol{\sigma}_1^*,\tilde{\boldsymbol{\sigma}}_2} - u_s^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} \\ &\leq u_s^{\boldsymbol{\sigma}_1^*,\tilde{\boldsymbol{\sigma}}_2} - \tilde{u}_s^{\boldsymbol{\sigma}_1^*,\tilde{\boldsymbol{\sigma}}_2} + \tilde{u}_s^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} - u_s^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} \\ &\leq 2\|u_s(\cdot,\cdot) - \tilde{u}_s(\cdot,\cdot)\|_\infty \\ &\leq 2\delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}\end{aligned}$$

The same result can be obtained for the loss of agent 2. □

Now, we focus on graph-based games.

Corollary 6.1 *(With $\rho = 0$.) The bound stated in theorem 6.3 can be generalized to graph-based games $\mathcal{G}$ as follows:*

$$\epsilon \leq \lim_{l(s)\to+\infty} 2\delta\frac{1-\gamma^{l(s)+1}}{1-\gamma} = \frac{2\delta}{1-\gamma}$$

Notice that the above bound is state independent and is meaningful (i.e., $\epsilon < 1$) only when $2\delta + \gamma < 1$.

Call $\Phi(s) = \sum_{s'\in \mathrm{FH}(s)} \tilde{u}_{s'}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_1} - \sum_{s'\in \mathrm{LH}(s)} \tilde{u}_{s'}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_1}$ where sets $\mathrm{FH}(s)$ and $\mathrm{LH}(s)$ are defined as follows. Call q_s the number of states s' reachable from s with a single action. $\mathrm{FH}(s)$ contains the $\lfloor q_s/2 \rfloor$ states with the highest $\tilde{u}_{s'}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2}$, while $\mathrm{LH}(s)$ contains the $\lfloor q_s/2 \rfloor$ states with the lowest $\tilde{u}_{s'}^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2}$. Call $\overline{\Phi} = \max_s\{\Phi(s)\}$.

Similarly to what has been done in Lemma 6.1, in the following we bound the difference between the two utility functions $u_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}$ and $\tilde{u}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}$ for any state s and any strategy profile $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$ when $\delta = 0$ and $\rho > 0$.

Lemma 6.2 *(With $\delta = 0$.) Given a tree-based game $\mathcal{G}$, for any state s and for any strategy profile $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$, the maximum absolute difference between utility of agent 1 computed according to $u(\cdot,\cdot)$ and utility of agent 1 computed according to $\tilde{u}(\cdot,\cdot)$ is less than $\delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$: $\|u_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty \leq \delta\frac{1-\gamma^{l(s)+1}}{1-\gamma}$.*

Proof. Given the bound on the transition probability functions in (6.16), we obtain:

$$\begin{aligned}
\|u_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty &\leq \|r_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{r}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty \\
&\quad +\gamma\|\sum_{s'}\left(p_{s,s'}^{\boldsymbol{\sigma}_c} u_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{p}_{s,s'}^{\boldsymbol{\sigma}_c}\tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\right)\|_\infty \\
&= \gamma\|\sum_{s'}\left(p_{s,s'}^{\boldsymbol{\sigma}_c} u_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - p_{s,s'}^{\boldsymbol{\sigma}_c}\tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\right) \\
&\quad + \sum_{s'}\left(p_{s,s'}^{\boldsymbol{\sigma}_c}\tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{p}_{s,s'}^{\boldsymbol{\sigma}_c}\tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\right)\|_\infty \\
&\leq \gamma\|\sum_{s'} p_{s,s'}^{\boldsymbol{\sigma}_c}\left(u_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\right)\|_\infty \\
&\quad +\gamma\|\sum_{s'}\left(p_{s,s'}^{\boldsymbol{\sigma}_c} - \tilde{p}_{s,s'}^{\boldsymbol{\sigma}_c}\right)\tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty \\
&\leq \gamma\max_{s'}\|u_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty + \gamma\rho\Phi(s) \\
&\leq \gamma\rho\overline{\Phi} + \gamma\max_{s'}\|u_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_{s'}^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty
\end{aligned}$$

By the recursive application of the above formula starting from an absorbing state $\underline{s}$ up to state s, we obtain: $\|u_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2} - \tilde{u}_s^{\boldsymbol{\sigma}_1,\boldsymbol{\sigma}_2}\|_\infty \leq \gamma\rho\overline{\Phi}\frac{1-(\gamma)^{l(s)+1}}{1-\gamma}$. □

Theorem 6.4 *(With $\delta = 0$.) Given a tree-based game $\mathcal{G}$ and a generic state s, the MPE strategy profile $(\tilde{\boldsymbol{\sigma}}_1, \tilde{\boldsymbol{\sigma}}_2)$ of the subgame whose root node is s when agents' utility functions are $\tilde{u}(\cdot,\cdot)$ and $-\tilde{u}(\cdot,\cdot)$, respectively, is an ϵ_s-MPE with $\epsilon_s \leq 2\gamma\rho\overline{\Phi}\frac{1-\gamma^{l(s)+1}}{1-\gamma}$ of such subgame when agents' utility functions are $u(\cdot,\cdot)$ and $-u(\cdot,\cdot)$.*

Proof. By theorem 6.2, we compute the upper bound over the expected utility loss of agent 1 at s as a function of the loss of agent 1 in the states directly reachable from s:

$$\begin{aligned}
\epsilon_s &= u_s^{\boldsymbol{\sigma}_1^*,\tilde{\boldsymbol{\sigma}}_2} - u_s^{\tilde{\boldsymbol{\sigma}}_1,\tilde{\boldsymbol{\sigma}}_2} \\
&\leq 2\|u_s(\cdot,\cdot) - \tilde{u}_s(\cdot,\cdot)\|_\infty \\
&\leq 2\gamma\rho\overline{\Phi}\frac{1-(\gamma)^{l(s)+1}}{1-\gamma}
\end{aligned}$$

The same result can be obtained for the loss of agent 2. □

Now, we focus on graph-based games.

Corollary 6.2 *(With $\delta = 0$.) The bound stated in theorem 6.4 can be generalized to graph-based games $\mathcal{G}$ as follows:*

$$\epsilon_s \leq \lim_{l(s)\to+\infty} 2\gamma\rho\overline{\Phi}\frac{1-(\gamma)^{l(s)+1}}{1-\gamma} = \frac{2\gamma\rho\overline{\Phi}}{1-\gamma}$$

As previously, the above bound is state independent and is significative (i.e., $\epsilon < 1$) only when $\gamma(1 + 2\rho\overline{\Phi}) < 1$.

Taken together, the results exposed above allow us to state the bound for the general case, in which $\delta, \rho \geq 0$:

Theorem 6.5 *Given a tree-based game $\mathcal{G}$ and a generic state s, the MPE strategy profile $(\tilde{\boldsymbol{\sigma}}_1, \tilde{\boldsymbol{\sigma}}_2)$ of the subgame whose root node is s when agents' utility functions are $\tilde{u}(\cdot,\cdot)$ and $-\tilde{u}(\cdot,\cdot)$, respectively, is an ϵ_s-MPE with $\epsilon_s \leq 2(\delta+\gamma\rho\overline{\Phi})\frac{1-(\gamma)^{l(s)+1}}{1-\gamma}$ of such subgame when agents' utility functions are $u(\cdot,\cdot)$ and $-u(\cdot,\cdot)$.*

The proof is easy, the two bounds being additive in our problem. Similarly, we obtain:

Corollary 6.3 *The bound stated in Theorem 6.5 can be generalized to graph-based games $\mathcal{G}$ as follows:*

$$\epsilon_s \leq \lim_{l(s)\to+\infty} 2(\delta + \gamma\rho\overline{\Phi})\frac{1-(\gamma)^{l(s)+1}}{1-\gamma} = \frac{2(\delta + \gamma\rho\overline{\Phi})}{1-\gamma}$$

Notice that this bound is meaningful (i.e., < 1) only when $2\delta + \gamma(1 + 2\rho\overline{\Phi}) < 1$.

Finally, we collect all the above results and we consider the situation in which an $\tilde{\epsilon}$-MPE computed on the approximated game $\tilde{\mathcal{G}}$ is used in the original game $\mathcal{G}$.

Theorem 6.6 *Given a game $\mathcal{G}$ and a generic state s, an $\tilde{\epsilon}$-MPE strategy profile $(\tilde{\boldsymbol{\sigma}}_1, \tilde{\boldsymbol{\sigma}}_2)$ of the approximated game $\tilde{\mathcal{G}}$, is an ϵ^*-MPE for game $\mathcal{G}$, with $\epsilon^* \leq \max_s\{\epsilon_s\} + \tilde{\epsilon}$.*

Again, the proof stems from the additivity of the two bounds and the definition of ϵ-MPE.

6.3 Experimental evaluation

We implemented our algorithm with Matlab 7.12 calling Yalmip R20120109 [45] and SDPT3 v.4 [37] to solve SDPs. We conducted the experiments on a UNIX computer with dual quad-core 2.33GHz CPUs and 16GB RAM.

We preliminarily evaluated the efficiency of our algorithm with medium/small polynomial games. We randomly generated 10 game instances with $|S| \in \{2, 4, 10, 20, 30, 40, 50\}$ and $m \in \{2, 5, 10, 15, 20\}$ as follows. Without loss of generality, $r_{s,k,j}$ has been uniformly drawn from $[-1, 1]$ except $r_{i,0,0}$ that is set equal to $r_{i,0,0} = m \cdot (m + 1)/2$ to guarantee that the reward functions are always positive on $[0, 1] \times [0, 1]$. We generated $p_{s,s',i}$ by exploiting SDP programming: we formulated the space of feasible polynomials ($p_{s,s',i} \geq 0$, $\sum_{s'} p_{s,s',i} = 1$) as an SDP and we randomly selected a feasible point of such space. The reward functions have been normalized such that $\max_s\{u_s\} = 1$ and $\min_s\{u_s\} = 0$.

We applied our algorithm to the above experimental setting. Fig. 6.1 shows how ϵ varies with the number of iterations (the plot is semi-log). Fixed the iteration, for each value of m we report the value of ϵ averaged over the instances with all the different $|S|$. At every iteration, ϵ is monotonically decreasing with m and the ratio between ϵ with $m = 20$ (max used degree) and ϵ with $m = 2$ (min used degree) is always less than 10 and hence the performances with different m are close. ϵ decreases exponentially with the number of iterations and gets very small - in $[10^{-5}, 10^{-4}]$ - even after few iterations. Then, we evaluated the computation time. About 98% of the compute time per iteration is required by the resolution of the SDPs (PS_i, DS_i, and BR_i require approximately the same compute time). In Fig. 6.2 we report how the average compute time needed by Yalmip and SDPT3 to solve a single SDP (precisely, PS_1) varies with $|S|$ for different m. The average compute time remains short even with $|S| = 50$ and $m = 20$, and, therefore, with small/medium instances, the algorithm scales very well finding ϵ-MPEs with very small ϵ by short compute time.

Finally, we preliminarily evaluated the effectiveness of our theoretical bounds for non-polynomial games by evaluating, with Caliari et al. [10], the approximation error δ with infinity norm for different classes of bivariate functions (their approximation is harder than that of univariate functions). In Tab. 6.1, we report, for dif-

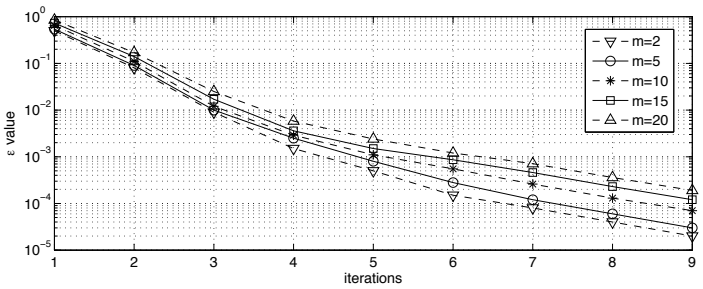


Figure 6.1: Average ϵ value.

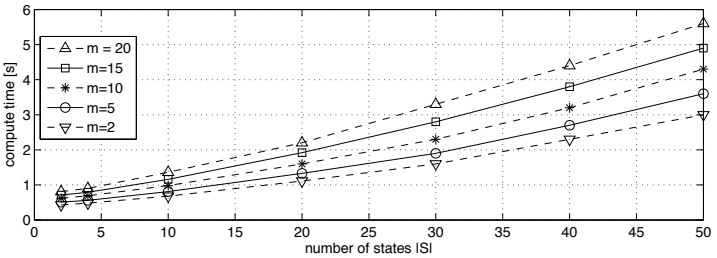


Figure 6.2: Average compute time per single SDP.

ferent m, the average δ with three different function classes (exp, sin, linear piecewise) by generating randomly 30 instances per class (10 per n). $\delta \approx 10^{-5}$ for every n when $m = 20$ with smooth functions (exp/sin), while $\delta \geq 10^{-2}$ with continuous but not differentiable functions (piecewise). Anyway, also with these last functions the error and hence the upper bound on ϵ is reasonably small.

6.4 Conclusions

We studied the problem to find and approximate an MPE with continuous two-player zero-sum stochastic games with switching control. We provided an algorithm based on SDP that converges to an MDP when the game is polynomial. When instead the game

class	n	polynomial degree (m)				
		2	5	10	15	20
$\sum_{j=1}^{n} \alpha_j \cdot \exp(\sum_{i=1}^{2} \beta_{i,j} \cdot (x_i - \gamma_{i,j}))$	1	$1.86 \cdot 10^{-1}$	$1.65 \cdot 10^{-2}$	$1.34 \cdot 10^{-5}$	$1.80 \cdot 10^{-5}$	$1.05 \cdot 10^{-5}$
	5	$2.83 \cdot 10^{-1}$	$0.59 \cdot 10^{-2}$	$2.03 \cdot 10^{-4}$	$6.41 \cdot 10^{-5}$	$1.31 \cdot 10^{-4}$
	10	$2.95 \cdot 10^{-1}$	$1.60 \cdot 10^{-2}$	$1.69 \cdot 10^{-4}$	$4.52 \cdot 10^{-5}$	$6.01 \cdot 10^{-5}$
$\sum_{j=1}^{n} \alpha_j \cdot \sin(\sum_{i=1}^{2} \beta_{i,j} \cdot (x_i - \gamma_{i,j}))$	1	$2.46 \cdot 10^{-1}$	$6.60 \cdot 10^{-3}$	$2.17 \cdot 10^{-4}$	$5.87 \cdot 10^{-5}$	$2.20 \cdot 10^{-5}$
	5	$2.52 \cdot 10^{-1}$	$6.80 \cdot 10^{-3}$	$2.16 \cdot 10^{-4}$	$5.62 \cdot 10^{-5}$	$4.72 \cdot 10^{-5}$
	10	$3.24 \cdot 10^{-1}$	$1.24 \cdot 10^{-2}$	$2.24 \cdot 10^{-4}$	$6.06 \cdot 10^{-5}$	$5.77 \cdot 10^{-5}$
$\sum_{j=1}^{n+1} \sum_{k=1}^{n+1} \sum_{i=1}^{2} \beta_{i,k,j} x_i x_i \in \mathcal{D}_{i,k}$	1	$1.68 \cdot 10^{-1}$	$1.00 \cdot 10^{-1}$	$2.88 \cdot 10^{-2}$	$3.67 \cdot 10^{-2}$	$3.85 \cdot 10^{-2}$
	5	$2.91 \cdot 10^{-1}$	$1.33 \cdot 10^{-1}$	$5.41 \cdot 10^{-2}$	$4.51 \cdot 10^{-2}$	$4.34 \cdot 10^{-2}$
	10	$2.51 \cdot 10^{-1}$	$1.26 \cdot 10^{-1}$	$6.20 \cdot 10^{-2}$	$3.99 \cdot 10^{-2}$	$4.26 \cdot 10^{-2}$

Table 6.1: Approximation error δ with different classes of non-polynomial functions.

is non-polynomial, we approximate the reward and the transition functions minimizing the error with infinity norm and then we apply our algorithm. In this case, we provide theoretical guarantees over the value of ϵ of the ϵ-MPE to which the algorithm converges. Finally, we experimentally evaluated our algorithm, showing that with small/medium games it is efficient and that the approximation error with infinity norm is small.

As future works, we aim at studying the scalability of the algorithm with large instances, evaluating our theoretical bounds with a variety of function classes, improving them, and extending the algorithm to solve general-sum games. In addition, we will remove the assumption of switching controller and we will explore both verification [21] and computation [20] of perfection-based solution concepts with continuous actions.

Part II

Learning in extensive-form games

Efficient evolutionary dynamics in extensive-form games 7

In this chapter, we originally explore the adoption of evolutionary game theory tools with sequence form for the study of extensive-form games.

We provide the following main contributions

- we show that the standard replicator dynamics for normal form cannot be adopted with the sequence form, the strategies produced by replication not being well-defined sequence-form strategies,
- we design an *ad hoc* version of the discrete-time replicator dynamics for sequence form and we show that it is sound, the strategies produced by replication being well-defined sequence-form strategies,
- we show that our replicator dynamics is realization equivalent to the standard discrete-time replicator dynamics for normal form and therefore that the two replicator dynamics evolve in the same way,
- we extend our discrete-time replicator dynamics to the continuous-time case, showing that the same properties are satisfied

and extending standard tools to study the stability of the strategies to our replicator.

This chapter is structured as follows. In Section 7.1 we introduce the discrete-time replicator dynamics for sequence-form representation. In Section 7.2 we show the realization equivalence between the normal-form replicator dynamics and the sequnce-form one. In Section 7.3 we extend the results of previous sections in the continuous-time case. Finally, in Section 7.4 we focus on characterizing a strategy profile in terms of evolutionary stability.

7.1 Discrete-time replicator dynamics for sequence-form representation

Initially, we show that the standard discrete-time replicator dynamics for normal form cannot be directly applied when sequence form is adopted. Standard replicator dynamics applied to the sequence form is easily obtained by considering each sequence q as a plan p and thus substituting $\mathbf{e}_q$ to $\mathbf{e}_p$ in (2.5)-(2.6) where $\mathbf{e}_q$ is zero for all the components q' such that $q' \neq q$ and one for the component q' such that $q' = q$.

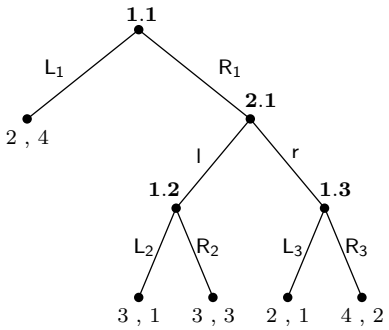


Figure 7.1: Running example.

Example 7.1 *A pair of sequence-form strategies of the game in Fig. 7.1 are:*

$$\mathbf{x}_1^T(t) = \begin{bmatrix} 1 & \frac{1}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \frac{2}{3} \end{bmatrix} \quad \mathbf{x}_2^T(t) = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$$

Proposition 7.1 *The replicator (2.5)-(2.6) does not satisfy the sequence-form constraints.*

Proof. The proof is by counterexample. Consider $\mathbf{x}_1(t)$ and $\mathbf{x}_2(t)$ equal to the strategies used in Example 7.1. At time $t+1$ the strategy profile generated by (2.5)-(2.6) is:

$$\mathbf{x}_1^T(t{+}1) = \begin{bmatrix} 0 & \frac{1}{3} & 0 & \frac{1}{2} & \frac{1}{6} & 0 & 0 \end{bmatrix} \quad \mathbf{x}_2^T(t{+}1) = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \end{bmatrix}$$

that does not satisfy the sequence-form constraints, e.g., $x_i(\mathsf{q}_\varnothing, t+1) \neq 1$ for all i. □

The critical issue behind the failure of the standard replicator dynamics lies in the definition of vector $\mathbf{e}_q$. Now we describe how the standard discrete-time replicator dynamics can be modified to be applied to the sequence form. In our variation, we substitute $\mathbf{e}_q$ with an opportune vector $\mathbf{g}_q$ that depends on the strategy $\mathbf{x}_i(t)$ and it is generated as described in Algorithm 6, obtaining:

$$x_1(q, t+1) = x_1(q,t) \cdot \frac{\mathbf{g}_q^T(\mathbf{x}_1(t)) \cdot U_1 \cdot \mathbf{x}_2(t)}{\mathbf{x}_1^T(t) \cdot U_1 \cdot \mathbf{x}_2(t)} \tag{7.1}$$

$$x_2(q, t+1) = x_2(q,t) \cdot \frac{\mathbf{x}_1^T(t) \cdot U_2 \cdot \mathbf{g}_q(\mathbf{x}_2(t))}{\mathbf{x}_1^T(t) \cdot U_2 \cdot \mathbf{x}_2(t)} \tag{7.2}$$

The basic idea behind the construction of vector $\mathbf{g}_q$ is:

- assigning "1" to the probability of all the sequences contained in q,
- normalizing the probability of the sequences extending the contained in q,
- assigning "0" to the probability of all the other sequences.

