The Learning Approach to Games

Melih İşeri* Erhan Bayraktar†

January 31, 2026

Abstract

This work introduces a unified framework for analyzing games in greater depth. In the existing literature, players' strategies are typically assigned scalar values, and equilibrium concepts are used to identify compatible choices. However, this approach neglects the internal structure of players, thereby failing to accurately model observed behaviors.

To address this limitation, we propose an abstract definition of a player, consistent with constructions in reinforcement learning. Instead of defining games as external settings, our framework defines them in terms of the players themselves. This offers a language that enables a deeper connection between games and learning. To illustrate the need for this generality, we study a simple two-player game and show that even in basic settings, a sophisticated player may adopt dynamic strategies that cannot be captured by simpler models or compatibility analysis.

For a general definition of a player, we discuss natural conditions on its components and define competition through their behavior. In the discrete setting, we consider players whose estimates largely follow the standard framework from the literature. We explore connections to correlated equilibrium and highlight that dynamic programming naturally applies to all estimates. In the mean-field setting, we exploit symmetry to construct explicit examples of equilibria. Finally, we conclude by examining relations to reinforcement learning.

ACM Classification: I.2.6; J.4

*Department of Mathematics, University of Michigan, United States, iseri@umich.edu.

†Department of Mathematics, University of Michigan, United States, erhan@umich.edu.

1 Introduction

Game theory, like every branch of mathematics, explores fundamental concepts for systems that are as general as possible, where individual components have potential choices to make. The opportunities for exploration are vast and deeply complex, with implications across a diverse array of fields. These include societal structures, competitions in numerous games, various dynamics of businesses, computational decision processes, financial models, and many more. Recent advancements have even surpassed human capabilities in various domains, prompting increased efforts to better understand our brains, the most fascinating dynamic system.

To motivate our goal intuitively, consider a swarm of self-driving cars operating in a city. From the traditional game-theoretic viewpoint, one might analyze the density of cars on the streets and emerging traffic patterns. For such a problem, it is neither feasible nor meaningful to account for the detailed internal structure of each vehicle. As a result, however, it cannot be used to specify or simulate how the cars themselves make decisions. We remark that such traditional methods remain rooted in a central-planner perspective. Roughly speaking, equilibrium concepts involve solving a control problem, augmented by a compatibility condition among agents' strategies. When the goal is to design optimal policies within well-defined environments, the traditional framework remains appropriate. However, when the focus shifts to designing complex players themselves, such as those enabled by modern reinforcement learning, the available methods are powerful, but a unifying language is lacking. This work aims to fill this gap.

The main approach of this work is to separate the concept of player from the external settings we design. This will provide a broader perspective on games, in which multiple complex decision-makers interact. In the literature, when there is a single decision-maker, designing external rewards in the environment guides which future observations the player will prefer. This, in turn, provides strong guidance for designing players that act in our interests. Still, designing players to tackle real-world problems requires significant effort beyond setting up the environment. We emphasize this perspective because once there is more than one player, the expressiveness of setting up the environment diminishes. We argue that omitting the structure of players and considering only stability conditions over the set of strategies cannot provide a sufficiently rich understanding.

Let us discuss two illustrative examples of competitive settings. The first concerns one of the simplest competitive games, rock-paper-scissors. Suppose a player competes against ten opponents in a tournament and can perfectly generate a uniform distribution over actions. Will the player choose the uniform equilibrium strategy? In that case, winning the tournament is highly unlikely. Of course, an equilibrium strategy is unexploitable, but players aiming to win the tournament

would naturally avoid it. Recognizing that no one would actually use the uniform strategy, the player's goal becomes twofold: to estimate the opponents' behavior and simultaneously to deceive their estimates. This shows two simple but crucial points: (i) for competing players, the objective is not to be unexploitable, and (ii) the essence of the game lies not in its formal rules, but in the players themselves. Even when the setting is simple, the strategic interplay among players can exhibit profound complexity, revealing that competition lies not in the game's structure but in its participants. The second example conveys the same ideas, but relies on the structure of the game rather than human sensory interactions. Suppose two players repeatedly change states in 0, 1, with their actions representing transition probabilities. One player aims to "catch" the other, while the second aims to "evade"; their rewards are defined in a zero-sum fashion. The same question arises: will they simply choose equal probabilities to remain unexploitable and earn nothing on average? In reality, if the players are truly competing, they will constantly change their strategies. For instance, the evading player might begin appearing in state 1 more frequently to lure the opponent into following, only to deceive them for a short-term gain. Note that no equilibrium strategy yields more than zero expected gain for either side, yet that is not what the players seek. The example motivates that competition is defined not by convergence, but by the very effort to avoid it. Hence, in the next section, we will say that players are cooperating when their behaviors actually converge. As these examples show, such cooperation may occur involuntarily, when one player seeks merely to remain unexploitable.

In Section 2, we introduce the main definition of a player, and discuss some intrinsic objectives of their components. In Section 3, we explore discrete games, treating mainly the standard components as estimates of the player. After presenting specific estimates, we define uncertain equilibrium to impose conditions such as optimality and recurrence. We then refine this notion by adding further conditions such as consistency and psychological states. This allows a richer characterization of both players and the equilibrium concept. We also demonstrate connections between our optimality assumption and the concept of correlated equilibrium. Later, we present a toy two-player game to illustrate the dynamic nature of games in the simplest settings. In Section 4, we examine mean-field games with constant estimates, except that the representative player estimates the population strategies. As observations can be generated by relying on symmetries, we introduce a learning algorithm with explicit uncertain equilibrium. In Section 5, we provide further connections to core structures in reinforcement learning, present a basic learning algorithm that does not rely on standard value-based methods, and review related concepts in multi-agent reinforcement learning.

Some related literature. Classic non-cooperative game theory traces back to the von Neumann-Morgenstern seminal work [1] and Nash [2]. We refer to the

book Maschler-Solan-Zamir [3] for a comprehensive treatment. There are many notions of equilibrium, and we highlight the correlated equilibrium introduced by Aumann [4]. The author raises common criticisms of the classic Nash equilibrium and introduces correlated equilibrium, which incorporates randomness in players' strategies. A correlated equilibrium is a single distribution over the players' strategies and is therefore typically motivated by a mediator who draws from this distribution. However, a single distribution cannot capture the differing knowledge each player may possess. Because of this structural connection, we will offer a comparison after introducing the uncertain equilibrium for discrete games.

Critiques of treating equilibrium concepts as central to games appear across several literatures. Kadane and Larkey [5], for example, adopt a subjective perspective, formalizing views that had previously been discussed informally (as reflected in the Editor's Note) and bringing them into the management science framework. In the multi-agent reinforcement learning literature, the technical note by Shoham, Powers and Grenager [6] (see also [7]) criticizes the lack of conceptual clarity arising from the unjustified use of equilibrium concepts as both learning objectives and evaluation criteria. In the behavioral game theory literature, Wright and Leyton-Brown [8] demonstrate that Nash equilibrium is a poor description of human behavior even in simple games, and provide a systematic comparison of different models of human behavior. We remark that all such models still omit a model of the human itself, mapping an external game description directly to a distribution over actions. Hartford, Wright, and Leyton-Brown [9] demonstrate that a neural network, essentially a function approximation, outperforms all such behavioral models. This strengthens the motivation for this work, as we argue that the primary modeling object should be the players themselves, and that games cannot be reduced to their external descriptions.

Learning in games has a rich literature, beginning with Brown [10], who introduced the notion of fictitious play. Players are assumed to have predefined learning rules, and the question is whether the long-run average of observed actions converges to an equilibrium. Although such convergence is not always guaranteed (see, for example, Daskalakis et al. [11]), Hart and Mas-Colell [12] combine regret with fictitious play to show convergence to correlated equilibrium. For a comprehensive treatment, see the book Fudenberg-Levine [13].

To address games with a large number of players, where equilibria become intractable, Lasry-Lions [14] and, independently, Huang-Malhamé-Caines [15] introduced the concept of mean-field games. Since then, the framework has been extensively studied. In this setting, agents interact only through the empirical distribution of their states and are indistinguishable from one another, allowing a continuum limit to be identified with a representative agent. We refer the reader to the excellent two-volume book Carmona-Delarue [16, 17].

In our framework, players may favor having a large collection of estimates, each of which must be learned from observations. We cannot hope to cover every relevant learning algorithm and its extensive literature, however, we will highlight some connections in Section 5. Here, we would like to point out the diverse work on random value functions. We also advocate that the value is fundamentally unknown to a learning player and is one of the main sources of uncertainty in planning future behavior. We refer to Thurstone [18], Luce [19], Block [20], McFadden [21], Train [22], and references therein for discussions rooted in psychology and economics. One of the oldest approaches is Thompson sampling, introduced in Thompson [23] and recently popularized by the empirical study Chapelle-Li [24]. Upper confidence bound algorithms in the context of multi-armed bandit problems, see for example Auer et al. [25], can also be viewed as a random value approach. Lastly, let us mention Bellemare-Dabney-Munos [26] who prominently promoted modeling the value function as a random variable in the context of Markov decision processes with applications to reinforcement learning.

2 Definition of Players

In this section, we provide a definition of a player and elaborate on general requirements we may impose. At the end, we also define cooperation and competition through their behavior. Let us first introduce some preliminary definitions and notation:

- Universe, or the environment, is an abstract probability space $(\Omega^u, \mathcal{F}^u, \mathbb{P}^u)$;
- $\mathcal{P}(E)$ denotes the set of probability distributions on an arbitrary set E;
- $\mathcal{E}$ is the space of observables, and $\mathfrak{E}$ is the set of finite sequences of $\mathcal{E}$;
- $\mathbb{A}$ is the space of actions, and $\mathfrak{A}$ is the set of finite sequences of $\mathbb{A}$;
- $\mathcal{M}_\Upsilon$ is the set of functions taking values in $\mathbb{A}$, called the space of behaviors;
- $\mathcal{M}_\varphi$ is the set of functions, called the space of estimates.

Similar to actions, we are keeping observables abstract. We have not yet specified the domains and ranges of the estimations. Also, φ denotes an index for estimates. We will introduce a collection of them in the upcoming sections.

In the realm of games, a player is defined by a sequence of observations, a collection of estimates, and a sequence of actions, all of which may be highly complex. We now introduce a definition of a player:

Definition 1 *We call* $(\mathcal{O}, \mathfrak{L}_\varphi, \Upsilon)$ *a player in the environment* $(\Omega^u, \mathcal{F}^u, \mathbb{P}^u)$ *with*

observations $\mathcal{O}$, learning algorithm $\mathfrak{L}_\varphi$, and with behavior Υ, where

$$\begin{aligned} \mathcal{O} &: \Omega^u \times \mathcal{A} \times \mathbb{N} \to \mathcal{E}, \\ \mathfrak{L}_\varphi &: \mathcal{E} \times \mathcal{M}_\varphi \times \mathcal{M}_\Upsilon \to \mathcal{M}_\varphi, \\ \Upsilon &: \mathcal{E} \times \mathcal{M}_\varphi \times \mathcal{M}_\Upsilon \to \mathcal{M}_\Upsilon. \end{aligned} \tag{2.1}$$

From the perspective of mathematics, the question is what natural conditions can be imposed on this collection of functions defining a player. We begin with the consistency condition for the observations:

Definition 2 *We say a player has consistent observations, if*

$$\begin{aligned} &\mathcal{O}(\omega^u, a_\cdot, n) \in \mathcal{E} \ \text{ is a subsequence of } \ \mathcal{O}(\omega^u, \tilde{a}_\cdot, n+1) \in \mathcal{E} \\ &\text{if } \ a_\cdot \in \mathcal{A} \ \text{ is a subsequence of } \ \tilde{a}_\cdot \in \mathcal{A}, \ \text{ for all } \ n \in \mathbb{N}, \omega^u \in \Omega^u \end{aligned}$$

We note that $\mathcal{O}$ sets the connection between the environment and the player, and is not available for the player to evaluate.

Next, we define a recurrence condition for a player's behavior. To do so, we first introduce the following definition:

Definition 3 *We call*

$$^n\Upsilon : \Omega^u \times \mathbb{N} \to \mathcal{M}_\Upsilon$$

the planned behavior of the player at age n, where

$$\begin{aligned} {}^n\Upsilon &:= \Upsilon({}^n\mathcal{O}, {}^n\mathfrak{L}_\varphi, {}^{n-1}\Upsilon) \in \mathcal{M}_\Upsilon, \\ {}^n\mathfrak{L}_\varphi &:= \mathfrak{L}_\varphi({}^n\mathcal{O}, {}^{n-1}\mathfrak{L}_\varphi, {}^{n-1}\Upsilon) \in \mathcal{M}_\varphi, \\ {}^n\mathcal{O} &:= \mathcal{O}(\omega^u, {}^{n-1}I, n) \in \mathcal{E}, \\ {}^nI &:= ({}^1\Upsilon(\omega^u, \cdot), \ldots, {}^n\Upsilon(\omega^u, \cdot)) \in \mathcal{A}. \end{aligned}$$

These functions ${}^n(\Upsilon, \mathfrak{L}_\varphi, \mathcal{O}, I) := ({}^n\Upsilon, {}^n\mathfrak{L}_\varphi, {}^n\mathcal{O}, {}^nI)$ with domain $\Omega^u \times \mathbb{N}$ are determined in the order ${}^{n-1}I \to {}^n\mathcal{O} \to {}^n\mathfrak{L}_\varphi \to {}^n\Upsilon$. Behaviors in $\mathcal{M}_\Upsilon$ depend on Ω^u, and evaluating at ω^u yields a sampled (or observed) action. That is, for ${}^k\Upsilon \in \mathcal{M}_\Upsilon$, we denote ${}^k\Upsilon(\omega^u, \cdot)$ as the action taken, suppressing the rest of the unspecified domain. Let us point out that a player might have the capacity to generate randomness independently of the surrounding environment. For now, we do not explicitly track potentially independent probability spaces, such as those a player might use to sample randomness, but instead include them within the general environment. Along these lines, only the component of $\omega^u \in \Omega^u$ that is relevant to the random variable under consideration is taken into account.