We describe the generation of vector $\mathbf{g}_q(\mathbf{x}_i(t))$, for clarity we use as running example the generation of $\mathbf{g}_{\mathsf{R}_1\mathsf{R}_3}(\mathbf{x}_1(t))$ related to Example 7.1:

- all the components of $\mathbf{g}_q(\mathbf{x}_i(t))$ are initialized equal to "0", e.g.,

$$\mathbf{g}_{\mathsf{R}_1\mathsf{R}_3}(\mathbf{x}_1(t))^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Algorithm 6 generate_$\mathbf{g_q}(\mathbf{x}_i(t))$

1: $\mathbf{g}_q(\mathbf{x}_i(t)) = \mathbf{0}$
2: **if** $x_i(q,t) \neq 0$ **then**
3: **for** $q' \in Q_i$ s.t. $q' \subseteq q$ **do**
4: $g_q(q', \mathbf{x}_i(t)) = 1$
5: **for** $q'' \in Q_i$ s.t. $q'' \cap q = q'$ **and** $q'' = q'|a| \ldots : a \in \rho(h), q \not\to h$ **do**
6: $g_q(q'', \mathbf{x}_i(t)) = \frac{x_i(q'',t)}{x_i(q',t)}$
7: **return** $\mathbf{g}_q(\mathbf{x}_i(t))$

- if sequence q is played, the algorithm assigns:
  - "1" to all the components $g_q(q', \mathbf{x}_i(t))$ of $\mathbf{g}_q(\mathbf{x}_i(t))$ where $q' \subseteq q$ (i.e., q' is a subsequence of q), e.g.,

$$\mathbf{g}_{\mathsf{R_1R_3}}(\mathbf{x}_1(t))^T = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

  - "$\frac{x_i(q'',t)}{x_i(q',t)}$" to all the components $g_q(q'', \mathbf{x}_i(t))$ of $\mathbf{g}_q(\mathbf{x}_i(t))$ where $q' \subseteq q$ with $q' = q'' \cap q$ and sequence q'' is defined as $q'' = q'|a| \ldots$ with $a \in \rho(h)$ and $q \not\to h$ (i.e., q' is a subsequence of q and q'' extends q' off the path identified by q), e.g.,

$$\mathbf{g}_{\mathsf{R_1R_3}}(\mathbf{x}_1(t))^T = \begin{bmatrix} 1 & 0 & 1 & \frac{1}{2} & \frac{1}{2} & 0 & 1 \end{bmatrix}$$

  - all the other components are left equal to "0",
- if sequence q is not played, $\mathbf{g}_q(\mathbf{x}_i(t))$ can be arbitrary, since the q-th equation of (7.1)-(7.2) is always zero given that $x_i(q,t) = 0$ for every t.

All the vectors $\mathbf{g}_q(\mathbf{x}_1(t))$ of Example 7.1 are:

	$\mathbf{g}_{q\varnothing}$	$\mathbf{g}_{L1}$	$\mathbf{g}_{R_1}$	$\mathbf{g}_{R_1L_2}$	$\mathbf{g}_{R_1R_2}$	$\mathbf{g}_{R_1L_3}$	$\mathbf{g}_{R_1R_3}$
$\mathsf{q}_\varnothing$	1	1	1	1	1	1	1
L_1	$\frac{1}{3}$	1	0	0	0	0	0
R_1	$\frac{2}{3}$	0	1	1	1	1	1
$\mathsf{R}_1\mathsf{L}_2$	$\frac{1}{3}$	0	$\frac{1}{2}$	1	0	$\frac{1}{2}$	$\frac{1}{2}$
$\mathsf{R}_1\mathsf{R}_2$	$\frac{1}{3}$	0	$\frac{1}{2}$	0	1	$\frac{1}{2}$	$\frac{1}{2}$
$\mathsf{R}_1\mathsf{L}_3$	0	0	0	0	0	1	0
$\mathsf{R}_1\mathsf{R}_3$	$\frac{2}{3}$	0	1	1	1	0	1

We show that replicator dynamics (7.1)-(7.2) do not violate sequence-form constraints.

Theorem 7.1 *Given a well-defined sequence-form strategy profile* $(\mathbf{x}_1(t), \mathbf{x}_2(t))$*, the output strategy profile* $(\mathbf{x}_1(t+1), \mathbf{x}_2(t+1))$ *of replicator dynamics (7.1)-(7.2) satisfies sequence-form constraints.*

Proof. The constraints forced by sequence form are:

- $x_i(\mathsf{q}_\varnothing, t) = 1$ for every i,
- $x_i(q,t) = \sum_{a\in\rho(w)} x_i(q|a,t)$ for every sequence q, action a, node w such that $w = h(q|a)$, and for every agent i.

Assume, by hypothesis of the theorem, that the above constraints are satisfied at t, we need to prove that constraints

$$x_i(\mathsf{q}_\varnothing, t+1) = 1 \tag{7.3}$$

$$x_i(q, t+1) = \sum_{a\in\rho(w)} x_i(q|a, t+1) \tag{7.4}$$

are satisfied. Constraint (7.3) always holds because $\mathbf{g}_{\mathsf{q}_\varnothing}(\mathbf{x}_1(t)) = \mathbf{x}_1(t)$. We rewrite constraints (7.4) as

$$x_i(q,t)\cdot\frac{\mathbf{g}_q^T(\mathbf{x}_i(t))\cdot U_i\cdot\mathbf{x}_{-i}(t)}{\mathbf{x}_i^T(t)\cdot U_i\cdot\mathbf{x}_{-i}(t)} = \\ = \sum_{a\in\rho(w)}\left(x_i(q|a,t)\cdot\frac{\mathbf{g}_{q|a}^T(\mathbf{x}_i(t))\cdot U_i\cdot\mathbf{x}_{-i}(t)}{\mathbf{x}_i^T(t)\cdot U_i\cdot\mathbf{x}_{-i}(t)}\right) \tag{7.5}$$

Conditions (7.5) hold if the following condition holds

$$x_i(q,t)\cdot \mathbf{g}_q^T(\mathbf{x}_i(t)) = \sum_{a\in\rho(w)} \left(x_i(q|a,t)\cdot \mathbf{g}_{q|a}^T(\mathbf{x}_i(t))\right) \qquad (7.6)$$

Notice that condition (7.6) is a vector of equalities, one per sequence q'. Condition (7.6) is trivially satisfied for components q' such that $g_q(q',\mathbf{x}_i(t)) = 0$. To prove the condition for all the other components, we introduce two lemmas.

Lemma 7.1 *Constraint (7.6) holds for all components $g_q(q',\mathbf{x}_i(t))$ of $\mathbf{g}_q(\mathbf{x}_i(t))$ such that $q' \sqsubseteq q$.*

Proof. By construction, $g_q(q',\mathbf{x}_i(t)) = 1$ for every $q' \sqsubseteq q$. For every extension $q|a$ of q, we have that $q' \sqsubseteq q \sqsubset q|a$. For this reason $g_{q|a}(q',\mathbf{x}_i(t)) = 1$. Thus

$$x_i(q,t)\cdot g_q(q',\mathbf{x}_i(t)) = \sum_{a\in\rho(w)} \left(x_i(q|a,t)\cdot g_{q|a}(q',\mathbf{x}_i(t))\right) \quad \text{iff}$$
$$x_i(q,t)\cdot 1 = \sum_{a\in\rho(w)} x_i(q|a,t)\cdot 1$$

that holds by hypothesis. Therefore the lemma is proved. □

Lemma 7.2 *Constraint (7.6) holds for all components $g_q(q'',\mathbf{x}_i(t))$ of $\mathbf{g}_q(\mathbf{x}_i(t))$ where $q' \sqsubseteq q$ with $q' = q''\sqcap q$ and sequence $q'' = q'|a'|\dots$ with $a' \in \rho(h)$ and $q \not\to h$.*

Proof. For all q'', $g_q(q'',\mathbf{x}_i(t)) = \frac{x_i(q'',t)}{x_i(q',t)}$ by construction. In the right side term of (7.6), for all a we can have either $q|a \not\sqsubset q''$ or $q|a \sqsubset q''$. In the former we have that $g_{q|a}(q'',\mathbf{x}_i(t)) = \frac{x_i(q'',t)}{x_i(q',t)}$, in the latter there exists only one action a such that $g_{q|a}(q'',\mathbf{x}_i(t)) = \frac{x_i(q'',t)}{x_i(q|a,t)}$, while for the other actions a^* the value of $g_{q|a^*}(q'',\mathbf{x}_i(t))$ is zero. Hence, we can have two cases: if $q|a \not\sqsubset q''$, then

$$x_i(q,t)\cdot g_q(q'',\mathbf{x}_i(t)) = \sum_{a\in\rho(w)} \left(x_i(q|a,t)\cdot g_{q|a}(q'',\mathbf{x}_i(t))\right) \quad \text{iff}$$
$$x_i(q,t)\cdot \frac{x_i(q'',t)}{x_i(q',t)} = \sum_{a\in\rho(w)} \left(x_i(q|a,t)\cdot \frac{x_i(q'',t)}{x_i(q',t)}\right)$$

that holds by hypothesis, otherwise if $q|a \subset q''$, then

$$x_i(q,t) \cdot g_q(q'', \mathbf{x}_i(t)) = \sum_{a \in \rho(w)} \left(x_i(q|a,t) \cdot g_{q|a}(q'', \mathbf{x}_i(t)) \right) \quad \text{iff}$$

$$x_i(q,t) \cdot \frac{x_i(q'',t)}{x_i(q,t)} = x_i(q|a,t) \cdot \frac{x_i(q'',t)}{x_i(q|a,t)}$$

that always holds. Therefore the lemma is proved. □

From the application of Lemmas 7.1 and 7.2, it follows that condition (7.6) holds. □

7.2 Replicator dynamics realization equivalence

We can show that the evolutionary dynamics of (7.1)-(7.2) are realization equivalent to the evolutionary dynamics of the normal-form replicator dynamics and therefore that the two replicator dynamics evolve in the same way.

Initially, we introduce two lemmas that we will exploit to prove the main result.

Lemma 7.3 *Given:*

- *a normal--form strategy* $\boldsymbol{\pi}_i$ *in reduced normal form,*
- *its equivalent behavioral strategy* $\boldsymbol{\sigma}_i$*,*
- *a subset of actions* $\{a_1, \ldots, a_m\} \subseteq A_i$*,*

it holds

$$\sum_{p \in P : a_1, \ldots, a_m \in p} \pi_i(p) = \prod_{a \in \bigcup_{j=1}^{m} q(a_j)} \sigma_i(a) \tag{7.7}$$

Proof. Suppose that $p = a_1, \ldots, a_n$. By (2.1) we know that

$$\pi_i(p) = \sigma_i(a_1) \cdot \cdots \cdot \sigma_i(a_n) \tag{7.8}$$

For all plan of actions $p \in P$ where $\{a_1, \ldots, a_m\} \in p$, given an actiona a such that $a \notin \{a_1, \ldots, a_m\}$, we can have two possibilities

1. $a \in \bigcup_{j=1}^{m} q(a_j)$, in this case the action a is present in every plan of actions p, being always present $\{a_1, \ldots, a_m\}$; thus

$$\sum_{p \in P : a_1, \ldots, a_m \in p} \pi_i(p') =$$
$$\sigma_i(a) \cdot \prod_{j=1}^{m} \sigma_i(a_j) \cdot \sum_{p \in P : a_1, \ldots, a_m \in p} (\sigma_i(a_{m+2}) \cdot \cdots \cdot \sigma_i(a_n))$$

2. $a \notin \bigcup_{j=1}^{m} q(a_j)$, in this case there is a subset $P' \subseteq P$ such that there is exactly a $p \in P'$ for each action $a' \in \rho(h)$, where $a \in \rho(h)$.

   By definition of behavioral strategy we know that

$$\sum_{a \in \rho(h)} \sigma_i(a) = 1 \qquad \forall h \in H$$

   Thus

$$\sum_{p \in P : a_1, \ldots, a_m \in p} \pi_i(p) =$$
$$\prod_{j=1}^{m} \sigma_i(a_j) \cdot \sum_{p \in P : a_1, \ldots, a_m \in p} (\sigma_i(a_{m+2}) \cdot \cdots \cdot \sigma_i(a_n)) \cdot \sum_{a \in \rho(h)} \sigma_i(a) =$$
$$\prod_{j=1}^{m} \sigma_i(a_j) \cdot \sum_{p \in P : a_1, \ldots, a_m \in p} (\sigma_i(a_{m+2}) \cdot \cdots \cdot \sigma_i(a_n))$$

Thus, we can write

$$\sum_{p \in P : a_1, \ldots, a_m \in p} \pi_i(p) = \sum_{p \in P : a_1, \ldots, a_m \in p} \sigma_i(a_1) \cdot \cdots \cdot \sigma_i(a_n)$$

where, by Point 1, we know that all the actions a that are in the path of some $a_1, \ldots, a_m$, $a \in \bigcup_{j=1}^{m} q(a_j)$, are present in every plan of actions

$$\sum_{p\in P:a_1,\dots,a_m\in p} \pi_i(p) =$$
$$= \prod_{a\in\bigcup_{j=1}^{m} q(a_j)} \sigma_i(a) \cdot \sum_{p\in P:a_1,\dots,a_m\in p} \sigma_i(a_{m+k}) \cdot \dots \cdot \sigma_i(a_n)$$

and, by Point 2, the other actions sum to "1"

$$\sum_{p\in P:a_1,\dots,a_m\in p} \pi_i(p) = \prod_{a\in\bigcup_{j=1}^{m} q(a_j)} \sigma_i(a)$$

This completes the proof of the lemma. □

Lemma 7.4 *Given*

- *a reduced-normal-form strategy* $\boldsymbol{\pi}_i(t)$ *of agent* i,
- *a sequence-form strategy* $\mathbf{x}_i(t)$ *realization equivalent to* $\boldsymbol{\pi}_i(t)$,

it holds that $x_i(q|a,t) \cdot \mathbf{g}_{q|a}^T(\mathbf{x}_i(t))$ *is realization equivalent to* $\sum_{p\in P:a\in p} \left(\pi_i(p,t) \cdot \mathbf{e}_p^T\right)$ *for all* $a \in A_i$ *and* $q \in Q_i$ *with* $q|a \in Q_i$.

Proof. We denote by $\tilde{\mathbf{x}}_p(t)$ the sequence-form strategy realization equivalent to $\mathbf{e}_p(t)$. According to [39], we can rewrite the thesis of the theorem as

$$x_i(q|a,t) \cdot \mathbf{g}_{q|a}^T(\mathbf{x}_i(t)) = \sum_{p\in P:a\in p} \left(\pi_i(p,t) \cdot \tilde{\mathbf{x}}_p(t)^T\right) \quad \forall a \in A_i \tag{7.9}$$

Notice that, for each action a and sequence q such that $q|a \in Q_i$, condition (7.9) is a vector of equality conditions. Given a and q, two cases are possible:

1. $x_i(q|a,t) = 0$ and then $\sum_{p\in P:a\in p} \pi_i(p,t) = 0$, thus conditions (7.9) hold;
2. $x_i(q|a,t) \neq 0$, in this case:

- for all components $g_{q|a}(q', \mathbf{x}_i(t))$ of $\mathbf{g}_{q|a}(\mathbf{x}_i(t))$ and $\tilde{x}_p(q', t)$ of $\tilde{\mathbf{x}}_p(t)$ such that $q' \subseteq q|a$, we have that $\tilde{x}_p(q', t) = 1$ for all $p \in P$ with $a \in p$ and Algorithm 1 sets $g_{q|a}(q', \mathbf{x}_i(t)) = 1$, thus we can rewrite (7.9) as

$$x_i(q|a,t) \cdot g_{q|a}(q', \mathbf{x}_i(t)) = \sum_{p \in P: a \in p} (\pi_i(p,t) \cdot \tilde{x}_p(q', t))$$
$$\text{iff} \quad x_i(q|a,t) \cdot 1 = \sum_{p \in P: a \in p} (\pi_i(p,t) \cdot 1)$$

  that holds by hypothesis and thus conditions (7.9) hold;
- for all components $g_{q|a}(q'', \mathbf{x}_i(t))$ of $\mathbf{g}_{q|a}(\mathbf{x}_i(t))$ and $\tilde{x}_p(q'', t)$ of $\tilde{\mathbf{x}}_p(t)$ such that q'' such that $q'' \cap q = q'$ and sequence $q'' = q'|a'| \ldots$ with $a' \in \rho(h)$ and $q \not\to h$, we have that $\tilde{x}_p(q'', t) = 1$ for all $p \in P$ with $a, a(q'') \in p$ and "0" otherwise, and Algorithm 1 sets $g_{q|a}(q'', \mathbf{x}_i(t)) = \frac{x_i(q'', t)}{x_i(q', t)}$, thus we can rewrite (7.9) as

$$x_i(q|a,t) \cdot g_{q|a}(q'', \mathbf{x}_i(t)) = \sum_{p \in P: a \in p} (\pi_i(p,t) \cdot \tilde{x}_p(q'', t))$$
$$\text{iff} \quad x_i(q|a,t) \cdot \frac{x_i(q'', t)}{x_i(q', t)} = \sum_{p \in P: a, a(q'') \in p} (\pi_i(p,t) \cdot 1)$$

  Using the relationship with the behavioral strategies, we can write

$$x_i(q|a,t) \cdot \frac{x_i(q'', t)}{x_i(q', t)} = \prod_{a' \in q|a} \sigma_i(a', t) \cdot \frac{\prod_{a' \in q''} \sigma_i(a', t)}{\prod_{a' \in q'} \sigma_i(a', t)}$$

  Being $q' \sqsubseteq q|a$ and $q' \sqsubseteq q''$ we have

$$x_i(q|a,t) \cdot \frac{x_i(q'', t)}{x_i(q', t)} =$$
$$\prod_{a' \in q|a \setminus q'} \sigma_i(a', t) \cdot \prod_{a' \in q'} \sigma_i(a', t) \cdot \prod_{a' \in q'' \setminus q'} \sigma_i(a', t) =$$
$$\prod_{a^* \in \bigcup_{a' \in \{a, a(q'')\}} q(a')} \sigma_i(a^*, t)$$

by using Lemma 7.3 it that can be easily rewrite as

$$\sum_{p\in P:a,a(q'')\in p} \pi_i(p,t) = \prod_{a^*\in\bigcup_{a'\in\{a,a(q'')\}} q(a')} \sigma_i(a^*,t)$$

and therefore conditions (7.9) hold.

This completes the proof of the lemma. □

Now we state the main result. It allows us to study the evolution of a strategy in a game directly in sequence form, instead of using the normal form, and it guarantees that the two dynamics (sequence and normal) are equivalent.

Theorem 7.2 *Given*

- *a normal-form strategy profile* $(\boldsymbol{\pi}_1(t), \boldsymbol{\pi}_2(t))$ *and its evolution* $(\boldsymbol{\pi}_1(t+1), \boldsymbol{\pi}_2(t+1))$ *according to (2.5)-(2.6),*
- *a sequence-form strategy profile* $(\mathbf{x}_1(t), \mathbf{x}_2(t))$ *and its evolution* $(\mathbf{x}_1(t+1), \mathbf{x}_2(t+1))$ *according to (7.1)-(7.2),*

if $(\boldsymbol{\pi}_1(t), \boldsymbol{\pi}_2(t))$ *and* $(\mathbf{x}_1(t), \mathbf{x}_2(t))$ *are realization equivalent, then also* $(\boldsymbol{\pi}_1(t+1), \boldsymbol{\pi}_2(t+1))$ *and* $(\mathbf{x}_1(t+1), \mathbf{x}_2(t+1))$ *are realization equivalent.*

Proof. Assume, by hypothesis of the theorem, that $(\mathbf{x}_1(t), \mathbf{x}_2(t))$ is realization equivalent to $(\boldsymbol{\pi}_1(t), \boldsymbol{\pi}_2(t))$. Thus, according to [39], for every agent i it holds

$$x_i(q|a,t) = \sum_{p\in P:a\in p} \pi_i(p,t) \qquad \forall a \in A_i$$

We need to prove that the following conditions hold:

$$x_i(q|a,t+1) = \sum_{p\in P:a\in p} \pi_i(p,t+1) \qquad \forall a \in A_i \tag{7.10}$$

By applying the definition of replicator dynamics, we can rewrite the conditions (7.10) as:

$$x_i(q|a,t) \cdot \frac{\mathbf{g}_{q|a}^T(\mathbf{x}_i(t)) \cdot U_i \cdot \mathbf{x}_{-i}(t)}{\mathbf{x}_i^T(t) \cdot U_i \cdot \mathbf{x}_{-i}(t)} =$$
$$= \sum_{p \in P: a \in p} \left(\pi_i(p,t) \cdot \frac{\mathbf{e}_p^T \cdot U_i \cdot \boldsymbol{\pi}_{-i}(t)}{\boldsymbol{\pi}_i^T(t) \cdot U_i \cdot \boldsymbol{\pi}_{-i}(t)} \right) \qquad \forall a \in A_i \quad (7.11)$$

Given that, by hypothesis, $\mathbf{x}_i^T(t) \cdot U_i \cdot \mathbf{x}_{-i}(t) = \boldsymbol{\pi}_i^T(t) \cdot U_i \cdot \boldsymbol{\pi}_{-i}(t)$, we can rewrite conditions (7.11) as:

$$x_i(q|a,t) \cdot \mathbf{g}_{q|a}^T(\mathbf{x}_i(t)) \cdot U_i \cdot \mathbf{x}_{-i}(t) =$$
$$= \sum_{p \in P: a \in p} \left(\pi_i(p,t) \cdot \mathbf{e}_p^T \cdot U_i \cdot \boldsymbol{\pi}_{-i}(t) \right) \qquad \forall a \in A_i$$

These conditions hold if and only if $\sum_{p \in P: a \in p} \left(\pi_i(p,t) \cdot \mathbf{e}_p^T \right)$ is realization equivalent to $x_i(q|a,t) \cdot \mathbf{g}_{q|a}^T(\mathbf{x}_i(t))$. By Lemma 7.4, this equivalence holds. □

7.3 Continuous-time replicator dynamics for sequence-form representation

The sequence-form continuous-time replicator equation is

$$\dot{x}_1(q,t) = x_1(q,t) \cdot [(\mathbf{g}_q(\mathbf{x}_1(t)) - \mathbf{x}_1(t))^T \cdot U_1 \cdot \mathbf{x}_2(t)] \quad (7.12)$$
$$\dot{x}_2(q,t) = x_2(q,t) \cdot [\mathbf{x}_1(t)^T \cdot U_2 \cdot (\mathbf{g}_q(\mathbf{x}_2(t) - \mathbf{x}_2(t))] \quad (7.13)$$

Theorem 7.3 *Given a well-defined sequence-form strategy profile* $(\mathbf{x}_1(t), \mathbf{x}_2(t))$*, the output strategy profile* $(\mathbf{x}_1(t + \Delta t), \mathbf{x}_2(t + \Delta t))$ *of replicator dynamics (7.12)-(7.13) satisfies sequence-form constraints.*

Proof. The constraints forced by sequence form are:

- $x_i(\mathsf{q}_\varnothing, t) = 1$ for every i,
- $x_i(q,t) = \sum_{a \in \rho(w)} x_i(q|a,t)$ for every sequence q, action a, node w such that $w = h(q|a)$, and for every agent i.