We are now ready to introduce an intrinsic concept for the behavior of player. Let us equip the space $\mathcal{M}_\Upsilon$ with a generic metric d and define:

Definition 4 *We say* ${}^*\Upsilon \in \mathcal{M}_\Upsilon$ *is a* (r, δ)*-recurrent behavior for a player, if*

$$\mathbb{P}^u\Big(\liminf_{n\to\infty} d({}^*\Upsilon, {}^n\Upsilon) > r\Big) \leq \delta.$$

Also, we say ${}^*\Upsilon$ *is almost surely a recurrent behavior of the player if* $r = \delta = 0$.

In words, we classify behaviors that may occur infinitely often as the player ages.

Finally, let us turn to the more intricate task of imposing conditions on estimates. Motivated by the brain's predictive nature, we introduce a notion of a player that estimates future abstract representations of observations. This lies at the core of behavior, and, roughly speaking, decision-making is about forming preferences over future observations. If we formalize observations as states, rewards, and actions, then it becomes crucial to understand the future states, rewards, and actions of other players. Then, we designate preferred future observations as those with higher total rewards. Given that observations are high-dimensional and complex, a player may need to simplify the task. For example, with visual observations, instead of predicting future pixels directly, one typically first forms useful embeddings to facilitate prediction. In case of actions, a player might estimate only intensions or goals of an oponnent. A similar reduction applies to rewards, rather than predicting future rewards directly, one may aim to learn the expected future reward. A more sophisticated agent might aim to learn a distribution of rewards.

To add structure, we introduce objects and relations formed from observations as the first layer of estimates. Let $\mathbb{N}_{\text{obj}} \subset \mathbb{N}$ be an index set for different objects. For each $j \in \mathbb{N}_{\text{obj}}$, let E^j_{obj} denote the space of states for object j. Finally, let E_{rel} be a space representing the set of relations. Then, let

$$\mathcal{M}_{\text{obj}} := \big\{\mathcal{E} \to \Pi_j E^j_{\text{obj}}\big\}, \;\; \mathcal{M}_{\text{con}} := \big\{\Pi_j E^j_{\text{obj}} \to 2^{\mathbb{N}_{\text{obj}}}\big\},$$

$$\text{and} \quad \mathcal{M}_{\text{rel}} := \big\{\Pi_j E^j_{\text{obj}} \times 2^{\mathbb{N}_{\text{obj}}} \to E_{\text{rel}}\big\}.$$

Correspondingly, we have the learning algorithms

$$\mathfrak{L}_{\{\text{obj,con,rel}\}} : \mathcal{E} \times \mathcal{M}_\varphi \times \mathcal{M}_\Upsilon \to \mathcal{M}_{\{\text{obj,con,rel}\}}$$

Then, ${}^n\mathfrak{L}_{\text{obj}} \in \mathcal{M}_{\text{obj}}$ is a mapping from observations to states of identified objects. The mapping ${}^n\mathfrak{L}_{\text{con}} \in \mathcal{M}_{\text{con}}$ creates connections between those objects. Finally, ${}^n\mathfrak{L}_{\text{rel}} \in \mathcal{M}_{\text{rel}}$ assigns relations to connections, which may take the form of discrete tags or numerical values. All of these learning algorithms may depend on past observations, estimates (denoted generically by φ), and the behavior mapping.

Now, the one-step prediction problem can be formulated as the task of modeling the next objects and relations. To represent this, let us introduce

$$\mathcal{M}_{\text{pre}} := \big\{\Omega^u \times \Pi_j E^j_{\text{obj}} \times 2^{\mathbb{N}_{\text{obj}}} \times E_{\text{rel}} \times \mathbb{A} \to \Pi_j E^j_{\text{obj}} \times 2^{\mathbb{N}_{\text{obj}}} \times E_{\text{rel}}\big\}$$

along with the learning algorithm $\mathfrak{L}_{\text{pre}}$ and its realization ${}^{n}\mathfrak{L}_{\text{pre}} \in \mathcal{M}_{\text{pre}}$ at age n.

Definition 5 *We say* $(\varphi_{\text{obj}}, \varphi_{\text{con}}, \varphi_{\text{rel}}, \varphi_{\text{pre}}) \in (\mathcal{M}_{\text{obj}}, \mathcal{M}_{\text{con}}, \mathcal{M}_{\text{rel}}, \mathcal{M}_{\text{pre}})$ *is one-step* ε*-predictive under some metric* d *on* $\mathcal{P}(E_{\text{obj}} \times 2^{\mathbb{N}_{\text{obj}}} \times E_{\text{rel}})$*, if*

$$\liminf_{n\to\infty} d\left(\text{Law}({}^{n+1}E|{}^{n}\mathcal{O}, {}^{n}I),\ \text{Law}(\varphi_{\text{pre}}(\omega^{u}, {}^{n}E, {}^{n}\Upsilon(\omega^{u}, \cdot))|{}^{n}\mathcal{O}, {}^{n}I) \right) \leq \varepsilon,$$

where ${}^{n}E := \left(\varphi_{\text{obj}}({}^{n}\mathcal{O}),\ \varphi_{\text{con}}(\varphi_{\text{obj}}({}^{n}\mathcal{O})),\ \varphi_{\text{rel}}(\varphi_{\text{obj}}({}^{n}\mathcal{O}), \varphi_{\text{con}}(\varphi_{\text{obj}}({}^{n}\mathcal{O})))\right)$

$\mathbb{P}^{u}$*-almost surely.*

We note that φ_{pre} is typically a random variable whose law models the distribution of ${}^{n+1}E$. Let us also point out a potentially confusing notation. Given $({}^{n}\mathcal{O}, {}^{n}I)$, the realized action ${}^{n}\Upsilon(\omega^{u}, \cdot)$ is determined, whereas $\varphi_{\text{pre}}(\omega^{u}, \cdot)$ typically includes an independent component to model a distribution.

Let us discuss some examples to motivate the definitions above. Some related references will be provided in Section 5.

• We can simplify by considering the overall state of the environment as an observation, treated as a single object without further decomposition and connections. And, φ_{rel} may assign values to those states. In this case, φ_{pre} models the one-step transition probabilities of the whole state, viewed as a random variable, and the next value associated with it.

• Encoding the external environment is typically necessary. For humans, for example, the eyes encode visual observations from the environment. In the context of machine learning, a suitable neural network architecture carries out the encoding. Various objectives may be assigned to encoders, and φ_{pre} represents the predictive one, which might be the next embedding, a reward component of the observations, or another useful aspect of the observations for a given task.

• Instead of encoding the environment as a single object, one may construct many objects, each with its associated states. Then, for example, φ_{con} may form pairs, and φ_{rel} may assign attention values.

• Similarly, in the context of large language models (LLMs), objects can correspond to token embeddings. Here, φ_{con} forms pairs, connecting each future token to all previous ones (forming a triangular matrix), and φ_{rel} uses keys and queries to assign weights to these connections. This constitutes only a small component of such models.

• In the context of chess, objects can be defined as pieces together with their positions on the board. A player may then assign a large number of connections between such objects through a highly dynamic φ_{con}, for example between pieces that protect each other or among larger groups representing the overall arrangement. Then, φ_{rel} may assign values to each connection, and φ_{pre} may model the behavior of the opponent.

Let us emphasize that the aim of this work is not to design new learning algorithms for estimating future observations. Our objective is to reframe common game settings from the perspective of the player and to introduce a concept of equilibrium via conditions on $(\mathcal{O}, \mathcal{L}_\varphi, \Upsilon)$. Furthermore, we emphasize that designing a player can be far more involved than setting up the game itself. We argue that understanding complex interactions between players cannot be reduced to the external setting of the game under consideration.

When there are multiple players, we keep track of them using the index $i \in \mathbb{N}_0 := \{1, 2, \dots\}$. For any symbol ψ introduced in the definitions above, we set $\vec{\psi} := \prod_{i\in\mathbb{N}_0} \psi^i$. In this case, $\mathcal{E}^i$, the space of observables for player i, usually includes the actions or states of the other players. They interact and influence one another, and their collective recurrent behaviors ${}^*\vec{\Upsilon}$ form a basis for an equilibrium.

Lastly, we introduce notions of cooperation and competition defined by the players' behavior rather than by the structure of the game itself. The goal is to separate these concepts from the external setting and use them to clarify our objectives when designing players. In essence, we ask whether the players we model are meant to exhibit stable, convergent behaviors, or to continually generate diversity through dynamic interaction.

Definition 6 *We say players* $(\vec{\mathcal{O}}, \vec{\mathfrak{L}}_\varphi, \vec{\Upsilon})$ *in the environment* $(\Omega^u, \mathcal{F}^u, \mathbb{P}^u)$ *are eventually cooperating at* ${}^*\vec{\Upsilon} \in \vec{\mathcal{M}}_\Upsilon$*, if*

$$\mathbb{P}^u\Big(\limsup_{n\to\infty} \sup_{i\in\mathbb{N}_0} d({}^*\Upsilon^i, {}^n\Upsilon^i) > 0\Big) = 0.$$

We say that players are indefinitely competing, if for any $\vec{\Upsilon} \in \vec{\mathcal{M}}_\Upsilon$*,*

$$\mathbb{P}^u\Big(\limsup_{n\to\infty} \sup_{i\in\mathbb{N}_0} d(\Upsilon^i, {}^n\Upsilon^i) > 0\Big) = 1.$$

As an illustration, consider a simple setting of algorithmic price competition between two firms. Although the environment is designed to be competitive, if each firm's pricing algorithm learns that the competitor's prices tend to move in a positively correlated manner, both algorithms can quickly converge to stable, elevated price levels. In this case, we regard the firms as cooperating, even though the underlying game remains competitive.

3 Discrete Games

Let $\mathbb{T} = \mathbb{N}$ denote the time indices, and $\mathbb{S}_t$ be a measurable state space for each $t \in \mathbb{T}$. Set $\mathbb{S} := \bigcup_{t\in\mathbb{T}} \mathbb{S}_t$. For arbitrary set E with Borel σ-algebra $\mathcal{B}(E)$, let

$\mathcal{P}(E)$ denote the set of all probability measures on E. We will always consider discrete indexing to avoid discussions on regularities and measurability.

Take $\Omega := \prod_{t\in\mathbb{T}} \mathbb{S}_t$ as the canonical space. Define $X : \mathbb{T} \times \Omega \to \mathbb{S}$ as the canonical process: $X_t : \Omega \to \mathbb{S}_t$, and $X_t(\omega) = \omega_t$ for each $\omega \in \mathbb{X}$ and $t \in \mathbb{T}$. Let $\mathbb{F}^X$ denote the filtration generated by X. We always require any function defined on $\mathbb{T} \times \Omega$ to be Markovian, similar to the canonical process, and denote their parameters as (t, x) where it is understood that $x \in \mathbb{S}_t$.

We use $i \in \mathbb{N}_0 := \mathbb{N} \setminus \{0\}$ as the index for players. For any $i \in \mathbb{N}_0$, let $\mathbb{A}^{t,x;i}$ be the action space of player i at $(t, x) \in \mathbb{T} \times \mathbb{S}$. Introduce

$$\vec{\mathbb{A}}^{t,x} := \prod_{i\in\mathbb{N}_0} \mathbb{A}^{t,x;i}, \qquad \vec{\mathbb{A}} := \bigcup_{(t,x)\in\mathbb{T}\times\mathbb{S}} \vec{\mathbb{A}}^{t,x}, \qquad \mathbb{A}^i := \bigcup_{(t,x)\in\mathbb{T}\times\mathbb{S}} \mathbb{A}^{t,x;i}.$$

Let us also introduce the space of controls;

$$\mathcal{A}^i := \left\{\alpha : \mathbb{T} \times \Omega \to \mathbb{A}^i \ : \ \alpha(t, x) \in \mathbb{A}^{t,x;i} \ \ \forall (t, x) \in \mathbb{T} \times \mathbb{S}_t\right\}, \quad \forall i \in \mathbb{N}_0$$

and set $\vec{\mathcal{A}} := \prod_{i\in\mathbb{N}_0} \mathcal{A}^i$.

For the connection between players and the environment, let $\mathcal{E}^i$ denote the space of observables for player i, and set $\mathcal{E}^i$ as the finite sequences of $\mathcal{E}^i$. Similar to previous notations, set $\vec{\mathcal{E}} := \prod_{i\in\mathbb{N}_0} \mathcal{E}^i$ and $\vec{\mathcal{E}} := \prod_{i\in\mathbb{N}_0} \mathcal{E}^i$. We will state every estimate as depending on (t, x), and hence we will assume $\mathbb{T} \times \Omega \subset \mathcal{E}^i$. Also, set $\mathcal{A}^i$ as finite sequences in $\mathbb{A}^i$, and $\vec{\mathcal{A}} := \prod_{i\in\mathbb{N}_0} \mathcal{A}^i$.

As for the learning parameters, we will now begin to introduce horizon, transitions between states, transition costs, state values, potential behaviors of other players, optimal controls and expectations of the players. Our choices are inherently limited as a player might be arbitrarily complicated. Our aim here is to restate the general setting for many of our games in the perspective of player , and demonstrate the concept of uncertain equilibrium. For simplicity, we will temporarily disregard the index i and focus solely on the perspective of a single player.

First, we let players have a *horizon* $\hat{T}$. Players cannot predict the future indefinitely with reasonable accuracy. In other words, as the horizon of prediction increases, the distribution of the state process contains progressively less useful information, eventually rendering it useless. Thus, let

$$\mathcal{M}_T := \left\{\hat{T} : \mathbb{T} \times \Omega \to \mathbb{T}\right\} \tag{3.1}$$

be the space of all such functions, where the corresponding learning algorithm will take values in. Notice that we allowed the horizon to depend on the state, since the player might be able to project further in well-trained states. More importantly,

rather than a fixed time, one can consider a stopping time $\hat{T}^{(t,x)}(s,y) : (\mathbb{T}\times\Omega)^2 \to \mathbb{T}$, where $\hat{T}^{(t,x)}$ is a stopping time for the future estimated process in (3.3).

Next, the player have an estimate of the transition probabilities;

$$\begin{aligned} &\hat{p} : \mathbb{T} \times \Omega \times \vec{\mathbb{A}} \times \mathbb{S} \to \mathbb{R}^+, \ \text{ where} \\ &\hat{p}(t,x,\vec{a};\cdot) \ \text{ is a probability measure on } \ \mathbb{S}_{t+1}, \\ &\text{for all } t \in \mathbb{T}, x \in \mathbb{S}_t, \ \text{and } \vec{a} \in \vec{\mathbb{A}}^{t,x}. \end{aligned} \tag{3.2}$$

Similarly, introduce $\mathcal{M}_p$ as the space of all such mappings in (3.2).

Given $\hat{p}$ as in (3.2), an initial $(t,x) \in (\mathbb{T}, \mathbb{S}_t)$, and $\vec{\alpha} \in \vec{\mathcal{A}}$, player induces a distribution $\mathbb{P}^{t,x,\vec{\alpha}} := \mathbb{P}^{\hat{p};t,x,\vec{\alpha}}$ for the canonical process as usual; for all $t \le s$ and $(\tilde{x}, y) \in (\mathbb{S}_s, \mathbb{S}_{s+1})$, initial condition is $\mathbb{P}^{t,x,\vec{\alpha}}(X_t = x) = 1$ and

$$\mathbb{P}^{t,x,\vec{\alpha}}(X_{s+1} = y | X_s = \tilde{x}) = \hat{p}(s,\tilde{x},\vec{\alpha}(s,\tilde{x});y). \tag{3.3}$$

Note that relaxed controls further integrate over the distribution of controls to define (3.3). We instead integrate the value below.

It is crucial that players learn about other players' behavior. To fully understand any complex game, we cannot overlook this fact. Knowledge of opponents' strategies intrinsically alters the observed events within the game. Even a player's value depends on it, as different opponents might tend to employ varying strategies. Consequently, the value associated with a strategy cannot disregard the opponents' reactions. Thus, we assume that a player learns potential controls of others based on their own control;

$$\hat{\Gamma}^i : \mathbb{T} \times \mathcal{A}^i \to \mathcal{P}(\vec{\mathcal{A}}) \quad \text{and set } \ \hat{\Gamma}^i_{t,\alpha}(d\vec{\alpha}) := \hat{\Gamma}^i_t(\alpha; d\vec{\alpha}) := \hat{\Gamma}^i(t,\alpha)(d\vec{\alpha}) \tag{3.4}$$

Denote $\mathcal{M}_\Gamma$ as the space of all such mappings. We remark two points for (3.4):
(i) We assume that players model the others potential controls depending on their own control. However, one might model that this depends on the path of states of the players, or any other observables are legitimate as long as the cost (3.6) is well-defined.
(ii) For competing players, a sophisticated player might have an estimate on how their actions could be exploitable in order to deceive an opponent, deviating their $\hat{\Gamma}$ from their actual planning. To not only compete with but also cooperate with other players, they may need to generate reliable estimates of the actions of others. In the two-player game discussed in Section 3.1, because the costs to the players depend on each other's states, omitting this aspect from the player model won't accurately capture the observed dynamics.

An important notion to introduce is the value of a player. As future observations are ranked by some associated rewards, value function captures a qualitative

information about the future rewards for a given strategy. Now, we introduce transition costs and state values:[1]

$$\hat{F} : \hat{\Omega} \times \mathbb{T} \times \Omega \times \vec{\mathbb{A}} \to \mathbb{R}, \qquad \hat{\phi} : \hat{\Omega} \times \mathbb{T} \times \Omega \to \mathbb{R}, \tag{3.5}$$

and let $\mathcal{M}_F, \mathcal{M}_\phi$ denote the sets of mappings as in (3.5). An important difference is that the player models these as random variables on some probability space $(\hat{\Omega}, \hat{\mathcal{F}}, \hat{\mathbb{P}})$, which we are now explicitly separating from the environment and view it as an independent component of it. In particular cases, it might be useful to characterize the measure space $(\hat{\Omega}, \hat{\mathcal{F}}, \hat{\mathbb{P}})$, however, once can also fix a sufficiently large probability space and concentrate on the random variables. We remark that state value $\hat{\phi}$ induces an ordering on states, and reaching a particular state by different intermediate paths, or different set of actions might have varying costs, which is aimed to be captured by the transition cost $\hat{F}$.

Now, given $(\hat{T}, \hat{p}, \hat{\Gamma}, \hat{F}, \hat{\phi}) \in \mathcal{M}_T \times \mathcal{M}_p \times \mathcal{M}_\Gamma \times \mathcal{M}_F \times \mathcal{M}_\phi$, the value of player becomes

$$\begin{aligned} J(t,x;\alpha) &:= \int_{\vec{\mathcal{A}}} J(t,x;\vec{\alpha})\hat{\Gamma}_t(\alpha; d\vec{\alpha}), \quad \text{where denoting } \mathbb{E}^{t,x,\vec{\alpha}} := \mathbb{E}^{\mathbb{P}^{t,x,\vec{\alpha}}}, \\ J(t,x;\vec{\alpha}) &:= \mathbb{E}^{t,x,\vec{\alpha}}\Big[\hat{\phi}(t+\hat{T}, X_{t+\hat{T}}) + \sum_{s=t}^{t+\hat{T}-1} \hat{F}(s, X_s, \vec{\alpha}(s, X_s))\Big], \end{aligned} \tag{3.6}$$

which is a random variable on $\hat{\Omega}$. Set $\mathcal{M}_J^i$ as the space of all such functions $(\hat{\Omega} \times \mathbb{T} \times \Omega \times \mathcal{A}^i \to \mathbb{R})$. We point out that requiring a random variable instead of a distribution allows us to refer to samples.

We remark again that a general abstract setting might be a simplification, and there might be many layers of various estimations, such as objects and their relations, before a player actually constructs its value estimate. Indeed, it may be the case for every other estimate too. Transition probabilities might be estimated from simple frequency analysis, or could be modeled by large attention architectures. In fact, a truly complex player, such as a human, would not only adapt existing estimates and behaviors, but also develop new types of estimates and behaviors over time. Just as a child starts with limited capacities and gradually acquires a rich repertoire, such dynamic structural growth might indeed be essential for modeling higher-order intelligence. While this remains far beyond current work, it represents a compelling direction for future research.

[1] We use cost and value interchangeably. In the case of scalar objectives as in this work, distinction is more pronounced. However, for multi-objective frameworks, there is typically no binary choice, but rather a continuum of choices.

Let us recall the game of chess, which serves as an excellent example to keep in mind throughout this work. In chess, $\hat{p}$ yields deterministic transitions. However, a player does not know what actions the opponent will take within $\{t, \dots, t + \hat{T}\}$, and beyond that, it is unclear what the transition costs of actions or the value of being in a particular state at $t + \hat{T}$ might be. These are all crucial components for a player to learn. Notably, the heuristic values assigned to pieces are designed to guide players in learning $\hat{F}$ and $\hat{\phi}$. While simplistic, these heuristics serve as an initial guide. Moreover, as we have mentioned, knowledge about the opponent can alter the values of strategies, which is captured in (3.6) abstractly. Let us also emphasize that the player's horizon may depend significantly on the current state. Towards the endgame, for instance, a well-trained chess player might be able to estimate many steps ahead, whereas this ability may be considerably more limited during the middle stages of the game.

As the player faces the optimization problem (3.6), it is not always feasible to solve for the optimal control. When $\hat{T} = 1$, the problem might be relatively simple, allowing for straightforward searches for ϵ-optimal actions. However, for longer horizons, the space of potential controls becomes excessively large, complicating the search for optimal solutions. To formalize this, let us first define

$$\alpha =^{t,x;i} \tilde{\alpha} \quad \text{if} \quad \alpha(s, y) = \tilde{\alpha}(s, y) \quad \forall s \in \{t, \dots, t + \hat{T}^i(t, x) - 1\},\ y \in \mathbb{S}_s$$

Under this equivalency relation, we introduce the quotient space

$$\mathcal{A}^{t,x;i} := \mathcal{A}^i / =^{t,x;i}$$

And then, to incorporate the potential difficulty and uncertainty in identifying the optimal control, we introduce the next policy estimation;

$$\begin{aligned} &\hat{\pi} : \hat{\Omega} \times \mathbb{T} \times \Omega \to \mathcal{A}^i \qquad \text{where,} \\ &\hat{\pi}(\hat{\omega}, t, x) \in \mathcal{A}^{t,x;i}, \quad \forall (\hat{\omega}, t, x) \in \hat{\Omega} \times \mathbb{T} \times \mathbb{S}_t \end{aligned} \tag{3.7}$$

Here, at $(\hat{\omega}, t, x)$, $\hat{\pi}$ approximates the potential optimal controls for $J(\hat{\omega}, t, x, \cdot)$, which will be dictated by the equilibrium condition below. We remark that $\hat{\Omega}$ might have a component for both value and policy, which we don't keep track explicitly. For example, we might have a collection of events where the value is determined, whereas $\hat{\pi}$ is still a random variable. Set $\mathcal{M}_\pi$ as the set of functions as in (3.7).

Now, even when optimal control can be solved exactly, uncertainty over the value will naturally induce a probability distribution over controls. First, introduce the space of behavior $\mathcal{M}_\Upsilon$ as functions of the form

$$\hat{\Upsilon} : \hat{\Omega} \times \mathbb{T} \times \Omega \to \mathbb{A}^i \tag{3.8}$$

Then, we introduce a behavior in a straightforward manner from the current estimates as follow;

$$\begin{aligned} &\Upsilon : \Pi_\varphi \mathcal{M}_\varphi \to \mathcal{M}_\Upsilon \quad \text{where,} \\ &\Upsilon(\hat{\pi}) = ((\hat{\omega}, t, x) \mapsto \hat{\pi}(\hat{\omega}, t, x)(t, x)) \end{aligned} \tag{3.9}$$

Here, behavior is choosing the immediate action assigned by the policy, and reflects the randomness induced by both the value and policy. This behavior is typical for the learning or playing phase and is, in essence, similar to Thompson sampling adapted to our framework. During a competition phase, one might choose a different Υ as deterministic, selecting the action corresponding to the mode of them. In particular situations, such as performing surgery, it is not only wrong but also unethical to forgo the most likely action and instead select one at random. Moreover, a player may adjust its behavior to occasionally select unlikely actions, exploring states that are disadvantageous or even entirely unseen when facing a weak opponent.

It is important to motivate the role of randomness in value, which then induces a probability distribution over actions by (3.9). Recall that in sufficiently complex settings, such as chess, values are inherently unknown and must be learned through significant effort. That is, the randomness of the value models what is unknown to the player. One key role of randomness in value is to allow players to explore systematically. If the player is not satisfied with the current value estimates, it is natural to shift the estimates for unexplored controls, or their outcomes, toward higher values, in anticipation that they may achieve better results than current estimates. This approach naturally leads the player to search for controls yielding more satisfactory outcomes. We will demonstrate a toy version for the two-player game in Section 3.1. This approach aligns with the common intuition that a better understanding of values should lead to less uncertain strategies.

Now that we have introduced the spaces of estimations and behavior, let us turn to players as in Definition 1. Observations may come from real-world experience, or, in the mean-field regime, players can generate observations by assuming that every other player is identical. For multiple players, we define the observations as mappings of the form

$$\mathcal{O}^i : \Omega^u \times \vec{\mathcal{A}} \times \mathbb{N} \to \mathcal{E}^i \tag{3.10}$$

satisfying consistency as in Definition 2. We then set $\vec{\mathcal{O}} = (\mathcal{O}^1, \mathcal{O}^2, \dots)$.

Next, we formally acknowledge the existence of learning algorithms. We say a collection of functions $\mathfrak{L}^i_\varphi$ for $\varphi \in \{T, p, \Gamma, F, \phi, \pi\}$ is the learning algorithm of player i. Recall that $\mathcal{M}_T, \mathcal{M}_p, \mathcal{M}_\Gamma, \mathcal{M}_F, \mathcal{M}_\phi, \mathcal{M}_\pi$ respectively denote the spaces of estimations as in (3.1), (3.2), (3.4), (3.5), and (3.7) respectively. Then, learning

algorithms are in general of the form

$$\mathfrak{L}^i_\varphi : \mathcal{E}^i \times \Pi_{\tilde{\varphi}} \mathcal{M}_{\tilde{\varphi}} \times \mathcal{M}_\Upsilon \to \mathcal{M}_\varphi, \quad \forall \varphi \in \{T, p, \Gamma, F, \phi, \pi\}. \tag{3.11}$$

Let us remark that, although it was not necessary, we introduced Υ explicitly given the other estimations. In its given form, it is a value-based approach in reinforcement learning (see Section 5). However, learning algorithms in (3.11) are crucial, and we do not attempt to simplify them. For example p and Γ might be defined directly from observations by keeping frequency statistics. Although this allows them to be more trackable, they are limited to simple settings. Our motivation in this work is to emphasize their inherent complexity instead. They must be subject to evaluations of their respective objectives, such as in Definition 5.