Assume, by hypothesis of the theorem, that constraints are satisfied at a given time point t, we need to prove that constraints

$$x_i(\mathtt{q}_\varnothing, t + \Delta t) = 1 \tag{7.14}$$

$$x_i(q, t + \Delta t) = \sum_{a \in \rho(w)} x_i(q|a, t + \Delta t) \tag{7.15}$$

are satisfied. Constraint (7.14) always holds because $\mathbf{g}_q(\mathbf{x}_1(t)) = \mathbf{x}_1(t)$. We rewrite constraints (7.15) as

$$\begin{aligned} & x_i(q,t) \cdot [(\mathbf{g}_q(\mathbf{x}_i(t)) - \mathbf{x}_i(t))^T \cdot U_i \cdot \mathbf{x}_{-i}(t)] = \\ & = \sum_{a \in \rho(w)} \left(x_i(q|a) \cdot [(\mathbf{g}_{q|a}(\mathbf{x}_i(t)) - \mathbf{x}_i(t))^T \cdot U_i \cdot \mathbf{x}_{-i}(t)] \right) \end{aligned} \tag{7.16}$$

Conditions (7.16) hold if the following conditions hold

$$x_i(q,t) \cdot \mathbf{g}_q^T(\mathbf{x}_i(t)) = \sum_{a \in \rho(w)} \left(x_i(q|a,t) \cdot \mathbf{g}_{q|a}^T(\mathbf{x}_i(t)) \right) \tag{7.17}$$

Notice that condition (7.17) is a vector of equalities. The above condition is trivially satisfied for components q' such that $g_q(q', \mathbf{x}_i(t)) = 0$. From the application of Lemmas 7.1 and 7.2, the condition (7.17) holds also for all the other components. □

Theorem 7.4 *Given*

- *a normal-form strategy profile* $(\boldsymbol{\pi}_1(t), \boldsymbol{\pi}_2(t))$ *and its evolution* $(\boldsymbol{\pi}_1(t + \Delta t), \boldsymbol{\pi}_2(t + \Delta t))$ *according to (2.7)-(2.8),*
- *a sequence-form strategy profile* $(\mathbf{x}_1(t), \mathbf{x}_2(t))$ *and its evolution* $(\mathbf{x}_1(t + \Delta t), \mathbf{x}_2(t + \Delta t))$ *according to (7.12)-(7.13),*

if $(\boldsymbol{\pi}_1(t), \boldsymbol{\pi}_2(t))$ *and* $(\mathbf{x}_1(t), \mathbf{x}_2(t))$ *are realization equivalent, then also* $(\boldsymbol{\pi}_1(t + \Delta t), \boldsymbol{\pi}_2(t + \Delta t))$ *and* $(\mathbf{x}_1(t + \Delta t), \mathbf{x}_2(t + \Delta t))$ *are realization equivalent.*

Proof. Assume, by hypothesis of the theorem, that $(\mathbf{x}_1(t), \mathbf{x}_2(t))$ is realization equivalent to $(\boldsymbol{\pi}_1(t), \boldsymbol{\pi}_2(t))$. Thus, according to [39], for every agent i it holds

$$x_i(q|a,t) = \sum_{p \in P: a \in p} \pi_i(p,t) \qquad \forall a \in A_i$$

We need to prove that the following conditions hold:

$$x_i(q|a,t+\Delta t) = \sum_{p \in P: a \in p} \pi_i(p,t+\Delta t) \qquad \forall a \in A_i \tag{7.18}$$

By applying the definition of replicator dynamics, we can rewrite the conditions (7.18) as:

$$\begin{aligned} & x_i(q|a,t) \cdot [(\mathbf{g}_{q|a}(\mathbf{x}_i(t)) - \mathbf{x}_i(t))^T \cdot U_i \cdot \mathbf{x}_{-i}(t)] = \\ & = \sum_{p \in P: a \in p} \left(\pi_i(p,t) \cdot [(\mathbf{e}_p - \boldsymbol{\pi}_i(t))^T \cdot U_i \cdot \boldsymbol{\pi}_{-i}(t)]\right) \qquad \forall a \in A_i \end{aligned} \tag{7.19}$$

Given that, by hypothesis, $\mathbf{x}_i^T(t) \cdot U_i \cdot \mathbf{x}_{-i}(t) = \boldsymbol{\pi}_i^T(t) \cdot U_i \cdot \boldsymbol{\pi}_{-i}(t)$, we can rewrite conditions (7.19) as:

$$\begin{aligned} & x_i(q|a,t) \cdot \mathbf{g}_{q|a}^T(\mathbf{x}_i(t)) \cdot U_i \cdot \mathbf{x}_{-i}(t) = \\ & \qquad = \sum_{p \in P: a \in p} \left(\pi_i(p,t) \cdot \mathbf{e}_p^T \cdot U_i \cdot \boldsymbol{\pi}_{-i}(t)\right) \qquad \forall a \in A_i \end{aligned}$$

These conditions hold if and only if $\sum_{p \in P: a \in p} \left(\pi_i(p,t) \cdot \mathbf{e}_p^T\right)$ is realization equivalent to $x_i(q|a,t) \cdot \mathbf{g}_{q|a}^T(\mathbf{x}_i(t))$. By Lemma 7.4, this equivalence holds. □

7.4 Analyzing the stability of a strategy profile

We focus on characterizing a strategy profile in terms of evolutionary stability. When the continuous-time replicator dynamics for normal-form is adopted, evolutionary stability can be analyzed by studying the eigenvalues of the Jacobian in that point [2]-non-positiveness of the eigenvalues is a necessary condition for asymptotical stability, while strict negativeness of the eigenvalues is sufficient. The Jacobian is

$$J = \begin{bmatrix} \frac{\partial \dot{x}_1(q_i,t)}{\partial x_1(q_j,t)} & \frac{\partial \dot{x}_1(q_i,t)}{\partial x_2(q_l,t)} \\ \frac{\partial \dot{x}_2(q_k,t)}{\partial x_1(q_j,t)} & \frac{\partial \dot{x}_2(q_k,t)}{\partial x_2(q_l,t)} \end{bmatrix} \quad \begin{array}{l} \forall q_i, q_j \in Q_1, \\ q_k, q_l \in Q_2 \end{array}$$

In order to study the Jacobian of our replicator dynamics, we need to complete the definition of $\mathbf{g}_q(\mathbf{x}_i(t))$. Indeed, we observe that some components of $\mathbf{g}_q(\mathbf{x}_i(t))$ are left arbitrary by Algorithm 1. Exactly, some q'' that are related to q' with $x_i(q',t) = 0$. While it is not necessary to assign values to such components during the evolution of the replicator dynamics, it is necessary when we study the Jacobian. The rationale follows. If $x_i(q',t) = 0$, then it will remain zero even after t. Instead, if, after the dynamics converged to a point, such a point has $x_i(q') = 0$ for some q', it might be the case that along the dynamics it holds $x_i(q') \neq 0$. Thus, in order to define these components of $\mathbf{g}_q(\mathbf{x}_i(t))$, we need to reason backward, assigning the values that they would have in the case such sequence would be played with a probability that goes to zero. In absence of degeneracy, Algorithm 2 addresses this issue assigning a value of "1" to a sequence q'' if it is the (unique, the game being non-degenerate) best response among the sequences extending q' and "0" otherwise, because at the convergence the agents play only the best response sequences. Notice that, in this case, $\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$ depends on both agents' strategies.

Algorithm 7 generate_$\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$

$\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t)) = \mathbf{0}$
for $q' \in Q_i$ s.t. $q' \sqsubseteq q$ **do**
  $g_q(q', \mathbf{x}_i(t), \mathbf{x}_{-i}(t)) = 1$
for $q'' \in Q_i$ s.t. $q'' \sqcap q = q'$ **and** $q'' = q'|a| \ldots : a \in \rho(h), q \not\to h$ **do**
  if $x_i(q',t) \neq 0$ **then**
    $g_q(q'', \mathbf{x}_i(t), \mathbf{x}_{-i}(t)) = \frac{x_i(q'',t)}{x_i(q',t)}$
  else if $q'' = \text{argmax}_{q^*:a(q^*)\in\rho(h)} \mathbb{E}[U_i(q^*, \mathbf{x}_{-i})]$ **then**
    $g_q(q'', \mathbf{x}_i(t), \mathbf{x}_{-i}(t)) = 1$
return $\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$

Given the above complete definition of $\mathbf{g}_q$, we can observe that all the components of $\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$ generated by Algorithm 2 are differentiable, being "0" or "1" or "$\frac{x_i(q'',t)}{x_i(q',t)}$". Therefore, we can derive the Jacobian as:

$$\frac{\partial \dot{x}_1(q_i,t)}{\partial x_1(q_j,t)} = \begin{cases} (\mathbf{g}_{q_i}(\mathbf{x}_1(t),\mathbf{x}_2(t)) - \mathbf{x}_1(t))^T \cdot U_1 \cdot \mathbf{x}_2(t) + \\ x_1(q_i,t) \cdot \\ \left[\left(\dfrac{\partial \mathbf{g}_{q_i}(\mathbf{x}_1(t),\mathbf{x}_2(t))}{\partial x_1(q_j,t)} - \mathbf{e}_i \right)^T \cdot U_1 \cdot \mathbf{x}_2(t) \right] & \text{if } i = j \\ x_1(q_i,t) \cdot \\ \left[\left(\dfrac{\partial \mathbf{g}_{q_i}(\mathbf{x}_1(t),\mathbf{x}_2(t))}{\partial x_1(q_j,t)} - \mathbf{e}_j \right)^T \cdot U_1 \cdot \mathbf{x}_2(t) \right] & \text{if } i \neq j \end{cases}$$

$$\frac{\partial \dot{x}_1(q_i,t)}{\partial x_2(q_l,t)} = x_1(q_i,t) \cdot [(\mathbf{g}_{q_i}(\mathbf{x}_1(t),\mathbf{x}_2(t)) - \mathbf{x}_1(t))^T \cdot U_1 \cdot \mathbf{e}_l]$$

$$\frac{\partial \dot{x}_2(q_k,t)}{\partial x_1(q_j,t)} = x_2(q_k,t) \cdot [\mathbf{e}_j^T \cdot U_2 \cdot (\mathbf{g}_{q_k}(\mathbf{x}_2(t),\mathbf{x}_1(t)) - \mathbf{x}_2(t))]$$

$$\frac{\partial \dot{x}_2(q_k,t)}{\partial x_2(q_l,t)} = \begin{cases} \mathbf{x}_1(t)^T \cdot U_2 \cdot (\mathbf{g}_{q_k}(\mathbf{x}_2(t),\mathbf{x}_1(t)) - \mathbf{x}_2(t)) + \\ x_2(q_k,t) \cdot \\ \left[\mathbf{x}_1(t)^T \cdot U_2 \cdot \left(\dfrac{\partial \mathbf{g}_{q_k}(\mathbf{x}_2(t),\mathbf{x}_1(t))}{\partial x_2(q_l,t)} - \mathbf{e}_k \right) \right] & \text{if } k = l \\ x_2(q_k,t) \cdot \\ \left[\mathbf{x}_1(t)^T \cdot U_2 \cdot \left(\dfrac{\partial \mathbf{g}_{q_k}(\mathbf{x}_2(t),\mathbf{x}_1(t))}{\partial x_2(q_l,t)} - \mathbf{e}_l \right) \right] & \text{if } k \neq l \end{cases}$$

With degenerate games, given a opponent's strategy profile $\mathbf{x}_{-i}(t)$ and a sequence $q \in Q_i$ such that $x_i(q,t) = 0$, we can have multiple best responses. Consider, e.g., the game in Example 7.1, with $\mathbf{x}_1^T(t) = [\ 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ]$, $\mathbf{x}_2^T(t) = [\ 1 \quad 1 \quad 0\ ]$ and compute $\mathbf{g}_{\mathsf{R_1L_3}}(\mathbf{x}_1(t),\mathbf{x}_2(t))$: both sequences $\mathsf{R_1L_2}$ and $\mathsf{R_1R_2}$ are best responses to $\mathbf{x}_2(t)$. Reasoning backward, we have different vectors $\mathbf{g}_q(\mathbf{x}_i,\mathbf{x}_{-i})$ for different dynamics. More precisely, we can partition the strategy space around $(\mathbf{x}_i,\mathbf{x}_{-i})$, associating a different best response with a different subspace and therefore with a different

$\mathbf{g}_q(\mathbf{x}_i, \mathbf{x}_{-i})$. Thus, in principle, in order to study the stability of a strategy profile, we would need to compute and analyze all the (potentially combinatory) Jacobians. However, we can show that all these Jacobians are the same and therefore, even in the degenerate case, we can safely study the Jacobian by using a $\mathbf{g}_q(\mathbf{x}_i, \mathbf{x}_{-i})$ as generated by Algorithm 2 except, if there are multiple best responses, Step 7-8 assign "1" only to one, randomly chosen, best response.

Theorem 7.5 *Given*

- *a specific sequence $q \in Q_i$ such that $x_i(q,t) = 0$,*
- *a sequence-form strategy $\mathbf{x}_{-i}(t)$,*
- *a sequence $q' \sqsubseteq q$,*
- *the number of sequences q'' such that $q'' \sqcap q = q'$ and $q'' = q'|a| \ldots : a \in \rho(h), q \not\to h$ and that are best responses to $\mathbf{x}_{-i}(t)$ is larger than one,*

the eigenvalues of the Jacobian are independent from which sequence q'' is choosen as best-response.

Proof. For each sequence q'' that is a best-response to $\mathbf{x}_{-i}(t)$, we can have different vectors $\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$. Suppose to take two different vectors $\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$ and $\mathbf{g}'_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$. To prove the equality of the two Jacobians we have to prove that each term is the same. All the terms multiplied by $\mathbf{x}_i(q,t) = 0$ can be discarded, they being equal to zero. For this reason the only term different from 0 in the Jacobian is $\frac{\partial \dot{x}_i(q,t)}{\partial \dot{x}_i(q,t)}$, thus we have to prove

$$(\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t)) - \mathbf{x}_i(t))^T \cdot U_1 \cdot \mathbf{x}_{-i}(t) = (\mathbf{g}'_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t)) - \mathbf{x}_i(t))^T \cdot U_i \cdot \mathbf{x}_{-i}(t) \quad (7.20)$$

We can rewrite the equality (7.20) as

$$\mathbf{g}_q^T(\mathbf{x}_i(t), \mathbf{x}_{-i}(t)) \cdot U_i \cdot \mathbf{x}_{-i}(t) = \mathbf{g}'^T_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t)) \cdot U_i \cdot \mathbf{x}_{-i}(t)$$

that always holds because $\mathbf{g}_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$ and $\mathbf{g}'_q(\mathbf{x}_i(t), \mathbf{x}_{-i}(t))$, even if they differ for some components, provide the same expected

utility by definition of best response. Even if an agent randomizes over multiple best responses, the theorem holds for the same reason. □

Q-learning based algorithm for the sequence form of extensive-form games 8

In this chapter, we develop, to the best of our knowledge, the first $\mathcal{Q}$-learning based algorithm working with the sequence form of an extensive-form game, thus allowing an exponential reduction of the length of the learning dynamics w.r.t. normal form. Differently from the normal-form case, with the sequence form the application of the $\mathcal{Q}$-learning is not straightforward and it requires a novel procedure to update the $\mathcal{Q}$-values. In particular, we provide the following contributions:

- we define a variation of the standard $\mathcal{Q}$-learning algorithm that is applicable to the sequence form of an extensive-form game, showing that, in addition to an exponential reduction of the learning times, it allows to learn strategy profiles that cannot be learned using the normal form representation,

- we show that, in expectation, the time-limit learning dynamics of our algorithm can be described by means of a new sequence-form replicator dynamics with a mutation term and that, when the mutation term tends to zero, the replicator dynamics converge to a Nash equilibrium with perfect-information games,

- we experimentally evaluate our $\mathcal{Q}$-learning based algorithm applied to the sequence form, comparing its dynamics w.r.t. the dynamics obtained when the standard $\mathcal{Q}$-learning algorithm is applied to the normal form and evaluating the accuracy of our replicator dynamics model as the learning parameters vary.

In Section 8.1 we present the variation of the standard $\mathcal{Q}$-learning algorithm that is applicable to the sequence form. In Section 8.2 we study the relationship between the time-limit learning dynamics of our algorithm and the replicator dynamics. Finally, in Section 8.3 we experimental evaluate our $\mathcal{Q}$-learning based algorithm applied to the sequence form.

8.1 Q-learning and sequence form

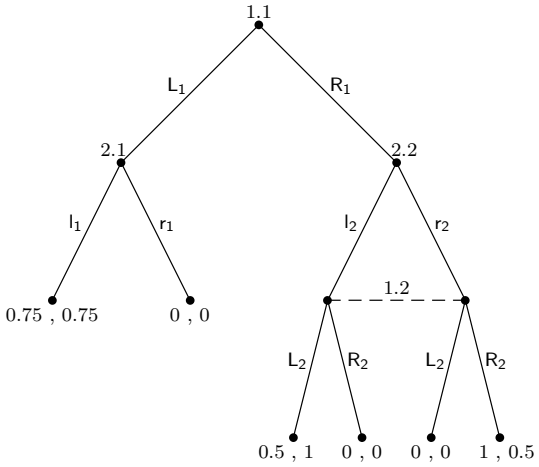


Figure 8.1: Example of two-agent imperfect-information extensive-form game used in for experimental results of Section 8.3.

The application of the $\mathcal{Q}$-learning algorithm to the normal form of an extensive-form game is straightforward as well as the derivation of its time-limit (standard) replicator dynamics based model. However, as discussed in the previous sections, the normal form being

exponentially large in the size of the game tree, the learning times get are exponentially long. In this section, we describe a variation of the $\mathcal{Q}$-learning algorithm for the sequence form of an extensive-form game, reducing exponentially the length of the learning times, while in the next section we show the connection of our algorithm with the sequence-form replicator dynamics described in [25]. Differently from [64], but consistently with the learning literature, we consider the learning rate α and the exploitation parameter $\tau(t)$ as functions of time.

The basic idea behind our algorithm is that each agent, at each repetition of the game, chooses a sequence-form pure strategy. Being the set of these strategies exponentially large in the size of the game (there is one sequence-form pure strategy per plan of the reduced normal form), each agent implicitly chooses a strategy by using a procedure that, given a strategy $\mathbf{x}_i$, builds a sequence-form pure strategy $\overline{\mathbf{x}}_i$ in polynomial time w.r.t. the size of the game. The procedure, summarized in Algorithm 8, is iterative and works as follows. In Steps 1-3, the empty sequence of an agent is chosen and the information sets directly reachable by the empty sequence are inserted in the set $inf\_sets\_to\_evaluate$. Then, until such set is not empty, in Step 5 the procedure extracts an information set h, in Step 6 randomly draws a sequence q with $a(q) \in \rho(h)$ according to $\mathbf{x}_i$ once normalized by the probability to reach h and assigns probability one to q in $\overline{\mathbf{x}}_i$, in Step 8 adds all the information sets directly reachable by q to $inf\_sets\_to\_evaluate$ and in Step 9 removes h from such set. Given that each information set is evaluated no more than once, the complexity of the procedure is linear in the size of the game tree.

In our learning algorithm, we associate a $\mathcal{Q}$-value with each sequence and we update the $\mathcal{Q}$-values according to the observed outcomes. Given that each agent plays a number of sequences at each repetition of the game (a sequence-form pure strategy includes multiple sequences), we need to modify the updating rule Eq. (2.10) allowing an agent to update multiple $\mathcal{Q}$-values (those corresponding to the sequences of the pure-strategy profiles $\overline{\mathbf{x}}$ chosen by Algorithm 8). Formally, for each $q|a$ such that $\overline{x}_i(q|a) = 1$ we have:

$$\mathcal{Q}_{t+1}(q|a) = (1-\alpha(t))\mathcal{Q}_t(q|a) + \alpha(t)\Big(r(q|a) + \gamma \max_{\substack{a' \in \rho(h):\\ a \in \rho(h)}} \mathcal{Q}_t(q|a')\Big) \tag{8.1}$$

Algorithm 8 sequence_form_pure_strategy($\mathbf{x}_i$)

1: $\overline{\mathbf{x}}_i = \mathbf{0}$
2: $\overline{x}_i(\mathsf{q}_\varnothing) = 1$
3: $inf\_sets\_to\_evaluate = \{h : \exists a \in \rho(h), \mathsf{q}_\varnothing | a \in Q_i\}$
4: **while** $inf\_sets\_to\_evaluate \neq \varnothing$ **do**
5: choose an information set $h \in inf\_sets\_to\_evaluate$
6: choose a sequence q such that $a(q) \in \rho(h)$ according $\mathbf{x}_i$
7: $\overline{x}_i(q) = 1$
8: $inf\_sets\_to\_evaluate = inf\_sets\_to\_evaluate \cup \{h' : \exists a \in \rho(h'), q | a \in Q_i\}$
9: $inf\_sets\_to\_evaluate = inf\_sets\_to\_evaluate \setminus \{h\}$
10: **return** $\overline{\mathbf{x}}_i$

where

$$r(q|a) = \overline{\mathbf{x}}_i^T(t) \cdot U_i \cdot \overline{\mathbf{x}}_{-i}(t).$$

The action-selection strategy is based on the Boltzmann distribution. More precisely, given the $\mathcal{Q}$-value of each sequence, we derive a sequence-form strategy profile as:

$$x_i(q|a,t) = x_i(q,t) \cdot \frac{e^{\tau(t)\mathcal{Q}_t(q|a)}}{\sum_{a' \in \rho(h): a \in \rho(h)} e^{\tau(t)\mathcal{Q}_t(q|a')}}. \tag{8.2}$$

The learning algorithm is summarized in Algorithm 9.