Lastly, to introduce the estimates and the planned behavior of players, we denote the priors as

$${}^0\vec{I} \in \vec{\mathcal{A}}, \quad \text{and} \quad {}^0\vec{\mathfrak{L}}_\varphi \in \vec{\mathcal{M}}_\varphi, \quad \forall \varphi \in \{T, p, \Gamma, F, \phi, \pi\}.$$

This sets ${}^0\vec{\Upsilon}$ as in (3.9). Then,

$$\begin{aligned}
{}^n\Upsilon^i &:= \Upsilon({}^n\mathfrak{L}^i_\varphi) \in \mathcal{M}_\Upsilon, \quad \forall i \in \mathbb{N}_0 \\
{}^n\mathfrak{L}^i_\varphi &:= \mathfrak{L}_\varphi({}^n\mathcal{O}^i, {}^{n-1}\mathfrak{L}^i_\varphi, {}^{n-1}\Upsilon^i) \in \mathcal{M}_\varphi, \quad \forall \varphi \in \{T, p, \Gamma, F, \phi, \pi\}, i \in \mathbb{N}_0 \\
{}^n\vec{\mathcal{O}} &:= \mathcal{O}(\omega^u, {}^{n-1}I, n) \in \vec{\mathcal{E}}, \\
{}^n\vec{I} &:= ({}^0\vec{I}, {}^1\vec{\Upsilon}(\hat{\omega}, t, x), \dots, {}^n\vec{\Upsilon}(\hat{\omega}, t, x)) \in \vec{\mathcal{A}}.
\end{aligned} \tag{3.12}$$

Also, ${}^n(\mathfrak{L}_T, \mathfrak{L}_p, \mathfrak{L}_\Gamma, \mathfrak{L}_F, \mathfrak{L}_\phi)^i$ defines ${}^nJ^i(t, x; \alpha)$ as in (3.6), and we may use the notation ${}^n\hat{\varphi}^i := {}^n\mathfrak{L}^i_\varphi$ if convenient.

Definition 7 (Uncertain Equilibrium of Discrete Games) *We say that players* $(\vec{\mathcal{O}}, \vec{\mathfrak{L}}_\varphi, \vec{\Upsilon})$ *admit* ${}^*\vec{\Upsilon} \in \vec{\mathcal{M}}_\Upsilon$ *as an* (ε, r, δ)*-uncertain equilibrium under metrics* d^i *on* $\mathcal{M}^i_\Upsilon$*, if for any prior* ${}^0\vec{I} \in \vec{\mathcal{A}}$*,*

(i)

$$\limsup_{n\to\infty} \sup_{i\in\mathbb{N}_0} \int_{\hat{\Omega}} \Big(\sup_{\tilde{\alpha}\in\mathcal{A}^i} {}^nJ^i(\hat{\omega}, t, x, \tilde{\alpha}) - {}^nJ^i(\cdot, {}^n\mathfrak{L}^i_\pi)(\hat{\omega}, t, x) \Big) \hat{\mathbb{P}}(d\hat{\omega}) \le \varepsilon,$$

(ii)

$$\mathbb{P}^u\Big(\liminf_{n\to\infty} \sup_{i\in\mathbb{N}_0} d^i({}^*\Upsilon, {}^n\Upsilon^i) > r \Big) \le \delta$$

Note that condition (iii) aligns with the recurrence defined in Definition 4. We further impose condition (ii) on estimations to obtain a more favorable notion of equilibrium. Later in this section, we will introduce an additional learning parameter and discuss how to incorporate it into this definition.

Let us revisit the example of chess. Consider a well-trained chess player, and suppose the game is nearing its end. At this late stage, there are often configurations in which subsequent moves are certain. That is, a particular action has an induced probability of one, and remains unchanged as the player continues to learn. Similarly, at the opening of the game, the player might have a distribution over different openings. Although we will not observe the same opening in each game, for a well-trained player this distribution may evolve only over long time scales. On the other hand, there may be many configurations where learning continues indefinitely. We remark that d is an arbitrary metric on functions of the form $\{\hat{\Omega}^i \times \mathbb{T} \times \Omega \to \mathbb{A}^i\}$, which can be designed to reflect these considerations.

Notice that we require players to approach a particular behavior independent of their previous history. Such independence implicitly requires that players explore sufficiently diverse behaviors to be able to reach this equilibrium. Furthermore, since players may be exploitable, they often change their behaviors. However, an equilibrium is one that recurs infinitely often.

To briefly elaborate on how players might solve their own optimization problems, they may do so by revisiting past observations with evolving estimations. As the player accumulates observations (that is, as n increases) and recalls past observations, the exploration of potential scenarios under the current estimations ${}^n\hat{\varphi}^i$ aims to capture the term $\sup_{\alpha \in \mathcal{A}^i} {}^n J^i(\hat{\omega}, t, x, \alpha)$. During this revaluation and exploration process, new strategies may be discovered, or an updated assessment of value might lead to changes in ${}^n\hat{\pi}^i$. Condition (ii) in the definition of uncertain equilibrium implies that players have explored potential strategies and are capable of generating the best ones under various scenarios of $\hat{\Omega}$ as learning continues.

On the other hand, the values of the player are driven externally. Not only is there no universal notion of what holds higher value, but values often involve multiple, conflicting objectives. These are shaped by needs, interactions, and self-evaluations, as reflected in the remarkable diversity of values across individuals and societies. In settings that we design, scalar values are again externally assigned to observations, and the value function is a representation of these assignments.

The choice of liminf instead of limsup is important. As mentioned in the Introduction and in Section 2, we interpret convergence of behavior as cooperation between players. By using liminf, we require that a particular behavior remains favorable and is used infinitely often, though not necessarily always.

Let us briefly explain the role of (ε, r, δ), which we take to be uniform over players for simplicity. Firstly, ε characterizes how effectively players can solve

their respective optimization problem. Next, r measures how closely the distribution of strategies approaches the equilibrium infinitely often. Lastly, δ reflects the likelihood that an equilibrium will be observed. Recall that $\mathbb{P}^u$ is associated with the universe in which the players exist; this may depend on the real underlying dynamics, and random choices generated by each player.

We remark that the planned behavior ${}^n\Upsilon^i(\cdot, t, x)$ in (3.12) is a function of the form $\mathbb{N} \times \Omega^u \to \mathbb{A}^{t,x;i}$, thus a discrete-time stochastic process taking values in the space of actions available at (t, x). From this perspective, an uncertain equilibrium can be viewed as a recurrent point of this process. The primary interest lies in understanding the evolution of this process under a specific design of player.

We now clarify the similarities to and differences from the concept of correlated equilibrium. To do so, we examine the optimality condition in Definition 7 of uncertain equilibrium (simplifying notation by omitting n n) as follows:

$$\int_{\hat{\Omega}} J^i(\hat{\omega}, \hat{\pi}^i(\hat{\omega}))\hat{\mathbb{P}}(d\hat{\omega}) = \int_{\hat{\Omega}} \int_{\vec{\mathcal{A}}} J^i(\hat{\omega}, \vec{\alpha})\hat{\Gamma}^i(\hat{\pi}^i(\hat{\omega}); d\vec{\alpha})\hat{\mathbb{P}}(d\hat{\omega})$$

and since $\hat{\Gamma}^i \in \mathcal{P}(\vec{\mathcal{A}})$ (see (3.4)),

$$\int_{\hat{\Omega}} \hat{\Gamma}^i(\hat{\pi}^i(\hat{\omega}); d\vec{\alpha})\hat{\mathbb{P}}(d\hat{\omega}) \in \mathcal{P}(\vec{\mathcal{A}}) \tag{3.13}$$

To connect with the concept of correlated equilibrium, consider any $\rho \in \mathcal{P}(\vec{\mathcal{A}})$. By disintegrating ρ with respect to its i-th component as $\rho(d\vec{\alpha}) = \rho^{-i}(d\vec{\alpha}|\alpha^i)\rho^i(d\alpha^i)$, we identify that ρ^{-i} corresponds to $\hat{\Gamma}^i$ and ρ^i corresponds to $\hat{\mathbb{P}}(\hat{\omega})$.[2] Roughly speaking, the equilibrium conditions can then be expressed (using simplified notations) as follows:

Nash-type Equilibrium:	$\int_{\mathcal{A}^i} \int_{\vec{\mathcal{A}}} \sup_{\tilde{\alpha}^i} J^i(\tilde{\alpha}^i, \vec{\alpha}^{-i})\rho^{-i}(d\vec{\alpha}\|\alpha^i)\rho^i(d\alpha^i)$
Correlated Equilibrium:	$\int_{\mathcal{A}^i} \sup_{\tilde{\alpha}^i} \int_{\vec{\mathcal{A}}} J^i(\tilde{\alpha}^i, \vec{\alpha}^{-i})\rho^{-i}(d\vec{\alpha}\|\alpha^i)\rho^i(d\alpha^i)$
Uncertain Equilibrium:	$\int_{\hat{\Omega}} \sup_{\tilde{\alpha}^i} \int_{\vec{\mathcal{A}}} J^i(\hat{\omega}, \tilde{\alpha}^i, \vec{\alpha}^{-i})\hat{\Gamma}^i(\tilde{\alpha}^i; d\vec{\alpha})\hat{\mathbb{P}}(d\hat{\omega})$
Coarse Correlated Equilibrium:	$\sup_{\tilde{\alpha}^i} \int_{\mathcal{A}^i} \int_{\vec{\mathcal{A}}} J^i(\tilde{\alpha}^i, \vec{\alpha}^{-i})\rho^{-i}(d\vec{\alpha}\|\alpha^i)\rho^i(d\alpha^i)$

The supremum over $\tilde{\alpha}^i$ has different dependence in each equilibrium concept. In the Nash-type equilibrium[3], it depends on $\vec{\alpha}^{-i}$; in the correlated equilibrium, it

[2] One can extend ρ^{-i} to the full $\vec{\mathcal{A}}$ by the Dirac distribution on α^i for the i-th marginal, which is also the case for $\hat{\Gamma}^i$.

[3] We refer to this as a Nash-type equilibrium due to its structure; however, as it is potentially impossible to satisfy, such a concept does not, to our knowledge, appear in the literature.

depends on α^i; in the uncertain equilibrium, it depends on $\hat{\omega}$; and in the coarse correlated equilibrium, it is independent of the controls. We replace $\hat{\pi}$ with $\sup_{\tilde{\alpha}^i}$ to reflect the optimality condition, and $\hat{\mathbb{P}}$ plays a role analogous to ρ^i. However, the key difference is that the correlated equilibrium, similar to the Nash equilibrium, considers deviations that do not affect other players, whereas in uncertain equilibrium, changing one's strategy influences others via learned estimations.

Recall that if regret, defined analogously to the correlated equilibrium, is sublinear, then the time-averaged empirical distribution of actions converges to a correlated equilibrium. Considering the estimated distributions over strategies in (3.13), one might expect all players to eventually induce the same distribution in symmetric situations. However, in general, there is no reason to expect that a single distribution characterizes every player's considerations. Moreover, a similar criticism applies to regret definitions, since changing prior actions would typically influence the future strategies of others.

The concept of Nash equilibrium focuses solely on controls, according to their associated scalar values, inherently excluding the intrinsic structure of a player. For example, in situations where a central planner announces policies for individual agents, such as environmental regulations, traffic management, public health initiatives, with the knowledge that every individual will act according to their own assessment of value, the Nash equilibrium is the appropriate framework. The central planner needs to model agents' individual values to construct stable policies. In such cases, it is not meaningful to model each and every player in detail through their learning algorithms. We also remark that the Nash equilibrium requires players to have exact knowledge of the strategies of others. In the pure equilibrium sense, this instability is significant enough that an equilibrium may fail to exist even in the simplest games. To address this, one typically adopts relaxed controls, which is essential, though not motivated by player design.

Let us highlight the importance of incorporating additional learnable parameters into our framework. This could include encoding raw observations, establishing communication protocols, and many other spaces of estimations to design more sophisticated player. One such simple yet interesting parameter is a player's expectation regarding the best achievable outcome, which defines a notion of regret for the player and alters the characteristics of exploration, as demonstrated in simplified form in Section 3.1. Consider functions of the form

$$\hat{B} : \mathbb{T} \times \Omega \to \mathbb{R} \tag{3.14}$$

and all the related definitions similar to other learning parameters. Then, we can introduce;

$$^n\kappa^i(t,x) := \hat{\mathbb{P}}\left({}^nJ^i(\cdot, {}^n\hat{\pi}^i)(\hat{\omega}, t, x) > {}^n\hat{B}^i(t,x) \right)$$

To relate this quantity to familiar concepts with which we all can relate, we say that at the state $(t,x) \in \mathbb{T} \times \Omega$, the player i is currently desperate if ${}^n\kappa^i(t,x) = 0$, and euphoric if ${}^n\kappa^i(t,x) = 1$. If the player is desperate, as the learning progresses, either ${}^n\hat{B}^i$ will decrease, leading the player in some sense to accept the situation, or ${}^nJ^i$ will assign higher values to underexplored strategies, encouraging the player to explore them. One can further describe the player's current situation using verbal subcategories like

Desperate – Discouraged – Doubtful – Determined – Confident – Optimistic – Euphoric

which can be interpreted as partitions of κ-values. Beyond providing a richer characterization of a player, this notion can be incorporated into Definition 7 of equilibrium by requiring

(ii')
$$\limsup_{n\to\infty} \sup_{i\in\mathbb{N}_0} {}^n\kappa^i(t,x) > \kappa,$$

and denoting it as $(\kappa, \varepsilon, r, \delta)$-uncertain equilibrium. In other words, we now search for player designs that further achieve confidence.

In addition to the design of the player, the choice of equilibrium is also diverse. For example, to recover the concept of Nash equilibrium, we can assume

(ii") for all $i \in \mathbb{N}_0$ and $n \in \mathbb{N}$ large enough,
$${}^n\hat{\Gamma}^i(t,x) = \mathrm{Law}({}^n\hat{\pi}^1(\cdot,t,x)) \times \mathrm{Law}({}^n\hat{\pi}^2(\cdot,t,x)) \times \cdots$$

where we extend $\hat{\Gamma}$ to depend naturally on x, while removing dependence on α. That is, players' estimates yield the strategies of others, effectively meaning that players are able to observe each other's future strategies. If every estimate is predetermined as part of the setting and no randomness is involved, then, together with condition (ii) in Definition 7, we recover that the players' policies $\hat{\pi}^i$ form a Nash equilibrium.

Next, time-consistency, or Dynamic Programming Principle, can be naturally introduced for any learning parameter and can likewise be required at equilibrium. The most important and familiar one is for the value function: We say that the estimate $(\hat{T}, \hat{p}, \hat{\Gamma}, \hat{F}, \hat{\phi}, \hat{\pi})$ yields a time consistent value almost surely, if for any $(t,x) \in \mathbb{T} \times \mathbb{S}_t$ and $0 \le T_0 \le \hat{T}(t,x)$, it holds $d\hat{\mathbb{P}}$-a.s. that

$$\int_{\mathcal{A}^i} J(T_0;\hat{\omega},t,x,\alpha)\hat{\pi}(\hat{\omega},t,x)(d\alpha) = \int_{\mathcal{A}^i} J(\hat{\omega},t,x,\alpha)\hat{\pi}(\hat{\omega},t,x)(d\alpha) \quad (3.15)$$

where $J(T_0; \hat{\omega}, t, x, \alpha)$ is defined exactly as in (3.6), but $\hat{T}$ is replaced by T_0. Notice that when $T_0 = t$ and $\hat{\pi}$ yields the optimal control for each $\hat{\omega} \in \hat{\Omega}$, (3.15) becomes

$$\begin{aligned}\hat{\phi}(t,x) &= \sup_{\alpha} J(t,x,\alpha) \\ &= \sup_{\alpha} \int_{\vec{\mathcal{A}}} \mathbb{E}^{t,x,\vec{\alpha}}\Big[\hat{\phi}(t+\hat{T}, X_{t+\hat{T}}) + \sum_{s=t}^{t+\hat{T}-1} \hat{F}(s, X_s, \vec{\alpha}(s, X_s))\Big]\hat{\Gamma}_t(\alpha; d\vec{\alpha})\end{aligned}$$

which closely resembles the standard time-consistency, or Dynamic Programming Principle, for the standard value function.

To illustrate how time consistency can be required of an estimation, we focus on $\hat{\pi}$; similar conditions can be formulated for $\hat{T}$, $\hat{p}$, and $\hat{\Gamma}$ as well.[4] The idea is straightforward: the distribution induced by $\hat{\pi}(\cdot, t, x)$ given a future state (T_0, x_{T_0}), should be the same as if the policy were formed at the state (T_0, x_{T_0}). Formally, we say that the estimate $(\hat{T}, \hat{p}, \hat{\Gamma}, \hat{F}, \hat{\phi}, \hat{\pi})$ yields a time-consistent distribution over controls almost surely at $(t, x) \in \mathbb{T} \times \mathbb{S}_t$ if, for any $t \leq T_0 \leq t + \hat{T}(t, x) - 1$, and $x_{T_0} \in \mathbb{S}_{T_0}$, $\hat{\pi}(\hat{\omega}, t, x)$ induces the same distribution as $\hat{\pi}(\hat{\omega}, T_0, x_{T_0})$ on the space $\mathcal{A}^{\hat{T}(t,x);T_0,x_{T_0},i}$. Here, the quotient space is defined similarly, with the relation terminating at $t + \hat{T}(t, x) - 1$ instead of $t + \hat{T}(T_0, x_{T_0}) - 1$.

3.1 Two player game example

In this section, we present a simple, repeatedly played two-player example. We demonstrate that even with a fixed one-step horizon, players can exhibit non-stationary dynamics. In this setting, both players learn transition costs $\hat{F}$ and the actions of their opponents $\hat{\Gamma}$, all within a fixed horizon $T = 1$. Our central argument is that formulating controls as an equilibrium does not adequately capture the dynamic strategies continually employed by the players. To address this shortcoming, we construct learning algorithms that capture these dynamic strategies. This example illustrates why a more general framework is necessary for effectively modeling games, and it also heuristically highlights how the concept of equilibrium is inherently shaped by the learning process and by the opponents themselves.

Consider fixed state and action spaces given by $\mathbb{S} = \{0, 1\}$ and $\mathbb{A} = [0, 1]$. Players' actions determine their transition probabilities at each step, and they can only observe each other's state. The first player loses \$1 if the second player appears in state 1 but gains by increasing their own transition probability to state 1.

[4] In our case, $\hat{p}$ is a single-step transition probability and is therefore automatically consistent.

The second player loses \$1 if they are not in the same state. [5]

Let us note that, with a one-step horizon, there exists a unique Nash equilibrium that the first player is unwilling to play. Of course, by virtue of Folk's theorem, any feasible outcome can be sustained in an infinitely repeated game. However, this result explicitly relies on the assumption that players are certain about their opponents' future actions over an indefinite horizon. This strong assumption allows for almost any feasible value to be supported as an equilibrium, leaving the question of which outcome will be observed without a clear answer. Moreover, since players do not announce their strategies, searching for a Nash equilibrium with a larger horizon does not necessarily model this game either. Such a search represents our external attempts to formalize the players' incentives. Instead, we emphasize once again that the core element in games is the learning algorithms employed by the players. These algorithms naturally govern the (random) evolution of probability distributions over actions, which in turn is sufficient to understand the evolution of the game.

Now, let us formally state the game. Consider fixed state and action spaces given by $\mathbb{S} = \{0, 1\}$ and $\mathbb{A} = [0, 1]$. Suppose the players are not learning the horizon or transition probabilities, that is, $\mathfrak{L}_T^i, \mathfrak{L}_p^i$ are constants yielding $\hat{T}^i = 1$ and $p^i(t, \vec{x}, \vec{a}, 1) = a^i$. Initially, to specify the rules of the game and enable comparison with the Nash equilibrium, we assume that the players are not learning the state value or transition costs;

$$\phi^1(t+1, X_{t+1}^1, X_{t+1}^2) = -\mathbf{1}_{\{X_{t+1}^2 = 1\}}, \qquad F^1(t, X_t^1, X_t^2, a^1, a^2) = ca^1,$$
$$\phi^2(t+1, X_{t+1}^1, X_{t+1}^2) = -\mathbf{1}_{\{X_{t+1}^1 \neq X_{t+1}^2\}}, \quad F^2 = 0$$

As mentioned, the first player wants the second player to move state 0, and the second player wants to be in the same state as the first. However, since $c > 0$, the first player gains by increasing the oods of moving to state 1. Now, costs of each player are

$$J^1(t, \vec{x}, \vec{a}) = ca^1 - \mathbb{P}^{t,\vec{x},\vec{a}}(X_{t+1}^2 = 1), \quad J^2(t, \vec{x}, \vec{a}) = -\mathbb{P}^{t,\vec{x},\vec{a}}(X_{t+1}^1 \neq X_{t+1}^2),$$
$$\mathbb{P}^{t,\vec{x},\vec{a}}(X_{t+1}^2 = 1) = a^2, \qquad \mathbb{P}^{t,\vec{x},\vec{a}}(X_{t+1}^1 \neq X_{t+1}^2) = a^1 + a^2(1 - 2a^1) \tag{3.16}$$

From now on, since the one-step game does not depend on the current time or state, we will drop them from notations. Let us also note that players make their decisions simultaneously. One could instead formulate the game as turn-based, but we aim to keep it as symmetric as possible, while excluding only the cost structure.

[5] We mention their gains and losses in dollar amounts, to relate easily to a potential game we might play in real life.

Note that when the horizon is fixed at 1, there exists a unique Nash equilibrium given by $\vec{a} = (1, 1)$. Although such an equilibrium exists, it is not necessarily useful for characterizing the potential behaviors of the players. Once the first player fixes the action of the second player, they lose awareness of the latter's underlying intentions.

Let us recap how the game is played from the perspective of the first player. First, we determine a probability a^1 of transitioning to state 1, and receive a payoff ca^1. Then, we lose \$1 if the other player ends up in state 1. From the perspective of the second player, objective is simply to follow the other player. We observe the past states of the other and attempt to end up in the same state, losing \$1 otherwise. Due to the simplicity of the game, behaviors that are expected to recur over time can be generated. Starting with the second player, whose cost structure is simpler:

- Determine an acceptable level of noise in observing the other player's state, based on expectations. If recent observations consistently yield a particular outcome (0 or 1) within the acceptable noise level, take the corresponding action. Otherwise, begin exploring other actions, with rationale to penalize the noise.

For the first player, the cost structure is slightly more intricate;

- If the second player consistently appears in state 0, explore larger actions to reduce the cost (due to ca^1). Continue increasing it until the second player begins to appear in state 1 frequently enough to offset these gains.

- If the second player appears at 1, which is costly, switch to actions that are not used recently. Continue exploring until the second player reappears in state 0 regularly enough to keep the realized cost consistent with expectations.

In the next section, we construct a learning algorithm and numerically explore how the corresponding behaviors evolve. We remark that a straightforward Q-learning algorithm can be used to model the players. However, Q-learning models only expected rewards for each action, it therefore converges to fixed actions and lacks the underlying dynamics we aim to demonstrate. This underscores the same point: it is crucial to incorporate the design of the player to understand games.

3.1.1 Details of the Learning Algorithm

We now introduce the relevant parts of the framework specific for this problem. In general, each player models a transition cost $\hat{F}$ and an estimate $\hat{\Gamma}$ on the other

player's actions, relying on their observations held in the memory. Also, each player has an expectation $\hat{B}$ as in (3.14), taken as a constant for simplicity.

Recall that observations of players are the realized states. That is, $\mathcal{E}^1 = \mathcal{E}^2 = \mathbb{S}$, and observations in (3.10) are depending on the realizations of the states. Here, Ω^u and $\mathbb{P}^u$ are determined by the random number generators that determines the transitions of states for the players at each round. We then set the observations $\mathcal{O}^1, \mathcal{O}^2$ in equation (3.10) in an obvious manner. In the simulations, each player keeps a memory of a certain length, recording the realized states and costs.

Now, let $\{\mathcal{N}_\Gamma^{i,k}\}_{k=1}^K$ be the i'th player simple feed-forward networks where $\mathcal{N}_\Gamma^{i,k} : \mathbb{A} \to \mathbb{A}$. We then assing $\mathfrak{L}_\Gamma^i : \mathcal{E}^i \to (\mathbb{A} \mapsto \mathcal{P}(\mathbb{A}))$ in (3.11) as the empirical distribution formed by $\{\mathcal{N}_\Gamma^{i,k}\}_k$'s. That is,

$$ {}^n\hat{\Gamma}^i := \mathfrak{L}_\Gamma^i({}^n\mathcal{O}^i) := \frac{1}{K}\sum_{k=1}^K \delta_{\mathcal{N}_\Gamma^{i,k}} $$

Note that the parameters of networks are depending on the observations, which is not explicit in notations as we view $\mathfrak{L}_\Gamma^i$ yielding the network with such parameters. To train these networks, after each step, players draw a batch of memories using the multinomial distribution with higher weights assigned to recent observations. Then, networks are getting trained to reduce the difference between estimated action and observed state.

Similarly, we denote $\{\mathcal{N}_F^{i,\ell}\}_{\ell=1}^K$, where $\mathcal{N}_F^{i,\ell} : [0,1] \to \mathbb{R}$. We then set $\hat{F} : \hat{\Omega} \times \mathbb{A} \to \mathbb{R}$ in (3.5) as

$$ \hat{F}^1(\ell, \hat{\omega}', a^1) = ca^1 + \mathcal{N}_F^{1,\ell}(\hat{\omega}')(a^1), \quad \hat{F}^2(\ell, \hat{\omega}', a^2) = \mathcal{N}_F^{2,\ell}(\hat{\omega}')(a^2) $$

We identify $(\ell, \hat{\omega}') \in \hat{\Omega} = \{1, \dots, K\} \times \hat{\Omega}'$ where $\hat{\mathbb{P}}$ assigns the first marginal as uniform distribution over $\{1, \dots, K\}$.[6] Each network $\mathcal{N}_F^{i,\ell}$ is further random by the virtue of dropout layers. Keeping the networks always in the training mode, one generates a random function with positive dropout probabilities, and $\hat{\Omega}'$ abstracts this. Here,

$$ \mathfrak{L}_F^i : \mathcal{E}^i \to \big((\hat{\Omega}, \mathbb{A}) \mapsto \mathbb{R}\big) $$

yielding ${}^n\hat{F}^i$ and ϕ^i's are taken as constant.

There are two objectives cost networks are training for: (i) there is an expected cost coming from the predictions of action networks, which is (3.16) integrated

[6] As in the case of action networks, this is only for simplicity. One might assign and evolve weights corresponding to networks, and capture more vibrant dynamics if the game is more sophisticated. For example, one might keep a subset of networks as trusted ones (high weights), and let other networks explore more wildly (low weights).

as in (3.6). If action networks are not perfect, expected cost will not match the observed expected cost. Cost networks are training to close this gap, by relying on costs in the memory. (ii): players have expectations over what is best possible as introduced in (3.14), which we took as constant for simplicity. Cost networks are also trained such that the players do not get desperate. For example, if the first player ends up with networks $\mathcal{N}_\Gamma^{1,k}$'s taking values close to 1, independent of the action, they start to play the Nash equilibrium $(1,1)$. That is, player 1 gets desperate, and then adjusts the random component of cost to increase values of other actions towards $\hat{B}^1$ to start exploring them.