In Steps 2-3, the algorithm initializes the $\mathcal{Q}$-value associated with each sequence of each agent to a random value. In Steps 4-5, the algorithm derives the strategy profile specifying the probability distribution over the sequences by Boltzmann equation on the basis of the $\mathcal{Q}$-values. In Step 6, the algorithm repeats the following steps until the agents have learned the optimal policy. In Steps 7-8, each agent draws a sequence-form pure strategy as prescribed by Algorithm 8. In Steps 9-10, for each sequence chosen by Algorithm 8, the algorithm updates the $\mathcal{Q}$-values applying the updating rule Eq. (8.1). In Steps 11-12, the algorithm updates the probability distribution over the sequences given the new $\mathcal{Q}$-values. The algorithm is repeated until the maximum variation of the $\mathcal{Q}$-values is less than a given threshold ϵ.

Algorithm 9 $\mathcal{Q}$_learning_for_sequence_form

$t = 0$
for all $q \in Q$ **do**
  $\mathcal{Q}_t(q)$ drawn uniformly from $\left[\frac{\min_{\mathbf{x}} U_i(\mathbf{x})}{1-\gamma}, \frac{\max_{\mathbf{x}} U_i(\mathbf{x})}{1-\gamma}\right]$
for all $q|a \in Q$ **do**
  $x_i(q|a,t) = x_i(q,t) \cdot \frac{e^{\tau(t)\mathcal{Q}_t(q|a)}}{\sum_{a' \in \rho(h): a \in \rho(h)} e^{\tau(t)\mathcal{Q}_t(q|a')}}$
repeat
  for all $i \in \{1,2\}$ **do**
    $\overline{\mathbf{x}}_i$ = sequence_form_pure_strategy($\mathbf{x}_i$)
  for all $q|a$ such that $\overline{x}_i(q|a) == 1$ **do**
    $\mathcal{Q}_{t+1}(q|a) = (1 - \alpha(t))\mathcal{Q}_t(q|a) + \alpha(t)(r(q|a) + \gamma \max_{a' \in \rho(h): a \in \rho(h)} \mathcal{Q}_t(q|a'))$
  for all $q|a \in Q$ **do**
    $x_i(q|a,t) = x_i(q,t) \cdot \frac{e^{\tau(t)\mathcal{Q}_t(q|a)}}{\sum_{a' \in \rho(h): a \in \rho(h)} e^{\tau(t)\mathcal{Q}_t(q|a')}}$
    $t = t + 1$
until $\max_{q \in Q} |\mathcal{Q}_{t+1}(q) - \mathcal{Q}_t(q)| < \epsilon$
return $\mathbf{x}$

The sequence-form representation, besides being exponentially smaller than the normal-form one, it is also more expressive allowing agents to learn strategy profiles that in normal form cannot be learned. To show it, we state the following theorem.

Theorem 8.1 *Given the Boltzmann distribution and a constant τ, an extensive-form game and both its sequence and normal form representations, the space of learning strategy profiles in the latter representation is strictly contained in the space of the former one.*

Proof. In order to prove the theorem we need to introduce two lemmas.

Lemma 8.1 *Given a constant τ, if (2.11) and (2.12) hold, the agents can learn only the strategy profiles such that*

$$\frac{\tau}{1-\gamma}\cdot\left(\max_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi}) - \min_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})\right) \geq \log\left(\frac{\max_{p\in P_i}\pi_i(p,t)}{\min_{p\in P_i}\pi_i(p,t)}\right). \tag{8.3}$$

Proof. We can rewrite equation (2.12) as

$$\mathcal{Q}_t(p) = \frac{\log\left(C\cdot\pi_i(p,t)\right)}{\tau},$$

where C is an appropriate constant. Given Eq. (2.11) we have that

$$C \in \left[\frac{e^{\tau\cdot\frac{\min_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})}{1-\gamma}}}{\pi_i(p,t)}, \frac{e^{\tau\cdot\frac{\max_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})}{1-\gamma}}}{\pi_i(p,t)}\right] \qquad \forall p \in P_i$$

that implies

$$C \in \left[\frac{e^{\tau\cdot\frac{\min_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})}{1-\gamma}}}{\min_{p\in P_i}\pi_i(p,t)}, \frac{e^{\tau\cdot\frac{\max_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})}{1-\gamma}}}{\max_{p\in P_i}\pi_i(p,t)}\right].$$

For consistency reasons, the following inequality must hold

$$\frac{e^{\tau\cdot\frac{\min_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})}{1-\gamma}}}{\min_{p\in P_i}\pi_i(a,t)} \leq \frac{e^{\tau\cdot\frac{\max_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})}{1-\gamma}}}{\max_{p\in P_i}\pi_i(p,t)}.$$

□

Lemma 8.2 *Given a constant τ, if (2.11) and (8.2) hold, the agents can learn only the strategy profiles such that*

$$\frac{\tau}{1-\gamma}\cdot\left(\max_{\mathbf{x}} U_i(\mathbf{x}) - \min_{\mathbf{x}} U_i(\mathbf{x})\right) \geq \log\left(\frac{\max_{q|a\in\rho(h)} x_i(q|a,t)}{\min_{q|a\in\rho(h)} x_i(q|a,t)}\right) \qquad \forall h \in H. \tag{8.4}$$

Proof. Lemma 8.2 can be easily adapted to the sequence-form case obtaining the set of constraints of Eq. (8.4). □

Given a strategy profile in sequence form $\mathbf{x}$ and a strategy profile in normal form $\boldsymbol{\pi}$ that are realization equivalent, we have that $\max_{\mathbf{x}} U_i(\mathbf{x}) = \max_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})$ and $\min_{\mathbf{x}} U_i(\mathbf{x}) = \min_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})$. The relationship between sequence-form strategy profiles and normal-form ones is

$$\pi_i(p) = \prod_{q|a:a\in p} \frac{x_i(q|a)}{x_i(q)}.$$

So we have

$$\frac{\max_{p\in P_i} \pi_i(p,t)}{\min_{p\in P_i} \pi_i(p,t)} = \frac{\max_{p\in P_i} \left(\prod_{q|a:a\in p} x_i(q|a,t)\right)}{\min_{p\in P_i} \left(\prod_{q|a:a\in p} x_i(q|a,t)\right)} = \prod_{h\in H} \frac{\max_{q|a\in\rho(h)} x_i(q|a,t)}{\min_{q|a\in\rho(h)} x_i(q|a,t)} \geq \frac{\max_{q|a\in\rho(h)} x_i(q|a,t)}{\min_{q|a\in\rho(h)} x_i(q|a,t)}.$$

For each $\boldsymbol{\pi}$ such that Eq. (8.3) holds, there always exists a realization equivalent $\mathbf{x}$ such that

$$\frac{\tau}{1-\gamma} \cdot \left(\max_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi}) - \min_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})\right) = \frac{\tau}{1-\gamma} \cdot \left(\max_{\mathbf{x}} U_i(\mathbf{x}) - \min_{\mathbf{x}} U_i(\mathbf{x})\right) \geq \log\left(\frac{\max_{p\in P_i} \pi_i(p,t)}{\min_{p\in P_i} \pi_i(p,t)}\right) \geq \log\left(\frac{\max_{q|a\in\rho(h)} x_i(q|a,t)}{\min_{q|a\in\rho(h)} x_i(q|a,t)}\right) \forall h \in H,$$

but there exist strategy profiles $\mathbf{x}$ that do not admit a realization equivalent $\boldsymbol{\pi}$ such that Eq. (8.3) holds. For example, if we take $\mathbf{x}$ such that

$$\frac{\tau}{1-\gamma} \cdot \left(\max_{\mathbf{x}} U_i(\mathbf{x}) - \min_{\mathbf{x}} U_i(\mathbf{x})\right) = \log\left(\frac{\max_{q|a\in\rho(h)} x_i(q|a,t)}{\min_{q|a\in\rho(h)} x_i(q|a,t)}\right), \qquad \forall h \in H$$

and it exists $h \in H$ such that $\frac{\max_{q|a\in\rho(h)} x_i(q|a,t)}{\min_{q|a\in\rho(h)} x_i(q|a,t)} > 1$ we have that

$$\log\left(\frac{\max_{q|a\in\rho(h)} x_i(q|a,t)}{\min_{q|a\in\rho(h)} x_i(q|a,t)}\right) = \frac{\tau}{1-\gamma}\cdot\left(\max_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi}) - \min_{\boldsymbol{\pi}} U_i(\boldsymbol{\pi})\right) < \log\left(\frac{\max_{p\in P_i}\pi_i(p,t)}{\min_{p\in P_i}\pi_i(p,t)}\right). \tag{8.5}$$

But Eq. (8.5) and Eq. (8.3) are in contradiction. □

8.2 Dynamical analysis

In this section we study the dynamics of Algorithm 9 in expectation w.r.t. its realizations and their relationship with the replicator dynamics when there are 2 agents. The generalization with more agents is straightforward. We assume, as in [64], that the game is continuously repeated, the time between a repetition and the subsequent one tending to zero. This allows us to calculate (in expectation) the derivative of Eq. (8.2) as:

$$\begin{aligned}\dot{x}_i(q|a,t) = \dot{x}_i(q,t)\cdot\frac{x_i(q|a,t)}{x_i(q,t)} \\ + x_i(q|a,t)\Big(\dot{\tau}(t)\mathcal{Q}_t(q|a) + \tau(t)\dot{\mathcal{Q}}_t(q|a) \\ - \sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}} \frac{x_i(q|a',t)}{x_i(q,t)}\Big(\dot{\tau}(t)\mathcal{Q}_t(q|a') + \tau(t)\dot{\mathcal{Q}}_t(q|a')\Big)\Big)\end{aligned} \tag{8.6}$$

and the time derivative of Eq. (8.1) as

$$\dot{\mathcal{Q}}_t(q|a) = \alpha(t)\left(r(q|a) + \gamma\max_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\mathcal{Q}_t(q|a') - \mathcal{Q}_t(q|a)\right). \tag{8.7}$$

By replacing $\dot{\mathcal{Q}}_t(q|a)$ in Eq. (8.6) with the regret term of Eq. (8.7) we obtain

$$\dot{x}_i(q|a,t) = \dot{x}_i(q,t)\cdot\frac{x_i(q|a,t)}{x_i(q,t)} + x_i(q|a,t)\alpha(t)\Bigg(\dot{\tau}(t)\mathcal{Q}_t(q|a)$$
$$+\tau(t)\Bigg(r(q|a)+\gamma\max_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\mathcal{Q}_t(q|a')-\mathcal{Q}_t(q|a)\Bigg)-\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\cdot$$
$$\cdot\Bigg(\dot{\tau}(t)\mathcal{Q}_t(q|a')+\tau(t)\Big(r(q|a')+\gamma\max_{\substack{a''\in\rho(h):\\ a'\in\rho(h)}}\mathcal{Q}_t(q|a'')-\mathcal{Q}_t(q|a')\Big)\Bigg)\Bigg).$$

Given that $\sum_{a'\in\rho(h):a\in\rho(h)} x_i(q|a',t) = x_i(q,t)$ by definition of sequence form, we have that

$$\max_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\mathcal{Q}_t(q|a')-\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\gamma\max_{\substack{a''\in\rho(h):\\ a'\in\rho(h)}}\mathcal{Q}_t(q|a'')=0$$

thus, we can rewrite

$$\dot{x}_i(q|a,t) = \dot{x}_i(q,t)\cdot\frac{x_i(q|a,t)}{x_i(q,t)}$$
$$+ x_i(q|a,t)\alpha(t)\Bigg(\dot{\tau}(t)\mathcal{Q}_t(q|a)+\tau(t)\Big(r(q|a)-\mathcal{Q}_t(q|a)\Big)$$
$$-\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\Bigg(\dot{\tau}(t)\mathcal{Q}_t(q|a')+\tau(t)\Big(r(q|a')-\mathcal{Q}_t(q|a')\Big)\Bigg)\Bigg). \tag{8.8}$$

We can rewrite the reward in expectation of a sequence as its fitness as

$$r(q|a) = \mathbf{g}_{q|a}^T(\mathbf{x}_i(t))\cdot U_i\cdot\mathbf{x}_{-i}(t)$$

thus Eq. (8.8) becomes

$$\dot{x}_i(q|a,t) = \dot{x}_i(q,t)\cdot\frac{x_i(q|a,t)}{x_i(q,t)} + x_i(q|a,t)\alpha(t)\left(\dot{\tau}(t)\mathcal{Q}_t(q|a)\right.$$
$$\left.+\tau(t)\left(\mathbf{g}^T_{q|a}(\mathbf{x}_i(t))\cdot U_i\cdot\mathbf{x}_{-i}(t) - \mathcal{Q}_t(q|a)\right) - \sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\cdot\right.$$
$$\left.\cdot\left(\dot{\tau}(t)\mathcal{Q}_t(q|a') + \tau(t)\left(\mathbf{g}^T_{q|a'}\left(\mathbf{x}_i(t)\right)\cdot U_i\cdot\mathbf{x}_{-i}(t) - \mathcal{Q}_t(q|a')\right)\right)\right.$$

Given that

$$\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\cdot\mathbf{g}^T_{q|a'}(\mathbf{x}_i(t))\cdot U_i\cdot\mathbf{x}_{-i}(t) =$$
$$\mathbf{g}^T_q(\mathbf{x}_i(t))\cdot U_i\cdot\mathbf{x}_{-i}(t)$$

we have

$$\dot{x}_i(q|a,t) = \dot{x}_i(q,t)\cdot\frac{x_i(q|a,t)}{x_i(q,t)} + x_i(q|a,t)\alpha(t)$$
$$\left.\cdot\left(\tau(t)\left(\left(\mathbf{g}_{q|a}(\mathbf{x}_i(t)) - \mathbf{g}_q(\mathbf{x}_i(t))\right)^T\cdot U_i\cdot\mathbf{x}_{-i}(t)\right) - \tau(t)\mathcal{Q}_t(q|a)\right.\right.$$
$$\left.\left.+\dot{\tau}(t)\mathcal{Q}_t(q|a) - \sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\left(\dot{\tau}(t)\mathcal{Q}_t(q|a') - \tau(t)\mathcal{Q}_t(q|a')\right)\right)\right.$$

By Boltzmann distribution we have $\frac{x_i(q|a',t)}{x_i(q|a,t)} = \frac{e^{\tau(t)\mathcal{Q}_t(q|a')}}{e^{\tau(t)\mathcal{Q}_t(q|a)}}$ for every q, a, a' and by sequence form definition we have $\sum_{a'\in\rho(h):a\in\rho(h)}\frac{x_i(q|a',t)}{x_i(q,t)} = 1$ for every q, so we obtain:

$$
\begin{aligned}
&-\tau(t)\mathcal{Q}_t(q|a)+\tau(t)\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\mathcal{Q}_t(q|a')\\
&\quad=\tau(t)\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\left(\mathcal{Q}_t(q|a')-\mathcal{Q}_t(q|a)\right)\\
&\quad=\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\log\left(\frac{x_i(q|a',t)}{x_i(q|a,t)}\right)
\end{aligned}
$$

Thus

$$
\begin{aligned}
&\dot{x}_i(q|a,t)=\\
&\quad x_i(q|a,t)\Bigg[\alpha(t)\tau(t)\left(\left(\mathbf{g}_{q|a}(\mathbf{x}_i(t))-\mathbf{x}_i(t)\right)^T\cdot U_i\cdot\mathbf{x}_{-i}(t)\right)\\
&\quad+\left(\alpha(t)+\frac{\dot{\tau}(t)}{\tau(t)}\right)\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(q,t)}\log\left(\frac{x_i(q|a',t)}{x_i(q|a,t)}\right)\Bigg]+\\
&\qquad\qquad\dot{x}_i(q,t)\cdot\frac{x_i(q|a,t)}{x_i(q,t)}.
\end{aligned}
\tag{8.9}
$$

When $q=\mathsf{q}_\varnothing$, the time derivative $\dot{x}_i(\mathsf{q}_\varnothing,t)=0$ because, by definition of sequence form, $x_i(\mathsf{q}_\varnothing,t)=1$ for every t. Thus, when $q=\mathsf{q}_\varnothing$ Eq. (8.9) becomes:

$$
\begin{aligned}
&\dot{x}_i(\mathsf{q}_\varnothing|a,t)=\\
&\quad x_i(\mathsf{q}_\varnothing|a,t)\Bigg[\alpha(t)\tau(t)\left(\left(\mathbf{g}_{\mathsf{q}_\varnothing|a}(\mathbf{x}_i(t))-\mathbf{x}_i(t)\right)^T\cdot U_i\cdot\mathbf{x}_{-i}(t)\right)\\
&\quad+\left(\alpha(t)+\frac{\dot{\tau}(t)}{\tau(t)}\right)\sum_{\substack{a'\in\rho(h):\\ a\in\rho(h)}}\frac{x_i(q|a',t)}{x_i(\mathsf{q}_\varnothing,t)}\log\left(\frac{x_i(\mathsf{q}_\varnothing|a',t)}{x_i(\mathsf{q}_\varnothing|a,t)}\right)\Bigg].
\end{aligned}
\tag{8.10}
$$

By substituting iteratively $\dot{x}_i(\mathsf{q}_\varnothing|a,t)$ in the term $\dot{x}_i(q,t)\cdot\dfrac{x_i(q|a,t)}{x_i(q,t)}$ of Eq. (8.9) for every q, we obtain:

$$\dot{x}_1(q|a,t) = x_1(q|a,t)\Bigg[\alpha(t)\tau(t)\left(\left(\mathbf{g}_{q|a}(\mathbf{x}_1(t))-\mathbf{x}_1(t)\right)^T\cdot U_1\cdot\mathbf{x}_2(t)\right) + \left(\alpha(t)+\frac{\dot{\tau}(t)}{\tau(t)}\right)\sum_{\substack{h:\exists a^*\in q|a\\ a^*\in\rho(h)}}\sum_{a'\in\rho(h)}\frac{x_1(q|a',t)}{x_1(q,t)}\log\left(\frac{x_1(q|a',t)}{x_1(q|a^*,t)}\right)\Bigg] \tag{8.11}$$

$$\dot{x}_2(q|a,t) = x_2(q|a,t)\Bigg[\alpha(t)\tau(t)\left(\mathbf{x}_1(t)^T\cdot U_2\cdot\left(\mathbf{g}_{q|a}(\mathbf{x}_2(t))-\mathbf{x}_2(t)\right)\right) + \left(\alpha(t)+\frac{\dot{\tau}(t)}{\tau(t)}\right)\sum_{\substack{h:\exists a^*\in q|a\\ a^*\in\rho(h)}}\sum_{a'\in\rho(h)}\frac{x_2(q|a',t)}{x_2(q,t)}\log\left(\frac{x_2(q|a',t)}{x_2(q|a^*,t)}\right)\Bigg]. \tag{8.12}$$

The replicator dynamics Eq. (8.11)-(8.12) is formed by two terms: the selection (exploitation) one,

$$x_i(q|a,t)\alpha(t)\tau(t)\left(\left(\mathbf{g}_{q|a}(\mathbf{x}_i(t))-\mathbf{x}_i(t)\right)^T\cdot U_i\cdot\mathbf{x}_{-i}(t)\right)$$

and the mutation (exploration) one,

$$x_i(q|a,t)\left(\alpha(t)+\frac{\dot{\tau}(t)}{\tau(t)}\right)\sum_{\substack{h:\exists a^*\in q|a\\ a^*\in\rho(h)}}\sum_{a'\in\rho(h)}\frac{x_i(q|a',t)}{x_i(q,t)}\log\left(\frac{x_i(q|a',t)}{x_i(q|a^*,t)}\right).$$

The learning rate $\alpha(t)$ (contained in both terms) has the function of increasing or decreasing the speed of dynamics, but it does not affect

the trajectory. More interesting is the exploitation parameter $\tau(t)$ whose function is to increase or decrease the prominence of the selection term w.r.t. the mutation term. When $\tau(t)$ increases (decreases), Algorithm 9 assigns a greater (smaller) probability to sequences with high Q-value, preferring the exploitation (exploration) w.r.t. the exploration (exploitation); the same reflects in the replicator dynamics because a greater (smaller) $\tau(t)$ means that the selection term affects the learning dynamics more (less) than the mutation term.

The selection term of replicator dynamics in Eq. (8.11)-(8.12) is realization equivalent to the selection term of replicator dynamics in Eq. (2.13)-(2.14) as shown in [25]. That is, the learning dynamics in expectation of our $\mathcal{Q}$-learning algorithm are realization equivalent to the learning dynamics of the model proposed in [64] once applied to the normal form when the mutation term is zero. When the mutation term tends to zero, the two replicator dynamics models have the same rest points, but they can have different trajectories. When instead the mutation term does not tend to zero, the two replicator dynamics have different trajectories and rest points.

Theorem 8.2 *Given*

- *a normal-form strategy profile* $(\boldsymbol{\pi}_1(t), \boldsymbol{\pi}_2(t))$ *and its evolution* $(\boldsymbol{\pi}_1(t+\Delta t), \boldsymbol{\pi}_2(t+\Delta t))$ *according to (2.13)-(2.14),*

- *a sequence-form strategy profile* $(\mathbf{x}_1(t), \mathbf{x}_2(t))$ *and its evolution* $(\mathbf{x}_1(t+\Delta t), \mathbf{x}_2(t+\Delta t))$ *according to (8.11)-(8.12),*

if $(\boldsymbol{\pi}_1(t), \boldsymbol{\pi}_2(t))$ *and* $(\mathbf{x}_1(t), \mathbf{x}_2(t))$ *are realization equivalent, in general* $(\boldsymbol{\pi}_1(t+\Delta t), \boldsymbol{\pi}_2(t+\Delta t))$ *and* $(\mathbf{x}_1(t+\Delta t), \mathbf{x}_2(t+\Delta t))$ *are not realization equivalent.*

Proof. The proof is by counterexample using the game in Fig. 8.1. Suppose that at t the strategy profiles of game are 8.1 are

$$\boldsymbol{\pi}_1(t) = \begin{cases} \pi_1(\mathsf{L}_1*, t) & = 0.5 \\ \pi_1(\mathsf{R}_1\mathsf{L}_2, t) & = 0.25 \\ \pi_1(\mathsf{R}_1\mathsf{R}_2, t) & = 0.25 \end{cases} \quad \boldsymbol{\pi}_2(t) = \begin{cases} \pi_2(\mathsf{l}_1\mathsf{l}_2, t) & = 0.25 \\ \pi_2(\mathsf{l}_1\mathsf{r}_2, t) & = 0.25 \\ \pi_2(\mathsf{r}_1\mathsf{l}_2, t) & = 0.25 \\ \pi_2(\mathsf{r}_1\mathsf{r}_2, t) & = 0.25 \end{cases}$$

$$\mathbf{x}_1^T(t) = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.25 & 0.25 \end{bmatrix}$$
$$\mathbf{x}_2^T(t) = \begin{bmatrix} 1 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$$

Now we compute the strategy profiles at time $t+1$ with $\tau(t) = 1$ and $\alpha(t) = 1$:

$$\boldsymbol{\pi}_1(t+1) = \begin{cases} \pi_1(\mathsf{L}_1 *, t+1) & = 0.3960 \\ \pi_1(\mathsf{R}_1\mathsf{L}_2, t+1) & = 0.2800 \\ \pi_1(\mathsf{R}_1\mathsf{R}_2, t+1) & = 0.3240 \end{cases}$$

$$\boldsymbol{\pi}_2(t+1) = \begin{cases} \pi_2(\mathsf{l}_1\mathsf{l}_2, t+1) & = 0.2859 \\ \pi_2(\mathsf{l}_1\mathsf{r}_2, t+1) & = 0.2649 \\ \pi_2(\mathsf{r}_1\mathsf{l}_2, t+1) & = 0.2332 \\ \pi_2(\mathsf{r}_1\mathsf{r}_2, t+1) & = 0.2160 \end{cases}$$

$$\mathbf{x}_1^T(t+1) = \begin{bmatrix} 1 & 0.5042 & 0.4958 & 0.2297 & 0.2661 \end{bmatrix}$$
$$\mathbf{x}_2^T(t+1) = \begin{bmatrix} 1 & 0.5592 & 0.4408 & 0.5168 & 0.4832 \end{bmatrix}$$

that are not realization equivalent. □

Being realization equivalent the selection terms, the non-equivalence is due to the mutation terms. In fact, if we increase the exploitation parameter $\tau(t)$, that means the selection influences the dynamics more than the mutation, the distance between the two dynamics decreases. A more detailed experimental analysis is proposed in Section 8.3.

8.3 Experimental results

Relationship between replicators dynamics

As shown by Theorem 8.2, the replicator dynamics (8.11)-(8.12) and the replicator dynamics (2.13)-(2.14) are not realization equivalent, but only their selection terms. However, as τ increases, the trajectories of the two replicators get close. In Fig. 8.2 we show the trajectories of the two replicators, starting from the same point, for increasing values of τ.

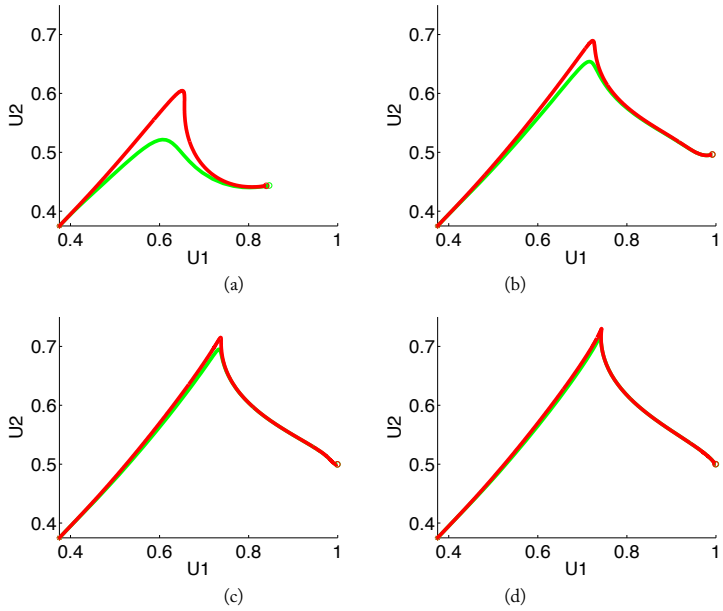


Figure 8.2: Sequence-form (red line) and normal-form (green line) replicator dynamics in the running example of Fig. 8.1 with $\alpha = 1$ for different τ: (a) $\tau = 5$, (b) $\tau = 10$, (c) $\tau = 15$ and (d) $\tau = 20$.

Learning dynamics length

We compare the learning dynamics length (in terms of iterations) obtained when the $\mathcal{Q}$-learning is applied to the normal form w.r.t. our sequence form version when the initial strategies are realization equivalent and with different configurations of the parameters as the size of the tree (branching factor b and depth d) varies. In our experimental setting τ is a linear increasing function of time starting from 0.0001 and ending to 0.5, α is exponential decreasing starting from 1 and ending to 0.2. The algorithm stops when the difference of expected utility between iterations n and $n-1$ of both agents is smaller than 0.001 for 1000 consecutive iterations. However, the average length is comparable only for very small game tree: with $b = 2$ and $d = 2$ the normal form requires about 1.5 times the number iterations required by the sequence form, while with $b = 2$ and $d = 3$

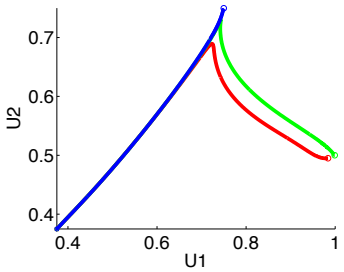


Figure 8.3: Example of different limit points of learning dynamics from the same starting point with constant but different τ; red line $\tau = 10$, green line $\tau = 20$, blue line $\tau = 30$.

the ratio is about 2.7; with larger d the ratio is larger than 1000. This is because the number of actions in the normal form is exponentially large w.r.t. the sequence form.

Parameters influence in the learning limit points

The parameters α and τ affect in different ways the trajectories of (8.11)-(8.12). While α only affects the speed of the dynamics without changing the trajectory, τ increases or decreases the prominence of selection term w.r.t. the mutation term. The parameter τ can dramatically affect the trajectories of the replicator dynamics changing both the learning limit points (by introducing small perturbations) and their basins of attraction. An example is shown in Fig. 8.3. Increasing τ from 10 to 20 the learning limit point is slightly perturbed getting close to the equilibrium with utilities $(1, 0.5)$, instead increasing τ from 20 to 30 the basins of attraction change and the learning trajectory converges to a different limit point.

Variability analysis

The replicator dynamics model provides an abstract (time-limit) description in expectation of the learning dynamics. We evaluate here the robustness of the model w.r.t. the execution of the algorithm. We executed 100 times our algorithm from the same point for each different combination of parameters $\alpha \in \{0.01, 0.05, 0.1\}$ and $\tau \in$

$\{5, 9\}$. For each parameters combination, we measured the distance between each trajectory and the trajectory prescribed by the replicator dynamics model. In Fig. 8.4 we reported the replicator dynamics trajectory and the curves containing the 25 closest dynamics (blue curve), the 50 ones (green curve) and the 75 ones (black curve) in the utility space (in the strategy space the graphical representation is not possible). Interestingly, the learning trajectories converge in a neighborhood of the limit point of the replicator dynamics independently of the value of τ. The specific value of τ affects the possibility to converge to (a neighborhood of) an equilibrium. Instead, the variance of the learning process is strictly related to the value of α. If α is very small, then the variance is small. Indeed, when α goes to zero, the learning process is well approximated by its continuous-time replicator dynamics model.

Figure 8.4: Statistical bounds that represent the distance between the expected learning dynamics (red line) and the real dynamics of learning agents that use our algorithm (black line 75%, green line 50%, blue line 25%) for different combinations of parameters α and τ; rows: first $\tau = 5$ and second $\tau = 9$; columns: (a) $\alpha = 0.01$, (b) $\alpha = 0.05$ and (c) $\alpha = 0.1$.

Learning with extensive-form games in heterogeneous populations 9

In many practical situations the study of the learning dynamics may be more important than the study of the equilibrium points: when learning dynamics are long, the overall agents' performance (in terms of expected utility) is strongly conditioned by their performance during the learning dynamics. In this chapter we provide models and tools to forecast these dynamics and allow, in future, a designer to exploit this information in the design of game mechanisms.

In this chapter, we provide four main contributions

- we design some solution concepts extending SCE to capture situations where agents may be of different (Bayesian) finite *types* and *individuals*,
- we study the relationships between different SCE extensions as the number of individuals and types change,
- with two agents, we formulate the problems of finding a solution with finite and infinite populations as two mixed-integer linear mathematical programs (MILP). Furthermore, we show that the problem of verifying whether or not a solution is an SCE extension is in $\mathcal{P}$ even when the input to the verification problem specifies only the strategies and not the beliefs (this is

common when we observe learning agents and we ask whether their strategies are a solution for some given solution concept), while searching for an SCE is $\mathcal{PPAD}$-complete. Furthermore, we discuss to enumerate the equilibria,

- we study the replicator dynamics with multiple individuals and we show how SCEs affect learning dynamics in games with perfect information: they are reachable states only when learning agents are purely greedy, otherwise they are saddle points attracting and then repelling the dynamics. New SCEs emerge as the individuals increase, changing the learning trajectories and delaying their convergence.

In Section 9.1 we introduce the solution concepts extending SCE. In Section 9.2 we characterize the relationship between different SCE extensions. In Section 9.3 we show the main computational results about computing, verifying and enumerating SCEs. In Section 9.4 we study the dynamics of learning agents when applied to extensive-form games. Finally, in Section 9.5 we apply the tools presented in the previous sections to a simple case.

9.1 Solution concepts

Different solution concepts extending the SCE can be provided according to each specific situation, see Tab. 9.1.

		individuals per type		
		1	n	∞
types per agent	1	USCE	FHSCE	IHSCE
	n	BUSCE	BFHSCE	BIHSCE

Table 9.1: Extensions of the SCE concept.

The basic solution concept, introduced in [18], is the *unitary self-confirming equilibrium* (USCE) that captures situations with a unique type per agent and a unique individual per type. A USCE constrains the agent's strategy to be best response to the belief over opponent's strategy and constrains belief to be correct (w.r.t. the strategy) on the equilibrium path.

Definition 9.1 (Unitary self-confirming equilibrium) *A USCE is a pair* $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$ *such that:*

- *each agent i has single type θ and single individual λ;*
- *for every agent i, $\boldsymbol{\sigma}_\lambda$ is a best response to $\hat{\boldsymbol{\sigma}}_\lambda^{-i}$, where λ is the agent i's individual;*
- *for every agent i, $\hat{\boldsymbol{\sigma}}_\lambda^{-i}$ is equal to $\boldsymbol{\sigma}_{\lambda'}$ on the equilibrium path, where λ and λ' are the individuals of agent i and agent $-i$, respectively.*

Upon the USCE concept, we build the solution concepts for other (more complex) situations. We consider the situation with one type per agent and multiple finite individuals: the solution concept is *finite heterogeneous* SCE (FHSCE).

Definition 9.2 (Finite heterogeneous self-confirming equilibrium) *An FHSCE is a pair* $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$ *such that:*

- *each agent i has a single type θ and a finite number of individuals λ;*
- *for every agent i, $\boldsymbol{\sigma}_\lambda$ is a best response to $\hat{\boldsymbol{\sigma}}_\lambda^{-i}$, where λ is an agent i's individual;*
- *for every agent i, $\hat{\boldsymbol{\sigma}}_\lambda^{-i}$ is equal to $\boldsymbol{\sigma}_{-i}$ on the equilibrium path identified by $\boldsymbol{\sigma}_\lambda$ and $\boldsymbol{\sigma}_{-i}$, where λ is an individual of agent i;*
- *for every agent i, $\boldsymbol{\sigma}_i$ is the aggregate strategy of all the agent i's individuals.*

Notice that the constraints over the beliefs of an individual λ and the equilibrium path she observes depend on $\boldsymbol{\sigma}_\lambda$. Thus, different individuals may have different strategies, each supported by different beliefs.

When there is an infinite number of individuals, the above definition is not operative, requiring one to specify an infinite number of strategies and beliefs. In this case, according to [18], we can work at the level of the single action a and we can state that $a \in A_i$ could be played by agent i if there is a belief (confirmed on the equilibrium path) such that this action is a best response. As a result, for every action $a \in A_i$ we need to define a belief $\hat{\boldsymbol{\sigma}}_{i,a}^{-i}$. As done above, we use

$\hat{\boldsymbol{\sigma}}_a^{-i}$ in place of $\hat{\boldsymbol{\sigma}}_{i,a}^{-i}$, under the assumption that each action a has a different label. The solution concept is the *infinite heterogeneous* SCE (IHSCE).

Definition 9.3 (Infinite heterogeneous self-confirming equilibrium) *An IHSCE is a pair* $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$ *such that:*

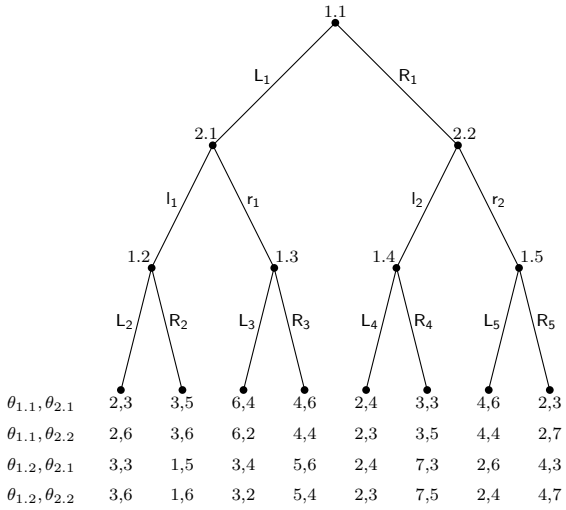
- *each agent* i *has a single type* θ *and an infinite number of individuals* λ*;*
- *for every agent* i, $\sigma_i(a) > 0$ *with* $a \in A_i$ *if* a *is best response to some belief* $\hat{\boldsymbol{\sigma}}_a^{-i}$*;*
- *for every action* $a \in A_i$ *and agent* i, $\hat{\boldsymbol{\sigma}}_a^{-i}$ *equals* $\boldsymbol{\sigma}_{-i}$ *on the equilibrium path identified by* $(\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_{-i})$.

We focus on Bayesian games. We report only the definition of the Bayesian USCE concept (BUSCE) because the redefinitions of FHSCE and IHSCE with Bayesian games (BFHSCE and BIHSCE, respectively) are similar.

Definition 9.4 (Bayesian unitary self-confirming equilibrium) *A BUSCE is a pair* $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$ *such that:*

- *each agent* i *has a finite number of types* θ *and a single individual* λ *per type;*
- *for every agent* i, $\boldsymbol{\sigma}_\lambda$ *is a best response to* $\hat{\boldsymbol{\sigma}}_\lambda^{-i}$*, where* λ *is the agent* i*'s individual and* $\hat{\boldsymbol{\sigma}}_\lambda^{-i}$ *is the aggregate belief of* λ *over agent* $-i$ *defined as* $\sum_{\theta \in \Theta_{-i}} \hat{\boldsymbol{\sigma}}_\lambda^{\theta} \cdot \hat{\omega}_\theta$*;*
- *for every agent* i, $\hat{\boldsymbol{\sigma}}_\lambda^{-i}$ *is equal to* $\boldsymbol{\sigma}_{-i}$ *on the equilibrium path, where* λ *is an individual of agent* i *and* $\boldsymbol{\sigma}_{-i}$ *is the aggregate strategy of agent* $-i$*.*

Finally, we remark that every game admits at least one SCE per extension, since the SCE concept relaxes the NE concept and every game admits at least an NE.



$\theta_{1.1}, \theta_{2.1}$	2,3	3,5	6,4	4,6	2,4	3,3	4,6	2,3
$\theta_{1.1}, \theta_{2.2}$	2,6	3,6	6,2	4,4	2,3	3,5	4,4	2,7
$\theta_{1.2}, \theta_{2.1}$	3,3	1,5	3,4	5,6	2,4	7,3	2,6	4,3
$\theta_{1.2}, \theta_{2.2}$	3,6	1,6	3,2	5,4	2,3	7,5	2,4	4,7

Figure 9.1: Running example: game tree and four different (independent) type profiles.

9.2 Equilibrium characterization

We characterize the relation between different SCE extensions with the aim of showing the expressivity of the model. For clarity, we use the following example.

Example 9.1 *In the game in Fig. 9.1, there are two agents, each with two types. Fig. 9.2.a and Fig. 9.2.b report the equilibria in the space of the agents' utilities when types are only $\theta_{1.1}, \theta_{2.1}$: A, B, C, D, and all the points between B and C are USCEs, while A is the unique SPE. Fig. 9.2.c and Fig. 9.2.d report the equilibria when types are only $\theta_{1.2}, \theta_{2.1}$: A, B, C, and all the points between B and C are USCEs, while A and all the points between D and C are SPEs.*

While it is known that with two agents USCEs and NEs are the same in terms of expected utility, see [23], the presence of individuals makes HSCEs (both FHSCEs and IHSCEs) and NEs dramatically different. We state the following, whose proof is given as counterexample below.

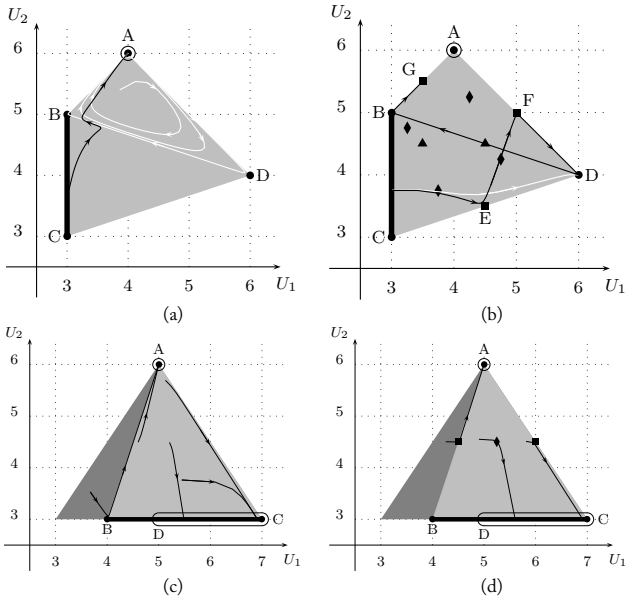


Figure 9.2: USCEs (at left) and FHSCEs (at right) in the running example of Fig. 9.1. The light gray polytopes contain the IHSCEs. The gray polytopes contain all solutions except for never-best response (trajectories are discussed in the following sections).

Proposition 9.1 *Types and individuals have a different expressivity.*

Example 9.2 *Different types can act in different ways only if their utilities are different, instead different individuals can act in different ways even with the same utility leading thus to new equilibria. In Fig. 9.2 the light gray polytopes contain all the IHSCEs. In Fig. 9.2.b and in Fig. 9.2.d, we report the FHSCEs with different combinations of individuals per agent (in this case U_1 and U_2 are the aggregate expected utilities of agent 1 and 2, respectively). More precisely, ■ marks FHSCEs (non-USCE) when agent 1 has two individuals (with $\omega_\lambda = .5$) and agent 2 has one individual; ▲ marks FHSCEs (non-USCE) when agent 1 has one individual and agent 2 has two individuals (with $\omega_\lambda = .5$). When both agents have two individuals (with $\omega_\lambda = .5$) new FHSCEs arise marked by ⧫.*

Thus, the presence of individuals introduces new equilibria. More importantly, although the number of FHSCEs increases as the number of individuals increases, IHSCEs represent a coarse approximation of FHSCEs when the number of individuals is small. This pushes for the need for FHSCEs.

Now, we study the relationships between solution concepts with a unitary individual and solution concepts with heterogeneous individuals, showing that the latter solution concepts are not easy generalization of the former ones. We now provide the following definition.

Definition 9.5 (Convex combination) *The* convex combination *of two USCEs* $(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2)$, $(\boldsymbol{\sigma}'_1, \boldsymbol{\sigma}'_2)$ *is defined as* $(\alpha \cdot \boldsymbol{\sigma}_1 + (1-\alpha) \cdot \boldsymbol{\sigma}'_1, \alpha \cdot \boldsymbol{\sigma}_2 + (1-\alpha) \cdot \boldsymbol{\sigma}'_2)$ *with* $\alpha \in [0,1]$.

We can show that FHSCEs and IHSCEs are convex combinations of different USCEs. We state the following theorem for the IHSCE concept, the same result can be derived for the FHSCE concept.