We note that (3.6) becomes

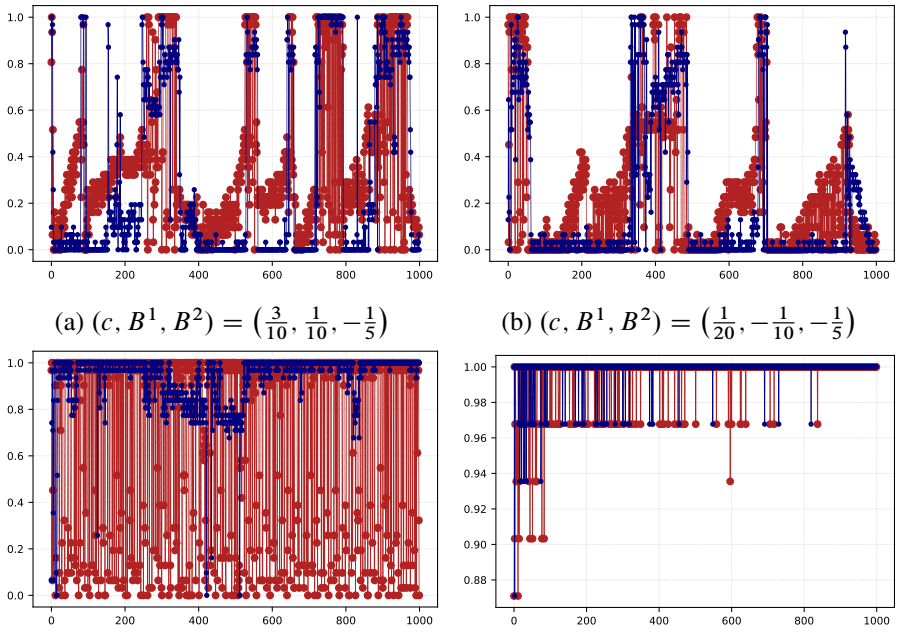
$$J^1(\ell,\hat{\omega}',a^1) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}_\Gamma^{1,k}(a^1) + ca^1 + \mathcal{N}_F^{1,\ell}(\hat{\omega}')(a^1), \text{ and}$$

$$J^2(\ell,\hat{\omega}',a^2) = \frac{1}{K}\sum_{k=1}^{K}\mathcal{N}_\Gamma^{2,k}(a^2) + a^2(1-2\mathcal{N}_\Gamma^{2,k}(a^2)) + \mathcal{N}_F^{2,\ell}(\hat{\omega}')(a^2)$$

To draw an action from (3.9), players uniformly choose one cost network $\ell \in \{1,\dots,K\}$, and observe one realization $\hat{\omega}'$ coming from dropout layers. Then, $\Upsilon^i = \hat{\pi}^i$ simply minimizes J^i over a^i yielding ε-optimal action, and plays it.

Before discussing the simulation results, let us mention that each parameter of networks of course plays a significant role, and we coarsely tuned them by hand to obtain simulations matching with expectations. Many of such parameters are taken as constant. Changing to different constants might of course yield poor results. On the other hand, generalizing them will improve the sophistication of agents strategies. Besides the network parameters, there are more structural parameters too. For example, what players are expecting as the best possible cost also changes the characteristics of actions. Especially if the first player is expecting much better than what is realistically possible, exploration gets out of hand. For an another example, if the memory is very long and not forgetting, then the first player starts to get advantage over the second player, as the second player becomes fixed after a while and estimates that the other player will still frequently move to state 0. The point we are aiming to convey here is that the learning algorithms of players are crucial to characterize what is going to be realized in reality.

Now, let us annotate the simulation results. In Figure 1, the actions taken by Player 1 (red) and Player 2 (blue) over 1000 games are plotted, with both players starting from arbitrary initial estimations. While keeping Player 2's parameters constant, four plots illustrate variations in c, the incentive parameter for Player 1, and B^1, as defined in (3.14), which represents the expectation of Player 1. Thin lines in the plots indicate the jumps between actions.



(a) $(c, B^1, B^2) = \left(\frac{3}{10}, \frac{1}{10}, -\frac{1}{5}\right)$ (b) $(c, B^1, B^2) = \left(\frac{1}{20}, -\frac{1}{10}, -\frac{1}{5}\right)$

(c) $\left(c, B^1, B^2\right) = \left(1, 1, -\frac{1}{5}\right)$ (d) $\left(c, B^1, B^2\right) = \left(1, 0, -\frac{1}{5}\right)$

Figure 1: Actions of Player 1 (red) and Player 2 (blue) over 1000 games, demonstrating the impact of varying the incentive parameter c and expectation B^1 for Player 1, while Player 2's parameters remain constant. Each subplot shows how different incentives and expectations influence Player 1's strategy and interaction dynamics in this toy problem.

In subplot (a), Player 1 has a somewhat large incentive ($c = 3/10$) to take larger actions, aiming for a reward of $B^1 = 1/10$. Thus, playing close to $(0, 0)$ does not suffice, and Player 1 searches for higher rewards, leading to frequent changes between different phases. Notice that as soon as (a^1, a^2) approaches $(1, 1)$, Player 1 begins to explore and pushes the game back towards $(0, 0)$. In subplot (b), the incentive is much smaller ($c = 1/20$) and the expectation of Player 1 is decreased to $B^1 = -1/10$. Consequently, Player 1 stays close to $(0, 0)$ for longer, before starting to think that Player 2 will always choose action 0. In subplots (c) and (d), the incentive for Player 1 is really high ($c = 1$), making deviations from $(1, 1)$ unnecessary. Specifically, in subplot (d), Player 1 is satisfied with a reward of 0, maintaining $(1, 1)$ almost always. Conversely, in subplot (c) where Player 1 expects to get an unrealistic reward of 1, which requires Player 1 to play 1 while Player 2 plays 0, exploration by Player 1 leads to worse results for both.

We refer to the supplementary online repository [27] for the animation showing the cost networks, action networks, and other observables in each case. We point out that these estimations are not converging networks, instead dynamic and yielding repeating patterns of behaviors.

We conclude this section by emphasizing once more that games are inherently complex and that observed behaviors require a detailed representation of players. In our working paper [28], we explore firm collusion in an online price competition setting. In Calvano et al. [29], the authors model each firm using Q-learning. Each firm maintains a single Q-function estimate, requiring an intensive training period to populate the Q table, an approach that is infeasible in online learning situations and offers little intuition about why collusion emerges. In [28], we design firms as outlined in this section. Beyond observing rapid convergence, we aim to explain how collusion, or competition, is driven by firms' design choices, potentially offering policymakers better guidance for regulating price competition.

4 Stated Mean-Field Games

In this section, we will introduce a mean-field type version of the framework. It is important to note that learning parameters are defined for each player individually. Therefore, embedding a mean-field game requires adjusting the learning parameters of a representative agent to model infinitely many similar players. In particular, to align with a similar structure in the literature, we assume that only $\hat{\Gamma}$ will be learned, while other parameters $(\hat{T}, \hat{p}, \hat{F}, \hat{\phi}) = (T, p, F, \phi)$ are modeled by the representative player as known (and not learned). We will thus refer to this case as the stated mean-field game.

Let the state space be $\mathbb{S}_t$, $\mathbb{S} := \bigcup_{t\in\mathbb{T}} \mathbb{S}_t$, and $\mathbb{A}$ be the common action space. Set the canonical space $\Omega := \prod_{t\in\mathbb{T}} \mathbb{S}_t$ and introduce the set of controls as

$$\mathcal{A} := \left\{\alpha : \mathbb{T}\times\Omega\times\mathcal{P}(\Omega)\to\mathbb{A} \;:\; \alpha(t,x,\mu)\in\mathbb{A},\ \forall (t,x,\mu)\in\mathbb{T}\times\mathbb{S}_t\times\mathcal{P}(\mathbb{S}_t)\right\}$$

As before, we require any function on $\mathbb{T}\times\Omega\times\mathcal{P}(\Omega)$ to be Markovian. Transition probabilities are given as

$$p(t,x,\mu,a;y) : \mathbb{T}\times\Omega\times\mathcal{P}(\Omega)\times\mathbb{A}\times\mathbb{S}\to\mathbb{R}^+, \quad \text{where}$$
$$p(t,x,\mu,a;\cdot) \text{ is a probability measure on } \mathbb{S}_t, \text{ for all } t\in\mathbb{T}, x\in\mathbb{S}_t, \mu\in\mathcal{P}(\mathbb{S}_t)$$

Next, along the idea that the representative agent is insignificant in the population, we assume $\hat{\Gamma}$ in (3.4) is constant as $\hat{\Gamma} : \mathbb{T}\to\mathcal{P}(\mathcal{P}(\mathbb{S}_t\times\mathcal{A}))$. Here, $\mathcal{P}(\mathbb{S}_t\times\mathcal{A})$ corresponds to $\vec{\mathcal{A}}$ in (3.4). In the case of countable players, indexing was keeping track of the connection between the state and the control of individual players. Here, state variable keeps track of distribution of controls used by the population.

Now, given a particular estimation of the population $\Xi_t \in \mathcal{P}(\mathbb{S}_t \times \mathcal{A})$ by the representative player at time t, introduce $\Xi_s \in \mathcal{P}(\mathbb{S}_s \times \mathcal{A})$ recursively as

$$\Xi_{s+1}(dy, d\alpha) = \int_{\mathbb{S}_t} p(s, x, \mu_s^{\Xi}, \alpha(s, x, \mu_s^{\Xi}); dy) d\,\Xi_s(x, d\alpha), \ \forall t \leq s, \quad (4.1)$$

where $\mu_s^{\Xi} := \Xi_s(\cdot, \mathcal{A})$. Note that μ^{Ξ} corresponds to (3.3) for the population. If the second marginal of Ξ is a Dirac measure δ_α independent of the state, we call it homogeneous, as it models every individual player using a single control α. Otherwise, we call it heterogeneous. In the homogeneous case, we do not need to keep track of the flow of the distribution of controls. Moreover, in the heterogeneous case, one can represent the flow of the population μ^{Ξ} using a single relaxed control instead of a distribution of controls. See [30] for the details.

Next, introduce the flow of the distribution for the representative player;

$$\mathbb{P}^{t,\Xi;x,\alpha}(X_{s+1} = dy | X_s = \tilde{x}) = p(s, \tilde{x}, \mu_s^{\Xi}, \alpha(s, \tilde{x}, \mu_s^{\Xi}); dy) \ \ \forall t \leq s, \quad (4.2)$$

with initial condition $\mathbb{P}^{t,\Xi;x,\alpha}(X_t = x) = 1$ where X is the canonical process. Notice that the player is observing the distribution of the population μ^{Ξ}, given the initial data $\Xi \in \mathcal{P}(\mathbb{S}_t \times \mathcal{A})$.

Recall that we assume the cost is known and not learned. Moreover, while defining (3.6), we started from the initial state x, and here we similarly start from the current distribution $\mu \in \mathcal{P}(\mathbb{S}_t)$. We will restrict the learning algorithm to yield $\hat{\Gamma}$ with its marginal on $\mathbb{S}_t$ as a Dirac measure at μ. Then, similar to (3.6), we assume the cost structure is given by

$$J(t, \mu; x, \alpha) := \int_{\mathcal{P}(\mathbb{S}_t \times \mathcal{A})} J(t, \Xi; x, \alpha) d\,\hat{\Gamma}_t(\Xi), \ \text{ where } \ \hat{\Gamma}_t((\mu, \mathcal{P}(\mathcal{A}))) = 1, \text{ and}$$

$$J(t, \Xi; x, \alpha) := \mathbb{E}^{t,\Xi;x,\alpha}\Big[\phi(X_{t+T}, \mu_{t+T}^{\Xi}) + \sum_{s=t}^{t+T-1} F(s, X_s, \mu_s^{\Xi}, \alpha(s, X_s, \mu_s^{\Xi}))\Big], \quad (4.3)$$

Set $\mathcal{M}_J$ as the space of all such functions $(\mathbb{T} \times \mathcal{P}(\Omega) \times \Omega \times \mathcal{A} \to \mathbb{R})$. Let us note that, we are mainly interested in the static $\{0, \dots, T\}$ problem for simplicity. One can dynamically set $\hat{T} = T - t$ (and repeats after T) by the learning algorithm to create a dynamic version. Or one might evolve the game indefinitely, keeping the $\hat{T}$ fixed if the structure allows it.

Given the cost, we now need to estimate the optimal controls by the learning parameter

$$\hat{\pi} : \mathbb{T} \times \mathcal{P}(\Omega) \times \Omega \to \mathcal{A}$$

There is no randomness in the value, and assuming that the value is sufficiently representative, we don't further impose randomness in the policy. Thus, given that

representative player is able to solve for the optimal control, $\hat{\pi}$ becomes deterministic, taking values on the set of optimal controls. Moreover, behavior is simply $\Upsilon(\hat{\pi}) = ((t, \mu; x) \mapsto \hat{\pi}(t, \mu; x))$. Due to the time-consistency, optimal policy determined at an initial condition stays optimal, but we will rely only on $\hat{\pi}$ to generate observations rather than the behavior.

Let us briefly recap the mean-field framework. We assume that the representative player starts with an initial guess of the population distribution over states and controls, determines the corresponding optimal control, and, relying on the assumption that everyone else is exactly the same, generates further observations using the chosen learning algorithm. Equivalently, one can say that there are infinitely many such players actually playing the game and observing the distribution of players, however, our framing is more consistent with applications. For example, let us consider a portfolio liquidation problem that someone faces in a financial market. Instead of solving an optimization problem without acknowledging that there are other players also facing a similar problem, as a first order approach without actually having information about other players, the player can model there is a distribution of others facing exactly the same problem.

Finally, we are ready to introduce the observations and the learning algorithm. Let $\mathcal{E} = \mathcal{P}(\mathcal{P}(\mathbb{S} \times \mathcal{A}))$ be the space of observables, and let $\mathcal{E}$ denote the space of finite sequences of $\mathcal{E}$. Recall that the learning algorithm is a mapping $\mathfrak{L}_\varphi : \mathcal{E} \to \mathcal{M}_\varphi$, where the player's estimation at age n is ${}^n\mathfrak{L}_\varphi := \mathfrak{L}_\varphi({}^n\mathcal{O})$. Here, $\mathcal{O} : \mathbb{N} \to \mathcal{E}$ represents the increasing sequence of observations. To provide intuition, we will construct a simple but explicit learning algorithm for $\mathfrak{L}_\Gamma$:

Suppose that the current distribution of the population at time 0 is $\mu \in \mathcal{P}(\mathbb{S}_0)$ and is fixed as given. We, as the representative player, start with an initial guess ${}^0\mathcal{O} = {}^0\Xi = \delta_{(\mu, \delta_{{}^0\alpha})}$ for some ${}^0\alpha$. That is, our initial observation is $\delta_{{}^0\Xi}$. Then, we determine the population flow $\mu^{{}^0\Xi}$ using (4.1), our flow $\mathbb{P}^{t, {}^0\Xi; x, \tilde{\alpha}}$ using (4.2), and solve the optimization problem to find an optimal control ${}^1\alpha$.