Theorem 9.1 *Every IHSCE is equivalent to a probability distribution over the set of USCEs.*

Proof. Given an IHSCE $(\mathbf{x}_1, \mathbf{x}_2)$, we partition the set of terminal sequences $q \in \overline{Q}_i$ such that $x_i(q) > 0$ by using the following equivalent relation: $[q] = \{q' \in \overline{Q}_i | \hat{\mathbf{x}}_{q'}^{-i} = \hat{\mathbf{x}}_q^{-i}\}$. For each equivalent class, we normalize the probability $\mathbf{x}_i$ such that $x_i^*(\mathsf{q}_\varnothing) = 1$ where $\mathbf{x}_i^*$ is the normalized probability and $\mathsf{q}_\varnothing$ is the empty sequence. We obtain: $x_i^*(q) = \frac{x_i(q)}{\sum_{q' \in [q]} x_i(q')}$ for every $q \in [q]$. Each equivalent class is a USCE because: (*a*) each $q \in [q]$ is best-response given $\hat{\mathbf{x}}_q^{-i}$ (by IHSCE definition) and (*b*) there exists a unique $\hat{\mathbf{x}}_q^{-i}$, $\forall q \in [q]$ (by partition rule). We define a probability distribution over the set of $[q]$ such that: $Pr([q]) = \sum_{q' \in [q]} x_i(q')$. Notice that $x_i(q) = x_i^*(q) \cdot Pr([q])$ for every $q \in \overline{Q}_i$. □

Theorem 9.2 *Every BIHSCE is equivalent to a probability distribution over the set of BUSCEs.*

With infinite populations, IHSCEs can be every strategy profile $(\sum_k \alpha_{1,k} \cdot \boldsymbol{\sigma}_{1,k}, \sum_k \alpha_{2,k} \cdot \boldsymbol{\sigma}_{2,k})$ with $\sum_k \alpha_{i,k} = 1$ and $\alpha_{i,k} \geq 0$, and

where $\boldsymbol{\sigma}_{i,k}$ is the strategy of agent i in the k-th USCE. Therefore, once we have all the USCEs, we can derive all the IHSCEs simply as convex combinations of USCEs. In the finite case, the set of possible FHSCEs is limited to a finite set of strategies, that are obtained by constraining the coefficients of the convex combination as follows: $\alpha_{i,k} = \sum_{\boldsymbol{\sigma}_\lambda = \boldsymbol{\sigma}_{i,k}} \omega_\lambda$.

We state the following theorem for the FHSCE concept, whose proof is given as counterexample below.

Theorem 9.3 *Given a finite set of individuals, a convex combination of USCEs feasible w.r.t. probabilities ω_λ may be not an FHSCE.*

Example 9.3 *In Fig. 9.2.b, some possible convex combinations, e.g., the ones between A and C, do not correspond to any FHSCE. In Fig. 9.2.d, the unique two convex combinations that correspond to FHSCEs are between A and C and between B and D.*

Thus, differently from IHSCEs, FHSCEs cannot be directly derived from the enumeration of USCEs.

9.3 Computational results

We study the problems of computing an equilibrium according to the solution concepts defined above, of verifying whether a given solution is one of such solution concepts, and of enumerating the equilibria.

Equilibrium computation. The solution concept definitions provided in Section 9.1 are based on behavioral strategies, but it is known that behavioral strategies pose severe computational issues [59]. However, since those constraints are defined exclusively on the equilibrium path, it is possible to redefine them in terms of sequence-form strategies without perturbations (as instead it is needed for SEs to capture the off-the-equilibrium-path behavior). As a result, we have $\mathbf{x}_\lambda$, $\hat{\mathbf{x}}_\lambda^{-i}$, $\hat{\mathbf{x}}_\lambda^{\theta'}$ for SCEs with finite populations, and $\mathbf{x}_\theta$, $\mathbf{x}_{\theta,q}$, $\hat{\mathbf{x}}_{\theta,q}^{-i}$, $\hat{\mathbf{x}}_{\theta,q}^{\theta'}$ for SCEs with infinite populations.

A specific sample of SCE extensions can be computed by searching for an NE and thus it can be solved by using the Lemke algorithm applied to the sequence form (in the case all the individuals have correct beliefs, all the individuals of the same type behave in the same way and therefore the game is essentially a Bayesian two-agent game). Hence, the problem of computing an SCE extension

with two agents is $\mathcal{PPAD}$-complete. However, the computation of a specific sample is not interesting when dealing with learning agents that can potentially achieve any equilibrium, while the aim is the derivation of the equilibrium conditions and the enumeration of all the equilibria. Indeed, knowing all the equilibria and their stability properties in terms of attractors, saddle points, and repellers, we can have a complete picture about all the possible learning dynamics of the agents.

The constraint expressing that a belief is confirmed on the equilibrium path is not linear and neither expressible as a linear complementarity constraint (instead it can be formulate as a non-linear complementary constraint). Indeed, the constraint over the belief on a sequence (or an action indifferently) must be active only if such sequence (or action) is on the equilibrium path. More precisely, sequence $q|a \in Q_{-i}$ is on the equilibrium path if both $q|a$ is played with strictly positive probability and there is at least a sequence $q' \in Q_i$ leading to $h \in H_{-i}$, where $a \in \rho(h)$, that is played with strictly positive probability. Given sequence $q|a \in Q_{-i}$, we denote by $f(q|a) \subseteq Q_i$ the set of sequences leading to $h \in H_{-i}$ such that $a \in \rho(h)$. A possible way to code the above constraint with USCEs is: $x_\lambda(q') \cdot x_{\lambda'}(q|a) \cdot (\hat{x}_\lambda^{-i}(q|a) - x_{\lambda'}(q|a)) = 0$ for every $q' \in f(q|a)$, where λ is an individual of agent i and λ' is an individual of agent $-i$. These constraints can be formulated as non-linear complementarity constraints. However, non-linear complementarity programming does not allow the computation of exact solutions, but only approximate ones. Thus, we resort to mixed-integer linear programming to linearize such non-linear constraints by using binary variables.

We define the binary variables as $s_\lambda(q)$ such that $s_\lambda(q) = 1$ when $q \in Q_i$ is played with positive probability by λ. Similarly, we use binary variables $s_\theta(q)$ such that $s_\theta(q) = 1$ when q is played with positive probability by at least one individual of θ, and we use binary variables $s_i(q)$ such that $s_i(q) = 1$ when q is played with positive probability by at least one individual of i. The previous non-linear constraint can be formulated as a pair of constraints: $\hat{x}_\lambda^{-i}(q|a) - x_{\lambda'}(q|a) \leq M \cdot (2 - s_\lambda(q') - s_{\lambda'}(q|a))$ and $x_{\lambda'}(q|a) - \hat{x}_\lambda^{-i}(q|a) \leq M \cdot (2 - s_\lambda(q') - s_{\lambda'}(q|a))$ for every $q' \in f(q|a)$, where M is an arbitrarily large constant. The problem of computing a BFHSCE when types are not interdependent can be formulated as the following MILP (the program for an FHSCE is a trivial simplification obtained by setting $|\Theta_i| = 1$ for every i):

$$v_\lambda(h) \geq \sum_{h' \in H_i(q|a)} v_\lambda(h') + \sum_{q' \in Q_{-i}} u_i(\theta, q|a, q') \cdot \hat{x}_\lambda^{-i}(q')$$
$$\forall i \in N, q \in Q_i, h \in H_i(q), a \in \rho(h), \theta \in \Theta_i, \lambda \in \Lambda_\theta \quad (9.1)$$

$$v_\lambda(h) \leq \sum_{h' \in H_i(q|a)} v_\lambda(h') +$$
$$\sum_{q' \in Q_{-i}} u_i(\theta, q|a, q') \cdot \hat{x}_\lambda^{-i}(q') + M(1 - s_\lambda(q|a))$$
$$\forall i \in N, q \in Q_i, h \in H_i(q), a \in \rho(h), \theta \in \Theta_i, \lambda \in \Lambda_\theta \quad (9.2)$$

$$\hat{x}_\lambda^{-i}(q|a) \geq \sum_{\lambda' \in \Lambda_{\theta'}, \theta' \in \Theta_{-i}} x_{\lambda'}(q|a) - M \cdot (2 - s_\lambda(q') - s_{-i}(q|a))$$
$$\forall i \in N, q \in Q_{-i}, q' \in f(q|a), \lambda \in \Lambda_\theta, \theta \in \Theta_i \quad (9.3)$$

$$\hat{x}_\lambda^{-i}(q|a) \leq \sum_{\lambda' \in \Lambda_{\theta'}, \theta' \in \Theta_{-i}} x_{\lambda'}(q|a) + M \cdot (2 - s_\lambda(q') - s_{-i}(q|a))$$
$$\forall i \in N, q \in Q_{-i}, q' \in f(q|a), \lambda \in \Lambda_\theta, \theta \in \Theta_i \quad (9.4)$$

$$x_\lambda(\mathsf{q}_\varnothing) = \omega_\lambda \qquad \forall i \in N, \lambda \in \Lambda_\theta, \theta \in \Theta_i \quad (9.5)$$

$$x_\lambda(q) = \sum_{a \in \rho(h)} x_\lambda(q|a) \qquad \forall i \in N, \lambda \in \Lambda_\theta, q \in Q, h \in H_{i,q}, \theta \in \Theta_i \quad (9.6)$$

$$\hat{x}_\lambda^{-i}(\mathsf{q}_\varnothing) = 1 \qquad \forall i \in N, \lambda \in \Lambda_\theta, \theta \in \Theta_i \quad (9.7)$$

$$\hat{x}_\lambda^{-i}(q) = \sum_{a \in \rho(h)} \hat{x}_\lambda^{-i}(q|a) \qquad \forall i \in N, \lambda \in \Lambda_\theta, q \in Q, h \in H_{i,q}, \theta \in \Theta_i \quad (9.8)$$

$$s_\lambda(q) \geq x_\lambda(q) \qquad \forall i \in N, \lambda \in \Lambda_\theta, \theta \in \Theta_i, q, \in Q_i \quad (9.9)$$

$$s_i(q) \geq s_\lambda(q) \qquad \forall i \in N, \lambda \in \Lambda_\theta, q \in Q_i, \theta \in \Theta_i \quad (9.10)$$

$$s_i(q), s_\lambda(q) \in \{0,1\} \qquad \forall i \in N, q \in Q_i, \theta \in \Theta_i, \lambda \in \Lambda_\theta \quad (9.11)$$

$$x_\lambda(q), \hat{x}_\lambda^{-i}(q) \geq 0 \qquad \forall i \in N, q \in Q_i, \theta \in \Theta_i, \lambda \in \Lambda_\theta \quad (9.12)$$

Here, constraints (9.1) and (9.2) force $v_\lambda(h)$ at each h of individual λ to be equal to the expected (w.r.t. the individual's beliefs $\hat{\mathbf{x}}_\lambda^{-i}$) utility of the best sequence available at h (similar constraints are used in MILP Nash formulation [53]); constraints (9.3) and (9.4) force the belief $\hat{x}_\lambda^{-i}(q)$ of individual λ over the aggregate opponent's strategy $\mathbf{x}_{-i}$ to be correct on the equilibrium path (notice that, when $2 - s_\lambda(f(q)) - s_{-i}(q) = 0$, q is on the equilibrium path); constraints (9.5) and (9.6) assure that x_λ is a well defined strategy; constraints (9.7) and (9.8) assure that $\hat{\mathbf{x}}_\lambda^{-i}$ is a well-defined belief; constraints (9.9) assure that $s_\lambda(q) = 1$ if q are played with positive probability by individual λ; constraints (9.10) assure that $s_i(q)$ are strictly positive if at least one individual of i plays q with positive probability; constraints (9.11) and (9.12) define the domains of the variables. Differently from the case of NE or its refinements, both strategies and beliefs are variables.

Notice that the above program keeps to be mixed-integer linear independently of the number of individuals per type. This is because each individual plays the best response to her beliefs, her beliefs are equal to the aggregate opponent's strategy on the equilibrium path, and each agent aggregate strategy is given by the sum of the single individual strategies. In practice, the number of individuals determines the number of constraints and variables present in the program, but it does not affect the linearity of the program.

The MILP for computing a BIHSCE is more complex. For each pair θ, q, we need a strategy $\mathbf{x}_{\theta,q}$ defined over the whole game and the corresponding values $\mathbf{v}_{\theta,q}$ per information set, and we need a belief $\hat{\mathbf{x}}_{\theta,q}^{-i}$ over the aggregate opponent's strategy. The MILP for computing a BIHSCE with non-interdependent types is the following (the program for IHSCE can be obtained with $|\Theta_i| = 1$ for every i):

$$v_{\theta,q|a}(h) \geq \sum_{h' \in H_i(q|a)} v_{\theta,q|a}(h') + \sum_{q' \in Q_{-i}} u_i(\theta, q|a, q') \cdot \hat{x}_{\theta,q|a}^{-i}(q')$$
$$\forall i \in N, q \in Q_i, h \in H_i(q), \theta \in \Theta_i, a \in \rho(h) \quad (9.13)$$

$$v_{\theta,q|a}(h) \leq \sum_{h' \in H_i(q|a)} v_{\theta,q|a}(h') + \sum_{q' \in Q_{-i}} u_i(\theta, q|a, q') \cdot \hat{x}^{-i}_{\theta,q|a}(q') + M(1 - s_{\theta,q|a}(q|a))$$
$$\forall i \in N, q \in Q_i, h \in H_i(q), \theta \in \Theta_i, a \in \rho(h) \quad (9.14)$$

$$\hat{x}^{-i}_{\theta,q}(q'|a) \geq \sum_{\theta' \in \Theta_{-i}} x_{\theta'}(q'|a) - M \cdot (2 - s_{\theta,q}(q'') - s_{-i}(q'|a))$$
$$\forall i \in N, q \in Q_i, q'' \in f(q'|a), q' \in Q_{-i}, \theta \in \Theta_i \quad (9.15)$$

$$\hat{x}^{-i}_{\theta,q}(q'|a) \leq \sum_{\theta' \in \Theta_{-i}} x_{\theta'}(q'|a) + M \cdot (2 - s_{\theta,q}(q'') - s_{-i}(q'|a))$$
$$\forall i \in N, q \in Q_i, q'' \in f(q'|a), q' \in Q_{-i}, \theta \in \Theta_i \quad (9.16)$$

$$x_{\theta,q'}(\mathsf{q}_\varnothing) = \omega_\theta \qquad \forall i \in N, \theta \in \Theta_i, q' \in Q_i \quad (9.17)$$

$$x_{\theta,q'}(q) = \sum_{a \in \rho(h)} x_{\theta,q'}(q|a) \qquad \forall i \in N, \theta \in \Theta_i, q, q' \in Q_i, h \in H_{i,q} \quad (9.18)$$

$$x_\theta(\mathsf{q}_\varnothing) = \omega_\theta \qquad \forall i \in N, \theta \in \Theta_i \quad (9.19)$$

$$x_\theta(q) = \sum_{a \in \rho(h)} x_\theta(q|a) \qquad \forall i \in N, \theta \in \Theta_i, q \in Q_i, h \in H_{i,q} \quad (9.20)$$

$$\hat{x}^{-i}_{\theta,q'}(\mathsf{q}_\varnothing) = 1 \qquad \forall i \in N, \theta \in \Theta_i \quad (9.21)$$

$$\hat{x}^{-i}_{\theta,q'}(q) = \sum_{a \in \rho(h)} \hat{x}^{-i}_{\theta,q'}(q|a) \qquad \forall i \in N, \theta \in \Theta_i, q \in Q_i, h \in H_{i,q} \quad (9.22)$$

$$s_{\theta,q'}(q) \geq x_{\theta,q'}(q) \qquad \forall i \in N, \theta \in \Theta_i, q, q' \in Q_i \quad (9.23)$$

$$s_i(q) \geq s_{\theta,q'}(q) \qquad \forall i \in N, \theta \in \Theta_i, q, q' \in Q_i \quad (9.24)$$

$$s_i(q), s_{\theta,q'}(q) \in \{0,1\} \qquad \forall i \in N, q, q' \in Q_i, \theta \in \Theta_i \quad (9.25)$$

$$x_\theta(q), x_{\theta,q'}(q), \hat{x}^{-i}_{\theta,q'}(q) \geq 0 \qquad \forall i \in N, q, q' \in Q_i, \theta \in \Theta_i \quad (9.26)$$

Here, constraints (9.13) and (9.14) force $v_{\theta,q}(h)$ at each h of type θ related to q to be equal to the expected utility of the best sequence available at h; constraints (9.15) and (9.16) ensure that $\hat{\mathbf{x}}^{-i}_{\theta,q}$ are correct on the equilibrium path identified by $\mathbf{x}_{\theta,q}$ and $\mathbf{x}_{-i}$; constraints (9.17) and (9.18) guarantee that $\mathbf{x}_{\theta,q}$ is well defined and not played with a probability larger than ω_θ; constraints (9.19) and (9.20) make the same with strategy $\mathbf{x}_\theta$; constraints (9.21) and (9.22) ensure that $\hat{\mathbf{x}}^{-i}_{\theta,q}$ are well defined; constraints (9.23) force $s_\theta(q) = 1$ if there is a belief of type θ such that q is a best response; constraints (9.24) force $s_i(q) = 1$ if at least a type of i plays q with positive probability; constraints (9.25) and (9.26) define the domain of the variables.

To compute a BFHSCE with interdependent types we need to introduce a belief $\hat{\mathbf{x}}^{\theta'}_\lambda$ over the strategy of each possible type $\theta' \in \Theta_{-i}$ of the agent i's opponent and a belief $\hat{\omega}_\theta$ over the possible types $\theta \in \Theta_{-i}$ of the opponent. We have also to add an argument to the utility function, that is the type of the opponent. The MILP for computing a BFHSCE with interdependent types is the following:

$$v_\lambda(h) \geq \sum_{h' \in H_i(q|a)} v_\lambda(h') + \sum_{q' \in Q_{-i}} u_i(\theta, \theta', q|a, q') \cdot \hat{x}^{\theta'}_\lambda(q')$$
$$\forall i \in N, q \in Q_i, h \in H_i(q), a \in \rho(h), \theta \in \Theta_i, \lambda \in \Lambda_\theta, \theta' \in \Theta_{-i} \quad (9.27)$$

$$v_\lambda(h) \leq \sum_{h' \in H_i(q|a)} v_\lambda(h') + \sum_{q' \in Q_{-i}} u_i(\theta, \theta', q|a, q') \cdot \hat{x}^{\theta'}_\lambda(q') + M(1 - s_\lambda(q|a))$$
$$\forall i \in N, q \in Q_i, h \in H_i(q), a \in \rho(h), \theta \in \Theta_i, \lambda \in \Lambda_\theta, \theta' \in \Theta_{-i} \quad (9.28)$$

$$\hat{x}_{\lambda}^{\theta'}(q|a) \geq x_{\lambda'}(q|a) - M \cdot (2 - s_{\lambda}(q') - s_{\lambda'}(q|a))$$
$$\forall i \in N, q \in Q_{-i}, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i, \lambda' \in \Lambda_{\theta'}, \theta' \in \Theta_{-i}, q' \in f(q|a) \tag{9.29}$$

$$\hat{x}_{\lambda}^{\theta'}(q|a) \leq x_{\lambda'}(q|a) + M \cdot (2 - s_{\lambda}(q') - s_{\lambda'}(q|a))$$
$$\forall i \in N, q \in Q_{-i}, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i, \lambda' \in \Lambda_{\theta'}, \theta' \in \Theta_{-i}, q' \in f(q|a) \tag{9.30}$$

$$x_{\lambda}(\mathsf{q}_{\varnothing}) = \omega_{\lambda} \qquad \forall i \in N, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i \tag{9.31}$$

$$x_{\lambda}(q) = \sum_{a \in \rho(h)} x_{\lambda}(q|a) \qquad \forall i \in N, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i, q \in Q, h \in H_{i,q} \tag{9.32}$$

$$\sum_{\theta \in \Theta_i} \hat{\omega}_{\theta} = 1 \qquad \forall i \in N \tag{9.33}$$

$$\hat{x}_{\lambda}^{\theta'}(\mathsf{q}_{\varnothing}) = \hat{\omega}_{\theta} \qquad \forall i \in N, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i, \boldsymbol{\theta'} \in \Theta_{-i} \tag{9.34}$$

$$\hat{x}_{\lambda}^{\theta'}(q) = \sum_{a \in \rho(h)} \hat{x}_{\lambda}^{\theta'}(q|a)$$
$$\forall i \in N, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i, q \in Q, h \in H_{i,q}, \theta' \in \Theta_{-i} \tag{9.35}$$

$$s_{\lambda}(q) \geq x_{\lambda}(q) \qquad \forall i \in N, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i, q, \in Q_i \tag{9.36}$$

$$s_{\lambda}(q) \in \{0, 1\} \qquad \forall i \in N, q \in Q_i, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i \tag{9.37}$$

$$x_{\lambda}(q), \hat{x}_{\lambda}^{\theta'}(q), \hat{\omega}_{\theta} \geq 0 \qquad \forall i \in N, q \in Q_i, \lambda \in \Lambda_{\theta}, \theta \in \Theta_i, \theta' \in \Theta_{-i} \tag{9.38}$$

Here, constraints (9.27) and (9.28) force each individual to plays her best response to her beliefs; constraints (9.29) and (9.30) assure that the beliefs are correct over the equilibrium path; constraints (9.31)

and (9.32) assure the strategy of each individual λ to be well defined; constraints (9.33) assure that the aggregate strategy of each agent is well defined; constraints (9.34) and (9.35) assure that the beliefs of each individual is well defined; constraints (9.36) assure that the binary variable $s(q)$ are equal to one if q is played with strict positive probability; constraints (9.37) and (9.38) define the domains of the variables.

To compute a BIHSCE with interdependent types we need to introduce a belief $\hat{\mathbf{x}}^{\theta'}_{\theta,q|a}$ over the strategy of each possible type $\theta' \in \Theta_{-i}$ of the agent i's opponent. As in the case of BFHSCE we have a belief $\hat{\omega}_\theta$ over the possible types $\theta \in \Theta_{-i}$ of the opponent and we have the type of the opponent as argument to the utility function. The MILP for computing a BIHSCE with interdependent types is the following:

$$v_{\theta,q|a}(h) \geq \sum_{h' \in H_i(q|a)} v_{\theta,q|a}(h') + \sum_{q' \in Q_{-i}} u_i(\theta, \theta', q|a, q') \cdot \hat{x}^{\theta'}_{\theta,q|a}(q')$$
$$\forall i \in N, q \in Q_i, h \in H_i(q), a \in \rho(h), \theta \in \Theta_i, \theta' \in \Theta_{-i} \quad (9.39)$$

$$v_{\theta,q|a}(h) \leq \sum_{h' \in H_i(q|a)} v_{\theta,q|a}(h') +$$
$$\sum_{q' \in Q_{-i}} u_i(\theta, \theta', q|a, q') \cdot \hat{x}^{\theta'}_{\theta,q|a}(q') + M(1 - s_{\theta,q|a}(q|a))$$
$$\forall i \in N, q \in Q_i, h \in H_i(q), a \in \rho(h), \theta \in \Theta_i, \theta' \in \Theta_{-i} \quad (9.40)$$

$$\hat{x}^{\theta'}_{\theta,q}(q'|a) \geq x_{\theta'}(q'|a) - M \cdot (2 - s_{\theta,q}(q'') - s_{\theta'}(q'|a))$$
$$\forall i \in N, q \in Q_i, q' \in Q_{-i}, q'' \in f(q'|a), \theta \in \Theta_i, \theta' \in \Theta_{-i} \quad (9.41)$$

$$\hat{x}^{\theta'}_{\theta,q}(q'|a) \leq x_{\theta'}(q'|a) + M \cdot (2 - s_{\theta,q}(q'') - s_{\theta'}(q'|a))$$
$$\forall i \in N, q \in Q_i, q' \in Q_{-i}, q'' \in f(q'|a), \theta \in \Theta_i, \theta' \in \Theta_{-i} \quad (9.42)$$

$$x_{\theta,q'}(\mathsf{q}_\varnothing) = \omega_\theta \qquad \forall i \in N, \theta \in \Theta_i, q' \in Q_i \quad (9.43)$$

$$x_{\theta,q'}(q) = \sum_{a \in \rho(h)} x_{\theta,q'}(q|a) \qquad \forall i \in N, \theta \in \Theta_i, q, q' \in Q_i, h \in H_{i,q} \quad (9.44)$$

$$x_\theta(\mathsf{q}_\varnothing) = \omega_\theta \qquad \forall i \in N, \theta \in \Theta_i \quad (9.45)$$

$$x_\theta(q) = \sum_{a \in \rho(h)} x_\theta(q|a) \qquad \forall i \in N, \theta \in \Theta_i, q \in Q_i, h \in H_{i,q} \quad (9.46)$$

$$\sum_{\theta \in \Theta_i} \hat{\omega}_\theta = 1 \qquad \forall i \in N \quad (9.47)$$

$$\hat{x}^{\theta'}_{\theta,q'}(\mathsf{q}_\varnothing) = \hat{\omega}_{\theta'} \qquad \forall i \in N, \theta \in \Theta_i, \theta' \in \Theta_{-i} \quad (9.48)$$

$$\hat{x}^{\theta'}_{\theta,q'}(q) = \sum_{a \in \rho(h)} \hat{x}^{\theta'}_{\theta,q'}(q|a) \qquad \forall i \in N, \theta \in \Theta_i, \theta' \in \Theta_{-i}, q \in Q_i, h \in H_{i,q} \quad (9.49)$$

$$s_{\theta,q'}(q) \geq x_{\theta,q'}(q) \qquad \forall i \in N, \theta \in \Theta_i, q, q' \in Q_i \quad (9.50)$$

$$s_\theta(q) \geq s_{\theta,q'}(q) \qquad \forall i \in N, \theta \in \Theta_i, q, q' \in Q_i \quad (9.51)$$

$$s_\theta(q), s_{\theta,q'}(q) \in \{0,1\} \qquad \forall i \in N, q, q' \in Q_i, \theta \in \Theta_i \quad (9.52)$$

$$x_\theta(q), x_{\theta,q'}(q), \hat{x}^{\theta'}_{\theta,q'}(q), \hat{\omega}_\theta \geq 0 \qquad \forall i \in N, q, q' \in Q_i, \theta \in \Theta_i, \theta' \in \Theta_{-i} \quad (9.53)$$

Here, constraints (9.39) and (9.40) force that the strategy parameterized to each sequence is the best response to associated beliefs; constraints (9.41) and (9.42) assure that the beliefs are correct over the equilibrium path; constraints (9.43) and (9.44) assure the strategy parameterized to each sequence to be well defined; constraints (9.45) and (9.46) make the same with the aggregate strategy of each type; constraints (9.47), (9.48), and (9.49) assure the beliefs to be well defined; constraints (9.50) and (9.51) assure that the binary variable $s(q)$ are equal to one if q is played with strict positive probability; constraints (9.52) and (9.53) define the domains of the variables.

Equilibrium verification. We focus on the problem of verifying whether a given solution is an SCE extension.

Proposition 9.2 *The problem of verifying whether or not a solution* $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$ *(even expressed in sequence form) is one of the above SCE extensions is in* $\mathcal{P}$.

Proof. The verification problem requires one to check whether or not the pertinent above constraints are satisfied. Being the number of those constraints polynomial in the size of the game, the verification problem can be solved in polynomial time. □

More interesting is the verification problem when the input solution is partially specified, the beliefs being omitted. This problem is common when we question whether learning dynamics, e.g. approaching some attractor, are converging to some solution concepts.

Theorem 9.4 *The problem of verifying whether or not there is a solution* $(\boldsymbol{\sigma}, \hat{\boldsymbol{\sigma}})$ *with a given strategy profile* $\boldsymbol{\sigma}$ *(even expressed in sequence form) that is one of the above SCE extensions is in* $\mathcal{P}$.

Proof. We focus on FHSCE (the proof for the other SCE extensions is similar). The verification problem when the input specifies only the strategies is equivalent to the problem, for each individual λ, to search for a belief confirmed on the equilibrium path such that the strategy of λ is best response. This problem is an overconstrained version (with a linear number of additional constraints) of the problem to verify whether or not a strategy is a never-best response, where the additional constraints are linear constraints. Since the problem to verify whether a strategy is a never-best response can be formulated as a linear mathematical programming problem and the additional constraints are linear, the resulting program is linear and therefore solvable with polynomial time. □

Summarily, these results show that finding and verifying an SCE extension is (in the worst case) as hard as finding an NE [21, 47].

Equilibrium enumeration. The enumeration of the USCEs can be performed by enumerating all the basic solutions of program (9.1)-(9.12) setting $|\Lambda_1| = |\Lambda_2| = 1$. This can be performed by exhaustively visiting the branch-and-bound tree generated by the MILP solver. The size of the tree is $O(2^{|Q_1|+|Q_2|})$. The problem is $\#\mathcal{P}$ and, in the case of degeneracy, the same approach adopted in [3] can be employed here. By the enumeration of the USCEs, we are

enumerating all the IHSCEs, given that these can be derived as convex combinations of USCEs as shown in Section 9.2. The enumeration of FHSCEs can be accomplished similarly: enumerating all the USCEs, computing all the possible finite (with the given ω) convex combinations, and then verifying whether each convex combination is an FHSCE, checking, according to the results discussed in Section 9.2, whether it satisfies program (9.1)-(9.12). Denoting with $\#USCEs$ the number of USCEs, this approach has a complexity $O(2^{|Q_1|+|Q_2|} + 2^{\log_2(\#USCEs)\cdot(|\Lambda_1|+|\Lambda_2|)})$ much faster than enumerating the basic solutions of the mathematical program for FHSCE when $\log_2(\#USCEs)$ is smaller than $|Q_1|$ and $|Q_2|$, this last problem having complexity $O(2^{|Q_1|\cdot|\Lambda_1|+|Q_2|\cdot|\Lambda_2|})$.

9.4 SCE and learning dynamics

In order to formally study the dynamics of learning agents when applied to an extensive-form game, we focus on the replicator dynamics with mutation simulating the (ϵ-greedy exploration/exploitation) Q-learning algorithm [64]. In the previous sections, we showed that the presence of individuals increases the number of equilibria (SCEs) w.r.t. the case without individuals (only NEs). Similarly, here, we can show that the presence of individuals dramatically conditions the learning dynamics.

Initially, we provide the appropriate replicator dynamics when there are multiple individuals. We start the replicator dynamics for sequence form developed in Section 7.3. Given that each individual can learn independently of the others we obtain the following replicator dynamics:

$$\dot{x}_\lambda(q,t) = x_\lambda(q,t)\cdot\left[\sum_{q_1\in Q_1}\sum_{q_2\in Q_2}\left(\sum_{\lambda'\in\Lambda_2} x_{\lambda'}(q_2,t)\cdot u_1(q,q_2)\cdot g_q(q_1,t)\right)\right. - \left.\sum_{q_1\in Q_1}\sum_{q_2\in Q_2}\left(\sum_{\lambda'\in\Lambda_2} x_{\lambda'}(q_2,t)\cdot u_1(q_1,q_2)\cdot x_\lambda(q_1,t)\right)\right] \quad \forall \lambda\in\Lambda_1, q\in Q_1 \quad (9.54)$$

$$\dot{x}_\lambda(q,t) =$$
$$x_\lambda(q,t)\cdot\left[\sum_{q_1\in Q_1}\sum_{q_2\in Q_2}\left(\sum_{\lambda'\in\Lambda_1} x_{\lambda'}(q_1,t)\cdot u_2(q_1,q)\cdot g_q(q_2,t)\right)\right.$$
$$\left.-\sum_{q_1\in Q_1}\sum_{q_2\in Q_2}\left(\sum_{\lambda'\in\Lambda_1} x_{\lambda'}(q_1,t)\cdot u_2(q_1,q_2)\cdot x_\lambda(q_2,t)\right)\right]$$
$$\forall\lambda\in\Lambda_2, q\in Q_2 \quad (9.55)$$

In (9.54)-(9.55) we have an equation for every individual of each population that represents the dynamics of the single individual when he plays independently of the others individuals of the same population against the aggregate population of the opponent. It can be observed that only FHSCEs are rest points of (9.54)-(9.55), the derivatives over all the actions of each individual being zero.

We briefly show that FHSCEs are not rest points without multiple individuals.

Proposition 9.3 *FHSCEs are not steady states without multiple individuals.*

Proof. Consider the game described in Fig. 9.1 where the types are $\theta_{1.1}, \theta_{2.1}$. Suppose that there is only one individual for agent 1 and four individuals for agent 2. The strategies of the individuals are

$$\mathbf{x}_1 = (x_1(\mathsf{L}_1) = x_1(\mathsf{L}_1\mathsf{R}_2) = x_1(\mathsf{L}_1\mathsf{L}_3))$$
$$\mathbf{x}_{2.1} = (x_{2.1}(\mathsf{l}_1) = x_{2.1}(\mathsf{r}_1) = 1)$$
$$\mathbf{x}_{2.2} = (x_{2.2}(\mathsf{l}_1) = x_{2.2}(\mathsf{r}_2) = 1)$$
$$\mathbf{x}_{2.3} = (x_{2.3}(\mathsf{l}_2) = x_{2.3}(\mathsf{r}_1) = 1)$$
$$\mathbf{x}_{2.4} = (x_{2.4}(\mathsf{l}_2) = x_{2.4}(\mathsf{r}_2) = 1)$$

where $\mathbf{x}_{x.y}$ denote the strategy of y-th individual of x-agent. Now suppose that the individual of agent 2 are drawn from a uniform distribution, the aggregate strategy is

$$\mathbf{x}_2 = \left(x_2(\mathsf{l}_1) = x_2(\mathsf{l}_2) = x_2(\mathsf{r}_1) = x_2(\mathsf{r}_2) = \frac{1}{2}\right) \quad (9.56)$$

It is easy to show that the strategy of each individual of agent 2 is the best response w.r.t. some beliefs about the opponent and the strategy of agent 1 is the best response to (9.56). So $(\mathbf{x}_1, \mathbf{x}_2)$ is a FHSCE. But in (9.54)-(9.55) when $|\Lambda_1| = |\Lambda_2| = 1$ the strategy profile $(\mathbf{x}_1, \mathbf{x}_2)$ is not a rest point. □

In [33], the author shows that, without individuals, only SPEs are asymptotically stable states for the replicator dynamics with (small) mutation when games are with perfect information. Therefore, learning agents adopting Q-learning algorithm can converge only to SPEs (interestingly, non-degenerate games admit only one SPE). We can extend such a result to the case with multiple individuals.

Proposition 9.4 *Only SPEs are asymptotically stable states of the replicator dynamics with (small) mutation even when agents can have multiple individuals.*

Proof sketch. The proof follows from [33], where the author shows that mutation corresponds to perturbation and all the NEs that are not SPEs are weakly dominated and therefore they cannot be asymptotically stable. The same reasoning holds here, where all the FHSCEs that are not SPEs are weakly dominated and therefore they cannot be asymptotically stable. □

Therefore, all the FHSCEs that are not SPEs (due to the presence of multiple individuals) are saddles, attracting the learning dynamics and then repelling them. Given that FHSCEs are the only steady states of the (9.54)-(9.55), we can completely characterize the learning dynamics by studying their dynamical properties. We can achieve that enumerating all the FHSCEs, as described in the previous section, and evaluating the eigenvalues of the Jacobian J of (9.54)-(9.55):

$$J = \begin{bmatrix} \frac{\partial \dot{\mathbf{x}}_{\lambda_1}}{\partial \mathbf{x}_{\lambda_1}} & \cdots & \frac{\partial \dot{\mathbf{x}}_{\lambda_1}}{\partial \mathbf{x}_{\lambda_n}} & \frac{\partial \dot{\mathbf{x}}_{\lambda_1}}{\partial \mathbf{x}_{\lambda'_1}} & \cdots & \frac{\partial \dot{\mathbf{x}}_{\lambda_1}}{\partial \mathbf{x}_{\lambda'_m}} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial \dot{\mathbf{x}}_{\lambda_n}}{\partial \mathbf{x}_{\lambda_1}} & \cdots & \frac{\partial \dot{\mathbf{x}}_{\lambda_n}}{\partial \mathbf{x}_{\lambda_n}} & \frac{\partial \dot{\mathbf{x}}_{\lambda_n}}{\partial \mathbf{x}_{\lambda'_1}} & \cdots & \frac{\partial \dot{\mathbf{x}}_{\lambda_n}}{\partial \mathbf{x}_{\lambda'_m}} \\ & & & & & \\ \frac{\partial \dot{\mathbf{x}}_{\lambda'_1}}{\partial \mathbf{x}_{\lambda_1}} & \cdots & \frac{\partial \dot{\mathbf{x}}_{\lambda'_1}}{\partial \mathbf{x}_{\lambda_n}} & \frac{\partial \dot{\mathbf{x}}_{\lambda'_1}}{\partial \mathbf{x}_{\lambda'_1}} & \cdots & \frac{\partial \dot{\mathbf{x}}_{\lambda'_1}}{\partial \mathbf{x}_{\lambda'_m}} \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{\partial \dot{\mathbf{x}}_{\lambda'_m}}{\partial \mathbf{x}_{\lambda_1}} & \cdots & \frac{\partial \dot{\mathbf{x}}_{\lambda'_m}}{\partial \mathbf{x}_{\lambda_n}} & \frac{\partial \dot{\mathbf{x}}_{\lambda'_m}}{\partial \mathbf{x}_{\lambda'_1}} & \cdots & \frac{\partial \dot{\mathbf{x}}_{\lambda'_m}}{\partial \mathbf{x}_{\lambda'_m}} \end{bmatrix}$$

with $\lambda_1, \ldots, \lambda_n \in \Lambda_1, \lambda'_1, \ldots, \lambda'_m \in \Lambda_2$, where

$$\frac{\partial \dot{\mathbf{x}}_{\lambda_i}}{\partial \mathbf{x}_{\lambda_j}} = \begin{cases} \begin{bmatrix} \frac{\partial \dot{x}_{\lambda_i}(a_1)}{\partial x_{\lambda_j}(a_1)} & \cdots & \frac{\partial \dot{x}_{\lambda_i}(a_1)}{\partial x_{\lambda_j}(a_n)} \\ \vdots & & \vdots \\ \frac{\partial \dot{x}_{\lambda_i}(a_m)}{\partial x_{\lambda_j}(a_1)} & \cdots & \frac{\partial \dot{x}_{\lambda_i}(a_m)}{\partial x_{\lambda_j}(a_n)} \end{bmatrix} & \text{if } \lambda_i, \lambda_j \in \Lambda_k \text{ and } i = j \text{ or if } \lambda_i \in \Lambda_k, \lambda_j \in \Lambda_{k'}, k \neq k' \\ 0 & \text{if } \lambda_i, \lambda_j \in \Lambda_k \text{ and } i \neq j \end{cases}$$

If all the eigenvalues are negative, the rest point is an attractor, if there are positive and negative eigenvalues, it is a saddle, and in the case where all of them are positive it is a repeller. Given the eigenvalues we can compute the eigenvectors that represent the directions of the dynamics of the system close to the points.

9.5 A simple case study

We apply the tools described in the previous sections to the game described in Fig. 9.1, where the types are $\theta_{1.1}, \theta_{2.1}$. More precisely,

- we enumerate all the FHSCEs (with different settings of individuals per agent, see below),

- we evaluate the eigenvalues and eigenvectors of the replicator dynamics at each FHSCE,
- we study the dynamics as the starting point changes.

One individual per agent. Initially, we consider the setting with only one individual per agent to show in the following how the presence of individuals affects the dynamics. Fig. 9.2.a displays two learning trajectories projected on the utility space. (Notice that such representation may hide some information since we are interested to study the stability properties of strategy profiles that may map into the same point of the utility space.) Tab. 9.2 reports the eigenvalues associated with each USCEs, where $x_{|y|}$ denotes an eigenvalue x with multiplicity y. The strategy profile $x_1(\mathsf{R}_1) = x_1(\mathsf{R}_1\mathsf{R}_4) = x_1(\mathsf{R}_1\mathsf{L}_5) = 1, x_2(\mathsf{l}_1) = x_2(\mathsf{r}_2) = 1$ is the only SPE, it has all negative eigenvalues. Point A is a stable state, instead the other USCEs, B, C and D are saddle points because they have also positive eigenvalues. From the study of the eigenvectors (omitted for reasons of space), it is possible to observe that the dynamics (in black) starting close to B-C are attracted by D, then repulsed and attracted by B, then repulsed and, finally, attracted by A. Other trajectories are reported in white. Consider Fig. 9.2.c. The trajectory starting outside the convex hull of USCEs quickly moves into such region, and, after being attracted by B, it converges to A.

	Strategies		Eigenvalues	
	Agent 1	Agent 2	Agent 1	Agent 2
A	$\mathsf{R}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$	$\mathsf{l}_1\mathsf{r}_2$	$-2_{\lvert 16\rvert}, -1_{\lvert 8\rvert}, 0_{\lvert 7\rvert}$	$-3_{\lvert 2\rvert}, 0_{\lvert 1\rvert}$
B	$\mathsf{L}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$	$\mathsf{l}_1\mathsf{r}_2$	$-1_{\lvert 16\rvert}, 0_{\lvert 7\rvert}, 1_{\lvert 8\rvert}$	$-1_{\lvert 2\rvert}, 0_{\lvert 1\rvert}$
C	$\mathsf{R}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$	$\mathsf{l}_1\mathsf{l}_2$	$-1_{\lvert 16\rvert}, 0_{\lvert 15\rvert}$	$0_{\lvert 1\rvert}, 3_{\lvert 2\rvert}$
D	$\mathsf{L}_1\mathsf{L}_2\mathsf{L}_3\mathsf{L}_4\mathsf{L}_5$	$\mathsf{r}_1\mathsf{l}_2$	$-4_{\lvert 8\rvert}, -3_{\lvert 8\rvert}, -2_{\lvert 8\rvert}, 0_{\lvert 7\rvert}$	$1_{\lvert 2\rvert}, 0_{\lvert 1\rvert}$

Table 9.2: Eigenvalues of USCEs for the game described in Fig. 9.1 where the types are $\theta_{1.1}, \theta_{2.1}$.

Multiple individuals per agent. We consider the setting with two individuals per agent where each individual has a probability of 0.5. Eigenvalues (only of the new equilibria E, F, G) are reported in Tab. 9.3: they present both positive and negative values, being saddles. Initially, we remove mutation (corresponding to turn off explo-

ration). As shown in Fig. 9.2.b, FHSCEs (D and G) can be asymptotically stable states, starting from opportune strategy profiles. More interestingly, the learning dynamics are dramatically conditioned by the presence of FHSCEs. The dynamics (in black) starting close to B-C are attracted by E and then repulsed, and so on by F, D, B, until G is achieved. With small mutation (exploration), we have very similar dynamics except that they continue to A, without stopping at G. However, long time (about 30%) is spent on strategies very close to FHSCEs and therefore, if the learning rate is sent to zero, the dynamics can stop to FHSCEs even when mutation is present. The figure reports also different dynamics, in white, that, without mutation, stop in D (the dynamics are conditioned by the saddle E). Similar results can be observed by comparing the trajectories depicted in Fig. 9.2.d and Fig. 9.2.c.

	Strategies		Eigenvalues	
	Agent 1	Agent 2	Agent 1	Agent 2
E	$\mathsf{L}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$ $\mathsf{R}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$	$\mathsf{r}_1\mathsf{l}_2$	$-4_{\|8\|}, -3_{\|8\|} - 2_{\|8\|},$ $-1_{\|8\|}, 0_{\|14\|}, 1_{\|8\|}, 3_{\|8\|}$	$\frac{1}{2}_{\|1\|}, \frac{3}{2}_{\|1\|}, 2_{\|1\|}$
F	$\mathsf{L}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$ $\mathsf{R}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$	$\mathsf{r}_1\mathsf{r}_2$	$-4_{\|8\|}, -2_{\|32\|}, -1_{\|8\|},$ $0_{\|14\|}$	$-\frac{3}{2}_{\|1\|}, -1_{\|1\|},$ $\frac{1}{2}_{\|1\|}$
G	$\mathsf{L}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$ $\mathsf{R}_1\mathsf{R}_2\mathsf{L}_3\mathsf{R}_4\mathsf{L}_5$	$\mathsf{l}_1\mathsf{r}_2$	$-2_{\|16\|}, -1_{\|24\|}, 0_{\|14\|},$ $1_{\|8\|}$	$-2_{\|1\|}, -\frac{3}{2}_{\|1\|},$ $-\frac{1}{2}_{\|1\|}$

Table 9.3: Eigenvalues of FHSCEs of game described in Fig. 9.1 where the types are $\theta_{1.1}, \theta_{2.1}$.