Now, following the fixed point idea, we learned that if the population is using ${}^0\alpha$, it is optimal to use ${}^1\alpha$. Since every player is equivalent, we may deduce that the population will use ${}^1\alpha$ with some probability c, and use ${}^0\alpha$ otherwise. That is, we set the learning algorithm as

$$\mathfrak{L}_\Gamma(\delta_{(\mu, \delta_{{}^0\alpha})}) = c\delta_{(\mu, \delta_{{}^1\alpha})} + (1 - c)\delta_{(\mu, \delta_{{}^0\alpha})} = {}^1\mathcal{O}$$

Notice that, for simplicity, we are assuming a homogeneous population. That is, everyone is assumed to be using a single control. Once can, for example, formulate it as a portion of the population will use ${}^1\alpha$. Next, we can repeat the same procedure to find another optimal control under the guess $\hat{\Gamma} = \mathfrak{L}_\Gamma(\delta_{(\mu, \delta_{{}^0\alpha})})$, denoted as ${}^2\alpha$, and so on. In general, our naive learning algorithm depends only on the last

observation, defined as

$$\mathfrak{L}_\Gamma\left({}^n\mathcal{O}\right) := c\ \delta_{(\mu,\delta_{n+1\alpha})} + (1-c)\ {}^n\mathcal{O},\ \ 0 \le c \le 1 \tag{4.4}$$

where we took ${}^n\mathcal{O} \in \mathcal{P}(\mathcal{P}(\mathbb{S}_0 \times \mathcal{A}))$ as a single observation rather than a sequence to simplify notation, and ${}^{n+1}\alpha$ is an optimal control under ${}^n\mathcal{O}$.

Let us remark on the similarity between the fictitious play-type algorithms introduced in Cardaliaguet-Hadikhanloo [31]. In fictitious play, one also starts with an initial guess δ_{0_α} and finds the optimal ${}^1\alpha$. A crucial difference, however, is that fictitious play considers the weighted average of ${\mu^0}^\Xi$ and ${\mu^1}^\Xi$ to find the next optimal control α^2. That is, the cost structure becomes

$$J^{\text{fictitious}}(t,\mu;x,\alpha) := J\left(t, \int_{\mathcal{P}(\mathbb{S}_t\times\mathcal{A})} \Xi\, d\hat{\Gamma}(\Xi); x, \alpha\right)$$

for J as in (4.3), with the $\hat{\Gamma}$ induced by the same $\mathfrak{L}_\Gamma$ but solving a different optimization. We leave the question of whether these approaches are equivalent for potential games to future research, which is a key assumption in [31] for the convergence result.

Lastly, we rephrase the definition of uncertain equilibrium with notations as in Section 2, and we explicitly compute the equilibrium under this basic learning algorithm in the next section.

Definition 8 (Uncertain Equilibria of stated Mean-Field Games) *We say that a player $(\mathcal{O}, \mathfrak{L}_\Gamma, \mathfrak{L}_\pi, \Upsilon)$ admit ${}^*\Upsilon \in \mathcal{M}_\Upsilon$ as an (ε,δ)-uncertain equilibrium under the metric d on $\mathcal{M}_\Upsilon$ at $(t,x,\mu) \in \mathbb{T}\times\mathbb{S}_t\times\mathcal{P}(\mathbb{S}_t)$, if for any prior ${}^0\alpha \in \mathcal{A}$,*

(i)

$$\limsup_{n\to\infty}\left(\sup_{\tilde{\alpha}\in\mathcal{A}} {}^nJ(t,\mu;x,\tilde{\alpha}) - {}^nJ(\cdot,{}^n\hat{\pi})(t,\mu;x)\right) \le \varepsilon$$

(ii)

$$\liminf_{n\to\infty} d({}^*\Upsilon, {}^n\Upsilon) \le \delta$$

4.1 One step stated mean-field game examples

We now present two examples in which we can explicitly compute and contrast the relaxed equilibrium and uncertain equilibrium under the learning algorithm described in (4.4). In the first example, while there is no standard Nash equilibrium, a relaxed equilibrium does exist. Conversely, in the second example, due to the cost function being discontinuous, there is no relaxed equilibrium; however, the uncertain equilibrium remains unchanged.

Example 1 *Set* $\mathbb{S} = \{0,1\}$, $\mathbb{T} = \{0,1\}$, *the action space* $\mathbb{A} = [0,1]$, *and the transition probability*

$$p(0,x,a,\mu;1) = a, \qquad p(0,x,a,\mu;0) = 1-a$$

Furthermore, introduce the cost as

$$J(\Xi;\alpha) := \mathbb{E}^{\mathbb{P}^{\Xi,\alpha}}\Big[\phi(X_1,\mu_1^{\Xi}) + F(\alpha)\Big]$$
$$\text{where} \quad \phi(x,\mu) := 2|\mu(1)|^2 - 4\mathbf{1}_{\{x=1\}}\mu(1), \ \text{ and } \ F(a) = (|a|^2 + a)$$

Then, there exists no standard Nash equilibrium and a unique relaxed equilibrium $\frac{1}{2}(\delta_0 + \delta_1)$. *The learning algorithm described in* (4.4) *oscillates around* $\frac{1}{2}(\delta_{\delta_0} + \delta_{\delta_1}) \in \mathcal{P}(\mathcal{P}(\mathbb{A}))$, *and induces an action distribution* δ_0 *or* δ_1 *infinitely often.*

Proof First, let us argue that there exists no standard Nash equilibrium. Main idea is, if the population distribution is symmetric $\mu_1^{\Xi}(1) = \mu_1^{\Xi}(0) = 1/2$, then the optimal actions are $\{0,1\}$. Whenever $\mu_1^{\Xi}(1) > 1/2$, optimal action becomes 0 and otherwise 1. That is, every player tries to stay away from the majority and there cannot be a deterministic fixed point.

As for the standard Nash equilibrium population is homogeneous, (4.1) simplifies to

$$\mu^a(1) := \mu_1^{\Xi}(1) = a$$

whenever the population is taking the action $a \in \mathbb{A}$, independent of the initial distribution. For the representative player, we reserve $\tilde{a} = \alpha(0,x)$ and compute the cost;

$$\begin{aligned} J(a,\tilde{a}) := J(\Xi;\alpha) &= 2|\mu^a(1)|^2 - 4\mu^a(1)\mathbb{P}^{\Xi,\alpha}(X_1 = 1) + |\tilde{a}|^2 + \tilde{a} \\ &= 2a^2 - 4a\tilde{a} + |\tilde{a}|^2 + \tilde{a} \end{aligned}$$

since it is quadratic in $\tilde{a}$, maximum occurs only if $\tilde{a} \in \{0,1\}$. Thus, noting that

$$J(0,\tilde{a}) = |\tilde{a}|^2 + \tilde{a}, \quad J(1,\tilde{a}) = 2 + |\tilde{a}|^2 - 3\tilde{a}$$

there exists no standard Nash equilibrium.

Now, to compute the relaxed equilibrium, we know from [30] that it is equivalent to consider the heterogeneous case. Thus, as the initial distribution is irrelevant, let $\Xi_0 \in \mathcal{P}(\mathcal{A}) = \mathcal{P}(\mathbb{A})$. Since there is only a single time step, the distribution of controls doesn't evolve either and (4.1) becomes

$$\Xi(1,da) := \Xi_1(1,d\alpha) = a\,\Xi_0(da),$$
$$\mu^{\Xi}(1) := \mu_1^{\Xi}(1) = \Xi(1,\mathbb{A}) = \int_{[0,1]} a\,\Xi_0(da)$$

Then,

$$J(\Xi_0;\tilde{a}) := J(\Xi;\alpha) = 2|\mu^{\Xi}(1)|^2 - 4|\mu^{\Xi}(1)|\tilde{a} + |\tilde{a}|^2 + \tilde{a}$$

and in this case, again from [30], equilibrium means that every action in the support of Ξ_0 is optimal. It is easy to check that if $\mu^{\Xi}(1) \neq 1/2$, then the optimal action is either 0 or 1 and there cannot be any equilibrium. If $\mu^{\Xi}(1) = 1/2$, then both 0 and 1 are optimal. Thus, $\Xi_0 = \frac{1}{2}(\delta_0 + \delta_1)$ corresponds to the relaxed equilibrium, since any action in the support is optimal.

Lastly, let us discuss the convergence of (4.4). Consider any $\Gamma = \sum_i c_i \delta_{\delta_{a^i}}$ which is an element of $\mathcal{P}(\mathcal{P}(\mathbb{A}))$ if $\sum_i c_i = 1$ representing any homogeneous estimate for the action of the population. Then, under appropriate notational simplifications of this example, (4.3) becomes

$$\begin{aligned} J(\tilde{a}) &= \int_{\mathcal{P}(\mathbb{A})} J(\Xi,\tilde{a}) d\,\Gamma(\Xi) = \sum_i c_i J(\delta_{a_i},\tilde{a}) \\ &= 2\sum_i c_i |a_i|^2 - 4\tilde{a}\sum_i c_i a_i + |\tilde{a}|^2 + \tilde{a} \end{aligned}$$

which is exactly as before a quadratic polynomial in $\tilde{a}$, hence optimal value occurs at either 0 or 1. Therefore, the algorithm (4.4) will quickly converge to a distribution having $\delta_0, \delta_1 \in \mathcal{P}(\mathbb{A})$ in its support, and the contribution from the initial guess will diminish with the factor $(1-c)^n$. Moreover, as the optimal $\tilde{a}$ becomes 0 or 1 depending on the estimated average of the population similarly as before, the algorithm will oscillate around $\frac{1}{2}(\delta_{\delta_0} + \delta_{\delta_1})$. Note that, by adjusting the constant parameter c in (4.4), one can achieve exact convergence too[7]. Here, since $\hat{\pi}$ is computed exactly as either δ_0 or δ_1, we also observe that the induced distribution oscillates in between them infinitely often. ■

Example 2 *Set* $\mathbb{S} = [0,1]$, $\mathbb{T} = \{0,1\}$, *the action space* $\mathbb{A} = [0,1]$, *and the transition probability*

$$p(0,x,a,\mu;dy) = \delta_a$$

Furthermore, introduce a discontinuous cost as

$$J(\Xi;\alpha) := \mathbb{E}^{\mathbb{P}^{\Xi;\alpha}}\left[X_1^{\alpha}\mathbf{1}_{\left\{\bar{\mu}_1^{\Xi}\in\left[0,\frac{1}{2}\right]\right\}} - X_1^{\alpha}\mathbf{1}_{\left\{\bar{\mu}_1^{\Xi}\in\left(\frac{1}{2},1\right]\right\}}\right]$$

where $\bar{\mu}^{\Xi} := \int_{[0,1]} x\, d\mu^{\Xi}$. *Whereas no relaxed equilibrium exists, the learning algorithm described in* (4.4) *again oscillates around* $\frac{1}{2}(\delta_{\delta_0} + \delta_{\delta_1}) \in \mathcal{P}(\mathcal{P}(\mathbb{A}))$, *and induces an action distribution* δ_0 *or* δ_1 *infinitely often.*

[7] Let us briefly take attention to the importance of the design of the learning algorithm, even for this simple setting. If the parameter c diminishes very fast with n, then one can conclude either δ_0 or δ_1 is optimal depending on the initial condition.

Proof The essence of this example is similar to that in Example 1. It is clear that there exists no relaxed equilibrium, as the optimal action is either δ_0 or δ_1 under any value of $\bar{\mu}_1^{\Xi}$, and neither constitutes an equilibrium.

For the learning algorithm (4.4), although the cost function is computed differently than in Example 1, $\hat{\pi}$ behaves exactly the same, depending on the population average. Thus, there is no difference from Example 1.

■

5 Reinforcement Learning

In this section, we review several core reinforcement-learning methods using the terminology developed within this framework. Our goal is not to survey the extensive literature, but rather to emphasize the complexity inherent in player design. In the following subsection, we consider the framework of Section 3 in the single-player setting to illustrate a learning method that does not primarily rely on Bellman-type updates. In the following subsection, we review multi-agent reinforcement learning methods.

Let us first review some main categories before detailing them further.

(i) Value-based design: The player $(\mathcal{O}, \mathfrak{L}_\varphi, \Upsilon)$ is defined so that the learning algorithms take the form $\mathfrak{L}_\varphi : \mathcal{E} \times \mathcal{M}_\varphi \times \mathcal{M}_\Upsilon \to \mathcal{M}_\varphi$, and the behavior is given by $\Upsilon : \mathcal{M}_\varphi \to \mathcal{M}_\Upsilon$. A simple example is $\mathcal{M}_\varphi := \{\mathbb{S} \times \mathbb{A} \to \mathbb{R}\}$ and $\mathcal{M}_\Upsilon := \{\mathbb{S} \to \mathbb{A}\}$, where Υ yields the maximizing argument over $\mathbb{A}$.

(ii) Policy-based design: The player $(\mathcal{O}, \Upsilon)$ does not rely on an estimate but instead trains the behavior directly. Here, the behavior takes the form $\Upsilon : \mathcal{E} \times \mathcal{M}_\Upsilon \to \mathcal{M}_\Upsilon$.