In addition to the form of the trajectory, the presence of FHSCEs dramatically affects the temporal length. Consider a uniform fully mixed strategy $(\mathbf{x}_1, \mathbf{x}_2)$ with the game depicted in Fig. 2.b. We compare, in term of number of iterations, the dynamics when there is a single individual per agent w.r.t. the dynamics when there are two individuals of agent 1 $(\lambda_{1.1}, \lambda_{1.2})$ whose aggregate strategy is the same of the case with a single individual. The results, reported in Tab. 9.4, show that when the two individuals have the same strategy, the dynamics length is the same of the case with a single individual. When instead the two individuals play different strategies, the dynamics get longer as the strategies get more different (the extreme is

$x_{\lambda_1} = \mathsf{L}_j$ and $x_{\lambda_2} = \mathsf{R}_j$ at each information set j: the length is double). Approximately, the length of the dynamics increases linearly in the number individuals per agent.

	1 individual per agent	2 individuals of agent 1				
		$x_\lambda(\mathsf{L})/x_\lambda(\mathsf{R})$ for individual 1, $x_\lambda(\mathsf{R})/x_\lambda(\mathsf{L})$ for individual 2				
		10^0	10^2	10^4	10^6	10^8
iterations	55	55	65	76	88	99

Table 9.4: Length of the learning dynamics.

Conclusions and future work 10

In Section 10.1 we conclude the work presented in this thesis and in Section 10.2 we discuss possible future developments.

10.1 Conclusions

The study of computational tools for general-sum extensive-form games is one of the most challenging issues in artificial intelligence. Such tools will allow the development of autonomous software agents able to play optimally.

In order to design these tools, we have to start from the scenario where they will be used. General-sum extensive-form games are a better way w.r.t. the strategic-form games to represent many real-world situations. Extensive-form games can represent scenarios that are deeply different between them and many designing problems arise. The most important is the choice of the suitable solution concept. This problem is related to the specific characteristic of the scenario, e.g. the agents' knowledge, the possibility of making mistakes, and so on. The information available to the agents defines the tools, potentially differents, to use to deal with these situations. When information is common, the appropriate solution concepts for extensive-form games refine the concept of Nash equilibrium, while,

when agents learn, the appropriate solution concepts relax the concept of Nash equilibrium introducing the concept of beliefs.

Given the suitable solution concept, a new problem arises: is this solution concept computable? To allow agents to work with a solution concept, its computation need to be easy, otherwise the agents cannot adopt it. The aim of this thesis is identifying problem that are easy to solve and design algorithm to solve it.

We focus on three main problems. The first was the verification problem, i.e., certifying that a given solution is an equilibrium according to some solution concept. The second one was the computational problem, i.e., compute a strategy profile that is an equilibrium according to some solution concept. The last problem was the learning problem of agent in situations where the information is not common, thus the agents cannot compute their optimal strategies.

10.2 Future work

The future developments of this thesis go in the direction of completing and extending the set of efficient algorithms for the verification, computation and learning problem in extensive-form games. A lot of work will be done in order to have a coherent set of tools to deal with extensive-form games in every possible scenario.

The verification problem of a number of solution concepts remains open. More specifically, it is open the problem of extensive-form perfect equilibrium and proper equilibria with two agents and the algorithms presented in Chapter 4 do not seem applicable with trivial adaptations. Other interesting problems concern the computation of QPE, EFPE, and NFPrE with more than two agents or when, different from Chapter 5, the input perturbation is not given. As future works, we aim at studying the scalability of the algorithm presented in Chapter 6 with large instances, evaluating our theoretical bounds with a variety of function classes, improving them, and extending the algorithm to solve general-sum games. In addition, we will remove the assumption of switching controller and we will explore both verification [21] and computation [20] of perfection-based solution concepts with continuous actions.

In future, we intend to explore the following problems: extending the results on multi-agent learning when sequence form is adopted taking into account also Nash refinements for extensive-form games

(we recall, while this is possible with sequence form, it is not with the normal form); extending the results of Chapter 7 to other forms of dynamics, e.g., best response dynamics, imitation dynamics, smoothed best replies, the Brown-von Neumann-Nash dynamics; comparing the expressivity and the effectiveness of replicator dynamics when applied to the three representation forms. We intend to extend the results presented in Chapter 8 as follows: defining a Q-learning based algorithm working with the agent form representation of extensive-form games, applying our algorithm to state-action model of extensive-form games, deriving theoretical bounds in probability (e.g., Hoeffding's and Chernoff's) over the distance between the trajectories predicted by the model and the actual trajectories. A further interesting future work is the adoption of tools presented in Chapter 9 to design game mechanisms. A designer could remove equilibria to change or remove learning dynamics, e.g. providing small utility, or to change the dynamical properties of some equilibrium, strengthening the attracting power to speed up the dynamics towards such equilibrium or weakening the attracting power to make the dynamics move farther from such equilibrium. Other future works consist in the extension of our tools with more than two agents where USCEs and NEs can be radically different.

Notation A

extensive-form representation	i	a generic agent
	a	a generic action
	w	a generic (terminal and non-terminal) node
	h	a generic information set
	N	set of agents
	A	set of agents' actions
	A_i	set of agent i's actions
	V	set of game tree decisional (non-terminal) nodes
	V_i	set of agent i's game tree decisional (non-terminal) nodes
	T	set of game tree terminal nodes
	H	set of information sets
	H_i	set of agent i's information sets
	$V_{i,h}$	set of nodes belonging to information set h of agent i
	ι	player function
	ρ	available action function
	χ	successor function
	u_i	agent i's utility function over terminal nodes
behavioural strategies	ϵ	symbolic perturbation
	$\boldsymbol{\sigma}_i$	agent i's (unperturbed) behavioral strategy
	$\sigma_{i,a}$	(unperturbed) probability of action a of agent i
	$\boldsymbol{\sigma}_i(\epsilon)$	agent i's behavioral strategy perturbed in ϵ
	$\sigma_{i,a}(\epsilon)$	probability of action a of agent i perturbed in ϵ
	$\boldsymbol{\sigma}$	(unperturbed) behavioral strategy profile
	$\boldsymbol{\sigma}(\epsilon)$	(perturbed) behavioral strategy profile
belief	$\boldsymbol{\mu}_i$	agent – beliefs over her nodes
	$\mu_{i,w}$	agent i's belief over node w
	$\boldsymbol{\mu}$	belief profile
sequence-form representation	q	a generic sequence
	$\mathsf{q}_\varnothing$	empty sequence
	$\lvert q\rvert$	length of sequence q
	Q	agents' set of sequences
	Q_i	agent i's set of sequences
	$\mathbf{x}_i$	agent i's (unperturbed) sequence-form strategy
	$x_{i,q}$	(unperturbed) probability of sequence q of agent i
	$\mathbf{x}_i(\epsilon)$	agent i's sequence-form strategy perturbed in ϵ
	$\mathbf{x}_i(\epsilon^k)$	vectors of coefficients of ϵ^k in $\mathbf{x}_i(\epsilon)$
	$x_{i,q}(\epsilon)$	probability of sequence q of agent i perturbed in ϵ
	$x_{i,q}(\epsilon^k)$	coefficient of ϵ^k in $x_{i,q}(\epsilon)$
others	$\mathbf{v}_i$	agent i's information set values
	$v_{i,h}$ (also v_h)	value of information set h of agent i
	$\mathbf{l}_i(\epsilon)$	agent i's perturbation in ϵ
	$l_{i,q}(\epsilon)$	agent i's perturbation over sequence q to find the QPE
	$\boldsymbol{\lambda}$	b-label vector
	λ_a	b-label of action a
	U_i	agent i's utility matrix
	F_i	agent i's sequence-form matrix

Table A.1: Notation used in the Part I of the thesis.

$\Theta = \Theta_1 \times \Theta_2$	the set of all the types
Θ_i	the set of types of agent i
Θ_{-i}	the set of all the possible profiles of types of all the agents except agent i
θ	a generic type
$u_i(\theta_i, \boldsymbol{\sigma}_i, \boldsymbol{\sigma}_{-i})$	the utility of agent i when types are non-interdependent
$u_i(\theta_i, \theta_{-i}, \boldsymbol{\sigma}_i, \boldsymbol{\sigma}_{-i})$	the utility of agent i when types are interdependent
Λ_i	the set of all the individuals of agent i
Λ_θ	the population of individuals (possibly infinite) for type $\theta \in \Theta_i$
λ	a generic individual
ω_θ	the probability with which an individual of type $\theta \in \Theta_i$ is drawn
ω_λ	the probability with which an individual $\lambda \in \Lambda_\theta$, $\theta \in \Theta_i$ is drawn
$\boldsymbol{\sigma}_\lambda$	the strategy of individual λ
$\boldsymbol{\sigma}_\theta = \sum_{\lambda \in \Lambda_\theta} \boldsymbol{\sigma}_\lambda \cdot \omega_\lambda$	the *aggregate strategy* of the individuals of type $\theta \in \Theta_i$
$\boldsymbol{\sigma}_i = \sum_{\theta \in \Theta_i} \boldsymbol{\sigma}_\theta \cdot \omega_\theta$	the *aggregate strategy* of the individuals of agent i
$\hat{\boldsymbol{\sigma}}_\lambda^{-i}$	the belief of an individual λ of agent i over the aggregate strategy of agent $-i$
$\hat{\boldsymbol{\sigma}}_\lambda^{\theta}$	the belief of an individual λ of agent i over the aggregate strategy of type $\theta \in \Theta_{-i}$
$\hat{\omega}_\lambda^{\theta}$	the belief of an individual λ of agent i over the pertinent probability ω_θ
$\boldsymbol{\sigma}_a$	the strategy parameterized w.r.t. action $a \in A_i$
$\hat{\boldsymbol{\sigma}}_a^{-i}$	the belief parameterized w.r.t. action $a \in A_i$ over over the aggregate strategy of agent $-i$
$\mathbf{x}_\lambda$	the strategy of individual λ
$\mathbf{x}_\theta$	the *aggregate strategy* of the individuals of type $\theta \in \Theta_i$
$\mathbf{x}_{\theta,q}$	the *aggregate strategy* of the individuals of type $\theta \in \Theta_i$ when plays $q \in Q_i$
$\hat{\mathbf{x}}_\lambda^{-i}$	the belief of an individual λ of agent i over the aggregate strategy of agent $-i$
$\hat{\mathbf{x}}_\lambda^{\theta'}$	the belief of an individual λ of agent i over the aggregate strategy of the individuals of type $\boldsymbol{\theta}' \in \Theta_{-i}$
$\hat{\mathbf{x}}_{\theta,q}^{-i}$	the belief of an individual of type $\theta \in \Theta_i$ when plays $q \in Q_i$ over the aggregate strategy of agent $-i$
$\hat{\mathbf{x}}_{\theta,q}^{\theta'}$	the belief of an individual of type $\theta \in \Theta_i$ when plays $q \in Q_i$ over the aggregate strategy of the individuals of type $\theta' \in \Theta_{-i}$
$\hat{\omega}_\theta$	the belief of agent i to play against an individual of type $\theta \in \Theta_{-i}$

Table A.2: Notation used in the Part I of the thesis.

Bibliography

[1] C. Archibald and Y. Shoham. Modeling billiards games. In *AAMAS*, pages 193-199, 2009.

[2] D.K. Arrowsmith and C.M. Place. *Dynamical Systems*. Chapman & Hall, London, 1992.

[3] D. Avis, G. D. Rosenberg, R. Savani, and B. von Stengel. Enumeration of Nash equilibria for two-player games. *ECON THEORY*, 42(1):9-37, 2010.

[4] S. Barnard. *Higher Algebra*. Barnard Press, 2008.

[5] N. Basilico, N. Gatti, and F. Villa. Asynchronous multi-robot patrolling against intrusions in arbitrary topologies. In *AAAI*, 2010.

[6] S. Belhaiza, C. Audet, and P. Hansen. On proper refinement of Nash equilibria for bimatrix games. *AUTOMATICA*, 2:297-303, 2012.

[7] M. Benisch, G. B. Davis, and T. Sandholm. Algorithms for closed under rational behavior (CURB) sets. *J ARTIF INTELL RES*, 38:513-534, 2010.

[8] L. Blum, A. Brandenburger, and E. Dekel. Lexicographic probabilities and equilibrium refinements. *ECONOMETRICA*, 59(1):81-98, 1991.

[9] G. Bonomi, N. Gatti, F. Panozzo, and M. Restelli. Computing equilibria with two-player zero-sum continuous stochastic games with switching control. In *AAAI*, Toronto, Canada, July 22-26 2012.

[10] M. Caliari, S. de Marchi, and M. Vianello. Padua2d: Lagrange interpolation at padua points on bivariate domains. *ACM TRANS MATH SOFTWARE*, 35(3):1-11, 2008.

[11] S. Ceppi, N. Gatti, and N. Basilico. Computing Bayes-Nash equilibria through support enumeration methods in Bayesian two-player strategic-form games. In *IAT*, pages 541-548, 2009.

[12] R. Chandrasekaran. Total unimodularity of matrices. *SIAM J APPL MATH*, 17(6):1032-1034, 1969.

[13] Xi Chen, Xiaotie Deng, and Shang-hua Teng. Computing Nash Equilibria: Approximation and Smoothed Complexity. *FOCS*, pages 603-612, 2006. doi: 10.1109/FOCS.2006.20. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4031395`.

[14] E.W. Cheney. *Introduction to approximation theory*. AMER MATH SOC, 1982.

[15] R. Cressman. *Evolutionary dynamics and extensive form games*. MIT Press, Cambridge, Mass., 2003.

[16] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The Complexity of Computing a Nash Equilibrium, 2006. URL `http://portal.acm.org/citation.cfm?doid=1132516.1132527`.

[17] Bruce Eckel, Chuck D. Allison, and Chuck Allison. *Thinking in C++, Vol. 2*. Pearson Education, 2 edition, 2003. ISBN 0130353132.

[18] D. Fudenberg and D.K. Levine. Self-confirming equilibrium. *ECONOMETRICA*, 61(3):523-545, 1993.

[19] D. Fudenberg and J. Tirole. *Game Theory*. 1991.

[20] N. Gatti and C. Iuliano. Computing an extensive-form perfect equilibrium in two-player games. In *AAAI*, pages 669-674, August 7-11 2011.

[21] N. Gatti and F. Panozzo. New results on the verification of nash refinements for extensive-form games. In *AAMAS*, pages 813-820, Valencia, Spain, June 4-8 2012.

[22] N. Gatti, F. Di Giunta, and S. Marino. Alternating-offers bargaining with one-sided uncertain deadlines: an efficient algorithm. *ARTIF INTELL*, 172(8-9):1119-1157, 2008.

[23] N. Gatti, F. Panozzo, and S. Ceppi. Computing a self-confirming equilibrium in two-player extensive-form games. pages 981-988, Taipei, Taiwan, May 2-6 2010.

[24] N. Gatti, F. Panozzo, and S. Ceppi. Mathematical programming formulations to compute steady states in two-player extensive-form games. In *Interactive Decision Theory and Game Theory Workshop* of *AAAI*, Atlanta, USA, July 11 2010.

[25] N. Gatti, F. Panozzo, and M. Restelli. Efficient evolutionary dynamics with extensive-form games. In *AAAI*, Bellevue, Washington, USA, July 14-18 2013.

[26] N. Gatti, F. Panozzo, and M. Restelli. Extensive-form games with heterogeneous populations: solution concepts, equilibria characterization, and learning dynamics. In *AAMAS*, Saint Paul, USA, May 6-10 2013.

[27] A. Ghouila-Houri. Caracterisation des matrices totalement unimodulaires. *C. R. Acad. Sci. Paris*, 254:1192-1194, 1962.

[28] A. Gilpin, T. Sandholm, and T. B. Sørensen. Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *AAAI*, pages 50-57, 2007.

[29] Andrew Gilpin. *Algorithms for abstracting and solving imperfect information games*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, April, 29 2009.

[30] S. Govindan and T. Klumpp. Perfect equilibrium and lexicographic beliefs. *INT J GAME THEORY*, 31(2):229-243, 2003.

[31] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[32] K. A. Hansen, P. B. Miltersen, and T. B. Sørensen. The computational complexity of trembling hand perfection and other equilibrium refinments. In *SAGT*, pages 198-209, 2010.

[33] S. Hart. Evolutionary dynamics and backward induction. *GAME ECON BEHAV*, 41(2):227-264, 2002.

[34] S. Hurkens. Learning by forgetful players. *GAME ECON BEHAV*, 11:304-329, 1995.

[35] P. R. Jordan, M. P. Wellman, and G. Balakrishnan. Strategy and mechanism lessons from the first ad auctions trading agent competition. In *ACM EC*, pages 287-296, 2010.

[36] S Karlin and L. S. Shapley. Geometry of Moment Spaces. *MEM AM MATH SOC*, 12:105, 1953.

[37] M.J. Todd K.C. Toh and R.H. Tutuncu. Sdpt3 - a matlab software package for semidefinite programming. *Optimization Methods and Software*, (11):545-581, 1999.

[38] E. Kohlberg and P. J. Reny. Independence on relative probability spaces and consistent assessments in game trees. *J ECON THEORY*, 75(2):280-313, 1997.

[39] D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *GAME ECON BEHAV*, 14(2):220-246, 1996.

[40] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *J ARITF INTELL RES*, 41:297-327, 2011.

[41] D. R. Kreps and R. Wilson. Sequential equilibria. *ECONOMETRICA*, 50(4):863-894, 1982.

[42] H. W. Kuhn. Extensive games. *Proc. Natl. Acad. Sci. U. S. A.*, 36:570-576, 1950.

[43] C. Lemke. Some pivot schemes for the linear complementarity problem. *Mathematical Programming Study*, 7:15-35, 1978.

[44] C. E. Lemke and J. J. T. Howson. Equilibrium points of bimatrix games. *SIAM J APPL MATH*, 12(2):413-423, 1964.

[45] J. Lofberg. YALMIP : a toolbox for modeling and optimization in MATLAB. *CACSD*, (4):284 - 289, 2004.

[46] J. F. Mertens. Two examples of strategic equilibrium. *GAME ECON BEHAV*, 8(2):378-388, 1995.

[47] P. B. Miltersen and T. B. Sorensen. Computing sequential equilibria for two-player games. In *SODA*, pages 107-116, 2006.

[48] P. B. Miltersen and T. B. Sørensen. Computing a quasi-perfect equilibrium of a two-player game. *ECON THEOR*, 42(1):175-192, 2010.

[49] John Nash. Non-Cooperative Games. *The Annals of Mathematics*, 54(No 2):286-295, 1951.

[50] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *J COMP SYS SCI*, 48 (3):498-532, 1994.

[51] Ya. L. Remez. On a method of tchebycheff type approximation of functions. *UKRAIN ANN*, 1935.

[52] T. Sandholm. Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions. *AI Magazine*, 28(3):45-58, 2007.

[53] T. Sandholm, A. Gilpin, and V. Conitzer. Mixed-integer programming methods for finding Nash equilibria. In *AAAI*, pages 495-501, 2005.

[54] Bruce Schmeiser and Luc Devroye. Non-Uniform Random Variate Generation. *J AM STAT ASSOC*, 83(403):906, September 1988. ISSN 01621459. doi: 10.2307/2289328. URL `http://www.jstor.org/stable/2289328?origin=crossref`.

[55] R. Selten. Spieltheoretische behandlung eines ein oligopolmodell mit nachfrageträgheit. *Zeitschrift für die Gesamte Staatswissenschaft*, 121:301-324 and 667-689, 1965.

[56] R. Selten. Reexamination of the perfectness concept for equilibrium points in extensive games. *INT J GAME THEORY*, 4 (1):25-55, 1975.

[57] Parikshit Shah and Pablo A. Parrilo. Polynomial Stochastic Games via Sum of Squares Optimization. *CDC*, pages 745-750, 2007. doi: 10.1109/CDC.2007.4434492. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4434492`.

[58] L S Shapley. Stochastic Games. *NAT ACAD SCIE USA*, 39(10):1095-1100, October 1953. ISSN 0027-8424. URL `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1063912&tool=pmcentrez&rendertype=abstract`.

[59] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press, 2008.

[60] J.A. Shohat and J.D. Tamarkin. The Problem of Moments. *American Mathematical Society Mathematical surveys*, 1, 1943.

[61] J. M. Smith and G. R. Price. The logic of animal conflict. *Nature*, 146(2):15-18, 1973.

[62] T. B. Sørensen. Computing a proper equilibrium of a bimatrix game. In *ACM EC*, Valencia, Spain, June 4-8 2012.

[63] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, USA, 1998.

[64] K. Tuyls, P. J. Hoen, and B. Vanschoenwinkel. An evolutionary dynamical analysis of multi-agent learning in iterated games. *AUTON AGENT MULTI-AG*, 12(1):115-153, 2006.

[65] E. van Damme. A relation between perfect equilibria in extensive form games and equilibria in normal form games. *INT J GAME THEORY*, 13(2):1-13, 1984.

[66] E. van Damme. *Stability and Perfection of Nash Equlibria*. Springer, 1991.

[67] D. Vermeulen and M. Jansen. The reduced form of a game. *EUR J OPER RES*, 106(1):204-211, 1998.

[68] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1944.

[69] B. von Stengel. Efficient computation of behavior strategies. *GAME ECON BEHAV*, 14(2):220-246, 1996.

[70] B. von Stengel, A. van den Elzen, and D. Talman. Computing normal form perfect equilibria for extensive two-person games. *ECONOMETRICA*, 70(2):693-715, 2002.

[71] O. J. Vrieze, S. H. Tijs, T. E. S. Raghavan, and J. A. Filar. A Finite Algorithm for the Switching Control Stochastic Game. *OR SPEKTRUM*, 5(1):15-24, 1983.

[72] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279-292, 1992.

[73] J. W. Weibull. *Evolutionary game theory*. MIT Press, Cambridge, Mass., 1995.

[74] M. Yannakakis. Equilibria, fixed points, and complexity classes. *STACS*, 2008.