(iii) Actor-critic design: The player $(\mathcal{O}, \mathfrak{L}_\varphi, \Upsilon)$ is defined so that the behavior takes the form $\Upsilon : \mathcal{E} \times \mathcal{M}_\varphi \times \mathcal{M}_\Upsilon \to \mathcal{M}_\Upsilon$. Here, estimations are trained similarly to value-based designs and are used to enhance the training of the policy-based behavior. The learning algorithms might also, and typically do, take the form $\mathfrak{L}_\varphi : \mathcal{E} \times \mathcal{M}_\varphi \times \mathcal{M}_\Upsilon \to \mathcal{M}_\varphi$ if they aim to learn the value estimate corresponding to the current behavior.

The simplest illustration is vanilla Q-learning. The agent maintains a single estimation space

$$\mathcal{M}_Q := \left\{ Q : \mathbb{S} \times \mathbb{A} \to \mathbb{R} \right\}$$

and, after training, adjusts the behavior to act greedily:

$$\Upsilon : \mathcal{M}_Q \to \mathcal{M}_\Upsilon \qquad \Upsilon(Q) = \arg\max_{a \in \mathbb{A}} Q(\cdot, a) \in \{\mathbb{S} \to \mathbb{A}\}$$

The learning algorithm, $\mathfrak{L}_Q : \mathcal{E} \times \mathcal{M}_\varphi \times \mathcal{M}_\Upsilon \to \mathcal{M}_Q$, implemented, for example, via temporal-difference updates, iteratively adjusts the estimates to assign higher values to favorable outcomes while preserving their time consistency. In this method, enforcing consistency before the value estimates are well trained often leads to instability. To mitigate this, one may introduce a slower-moving "target" network to anchor the estimates while continuing to explore, employ two value networks with decoupled or conservative targets to reduce overestimation bias or apply additional regularization terms to stabilize both the estimation and behavioral updates. See, for example, [32, 33, 34, 35, 36] for foundational and modern off-policy implementations.

In on-policy learning, the estimates aim to capture the value of the current behavior. This mitigates instabilities that arise from enforcing consistency early in training, but at the expense of making past observations less useful because the behavior continually evolves. See, for example, [37, 38, 39]. For training such estimates, which we do not elaborate on here, see also [40, 41] for important methodological considerations.

Distributional RL addresses the setting in which estimates represent distributions rather than expectations, with estimates taking the form $\mathcal{M}_\varphi : \{\mathbb{S} \times \mathbb{A} \to \mathcal{P}(\mathbb{R})\}$. See, for example, [26, 42, 43, 44]. In the next subsection, we present a toy example in which the estimates are random variables, $\mathcal{M}_\varphi : \{\hat{\Omega} \times \mathbb{S} \to \mathbb{R}\}$, similar to the concept of ensemble methods such as [45]

Model-based RL incorporates further spaces of estimations for the upcoming observations. As we briefly mentioned in Section 2, such estimates may include future states, rewards, and actions. Furthermore, one might first consider embeddings of the observations to facilitate predicting future embeddings. See, for example, [46, 47, 48] for some modern implementations.

Once players have an estimate for future observations, they can plan for the future in a time-inconsistent manner. Similar to the consideration in Section 3, suppose an estimated policy takes values in the space of controls rather than actions. In such cases, the plan devised for a potential future observation may differ from what is actually planned when that situation is realized. This is expected, as the player tailors its efforts to the current situation. This is an important generalization, which is typically absent in model-free approaches, and suggests that time-consistency is an important but not fundamental property. We will mention three approaches in this direction:

• Model Predictive Control and its modern applications in RL can be seen as a direct example. See, for instance, [49, 50].

• Monte Carlo Tree Search methods also plan into the future, using policy solely to better evaluate the current possible actions; see, for example, [51, 52, 53].

• Hierarchical methods include additional estimations that assign goals to the be-

havior. This enables the generation of diverse strategies and, in the case of model-free implementations, introduces a form of time inconsistency. However, rather than discarding the future plan, the player commits to the assigned goal. This provides the flexibility to behave differently later, depending on the previously assigned goal; see, for example, [54, 55, 56].

Next, we briefly mention methods for learning representations from observations. The main objective is to reduce the complexity of the observations while retaining a representation that is sufficiently rich for the task at hand. Broadly speaking, one may consider predictive or contrastive objectives. For predictive objectives, see for example, [46, 48, 57]. Contrastive objectives aim to leverage invariance and discriminability, grouping "similar" observations together while separating "dissimilar" ones. Examples include [58, 59, 60].

Not only learning observations is important, but exploring the diversity of those observations is also crucial. A widely adopted strategy is to provide intrinsic rewards. For example, a player may favor observations where internal estimations are failing to accurately predict their targets. See, for example, [61, 62, 63]. Another approach is to promote unfamiliar states to the player, such as, [64, 65, 66].

To conclude this brief overview, we point out that even when there is only a single decision-maker, setting up an environment through a Markov decision process (or one of its variants) does not capture the full picture. Optimization is, of course, crucial for formalizing which observations are preferred, and it provides strong guidance for designing the players, as previously discussed. However, when many players interact, seeking compatible strategies without accounting for their internal estimations does not necessarily provide clear guidance on which observations are going to be favored by the players. We remark that, in Definition 7 of uncertain equilibrium, each player optimizes their own objective.

5.1 A learning algorithm for CartPole

We first briefly illustrate a method for learning CartPole. The goal is not to propose a better learning method, but a novel one to draw an attention to potential variety of learning algorithms for forming estimations. We learn $\hat{\phi}$ with fixed horizon $T = 8$. It is a value-based method with model given. We trained value networks without Bellman-type updates and instead promote such time-consistency afterwards.

Let us revisit the basics of the CartPole problem. The state space is $\mathbb{S} = \mathbb{R}^4$ representing position, velocity, angle, and angular velocity. The action space is $\mathbb{A} = \{0, 1\}$, where the actions represent applying force to the left or right of the cart. The goal is to keep the pole attached to the cart in the upright position.

The player has a memory, a function of observations, for recording observations, including states, actions, and episode-level evaluations. After each episode,

a performance metric is computed as the average of the agent's relative and absolute performance, and assigned as the evaluation of the episode. Here, the relative performance is computed from the players moving average of how many steps pole was upright. The absolute performance is computed depending on the maximum potential steps, which is 500.

In this problem, $\hat{p}$ is deterministic and can be learned. However, as it is not of our interest, we took it as given. We also set $\hat{F} = 0$ to only model the state values $\hat{\phi}$. Introduce $\{\mathcal{N}_\phi^k\}_{k=1}^{K_\phi}$ for some $K_\phi \in \mathbb{N}$ as neural networks $\mathbb{S} \to \mathbb{R}_+$ for state values. Let $\mathfrak{L}_\phi : \mathcal{E} \times \mathcal{M}_\phi \to (\hat{\Omega} \times \mathbb{S} \to \mathbb{R}_+)$ and

$$\mathfrak{L}_\phi({}^n\mathcal{O}, {}^{n-1}\hat{\phi}) := \sum_{k=1}^{K_\phi} \mathcal{N}_\phi^k \, \mathbf{1}_{\{k\}}(\hat{\omega}), \quad \hat{\omega} \in \hat{\Omega} := \{1, \dots, K_\phi\}, \text{ and } \hat{\mathbb{P}} \text{ uniform.}$$

Here, ${}^n\hat{\phi} := \mathfrak{L}_\phi({}^n\mathcal{O}, {}^{n-1}\hat{\phi})$ is representing abstractly the whole training process. $\hat{\mathbb{P}}$ is taken as uniform means that the player has no information about which network is providing better estimations, which is again for simplicity only. We took the range of each $\mathcal{N}_\phi^k$ as $[0, 100]$, and trained these networks relying on the memory. After 6 episodes, the player goes over the memory to recall best and worst performances, and forms clusters of states to assign higher or lower values to them. We note that, values are changing as the performance increase over time. Also, the training is not done by assigning expected values, but instead directly assigns higher or lower values. Then, we separately promoted time-consistency of these assignments.

To ensure that the value networks are properly trained, we consider the set of all controls $\mathbb{A}^T = \{0, 1\}^T$ and index them by $\{\alpha^k\}_{k=1}^{2^T}$.[8] Then, we let $\mathfrak{L}_\pi : \mathcal{M}_\phi \to (\tilde{\Omega} \times \hat{\Omega} \times \mathbb{S} \to \mathbb{A}^T)$ as

$$\mathfrak{L}_\pi := \sum_{k=1}^{K_\phi} \sum_{\ell=1}^{2^T} \delta_{\alpha^\ell} \mathbf{1}_{\{l\}}(\tilde{\omega}) \mathbf{1}_{\{k\}}(\hat{\omega}) \quad \text{where, } \tilde{\omega} \in \tilde{\Omega} := \{1, \dots, 2^T\},$$

$$\hat{\mathbb{P}}(\tilde{\omega}, \hat{\omega}) := \frac{1}{Z} \exp\left(J(\hat{\omega}, x, \alpha^{\tilde{\omega}})\right), \text{ and } J(\cdot, x; \alpha) := \mathbb{E}^{\mathbb{P}^{x,\alpha}}\left[\hat{\phi}(\cdot, X_T)\right]$$

Here, we extended the probability space by adding $\tilde{\Omega}$ to separately keep track of randomness coming from value networks, and the policy that they induce. $\hat{\mathbb{P}}$ is also now viewed as a probability over $\tilde{\Omega} \times \hat{\Omega}$, and Z is the normalizing constant.

[8] We note that it is considerably easier learning controls in this simple setting, since $\mathbb{A}^T$ is finite. We first included a separate policy learning to generate a small amount of controls, and aimed to use value networks to select from them, but the player performs quite well even without properly trained value networks. Hence, we omitted from the discussion and evaluate all of the potential controls to make sure value is well-trained.

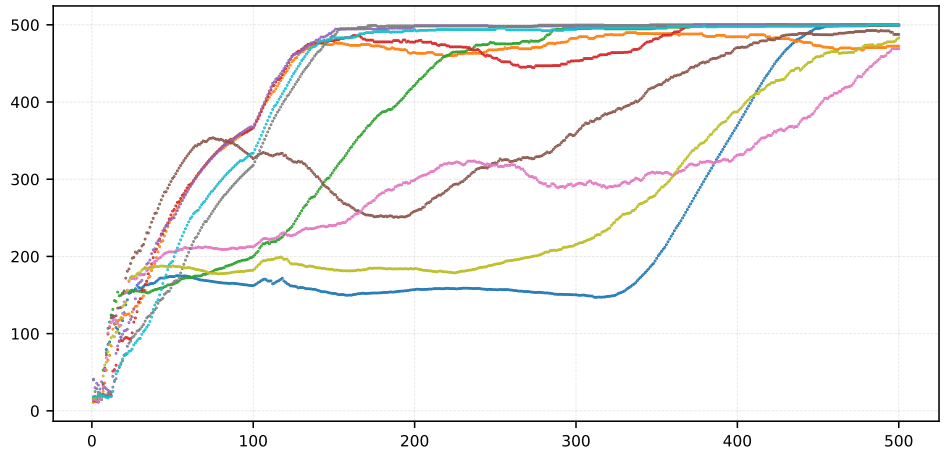


Figure 2: Performance of the CartPole Game Across 10 Selected Runs. The x-axis represents the number of games (or episodes), while the y-axis shows the total reward for each episode. Each line corresponds to one of the best 10 runs out of 32, with lines showing the moving average calculated from up to the last 100 episodes.

Lastly, given $(\hat{\phi}, \hat{\pi})$ as $(\mathfrak{L}_\phi, \mathfrak{L}_\pi)({}^n\mathcal{O})$, the behavior $\Upsilon : \mathcal{M}_\phi \times \mathcal{M}_\pi \to (\tilde{\Omega} \times \hat{\Omega} \times \mathbb{S} \to \mathbb{A})$ is taken as

$$\Upsilon(\hat{\pi})(\hat{\omega}, x) := \sum_{k=1}^{K_\phi} \sum_{\ell=1}^{2^T} \delta_{\alpha^\ell(x)} \mathbf{1}_{\{l\}}(\tilde{\omega}) \mathbf{1}_{\{k\}}(\hat{\omega})$$

In fact, the behavior commits for T steps, but that would need to introduce the parameter t and we omitted it for brevity. See Figure 2 for the performance, and refer to the repository [67] for implementation details. Automatically generated reports, produced using LLMs, are also provided to enhance the accessibility of the implementation.

5.2 Multi-agent Reinforcement Learning

Our aim in this section is to provide a broad view of the multi-agent reinforcement learning literature through the lens of the language developed in this work. Once multiple players are present, relevant considerations expand dramatically. Agents may model opponents, communicate, teach, manipulate, or form temporary alliances, and it is difficult to place principled bounds on such possibilities.

In particular, if a collection of agents is capable of developing a language to communicate observations or situations they have encountered, this opens the door to higher-order interactions: agents may revisit past experiences, reassess estimations, propose alternative plans, negotiate over future actions, implement voting mechanisms, or assign specialized roles within the group. While such behaviors are routine for us, they illustrate how rich multi-agent learning can become once agents are viewed as structured decision-makers rather than as static policies.

We begin by mentioning a major line of work in multi-agent learning, which is motivated by convergence to equilibrium via regret minimization. Initiated by [68] in games with incomplete information, these methods iteratively update policies so as to minimize regret, thereby converging to unexploitable strategies. This approach has led to notable success in heads-up limit Hold'em poker [69]. Subsequent work has focused on scaling beyond tabular representations, including neural network approximations [70] and their integration with search-based planning [71]. This line of research is fundamentally equilibrium-oriented. From the perspective of a central designer seeking stable strategies to announce, such methods play a crucial role. However, as our focus is on player design rather than equilibrium computation, we do not pursue this direction further. We refer the interested reader to earlier equilibrium-motivated approaches based on Q-learning [72], and to [73] for an analysis of convergence dynamics in differentiable games.

To extend value-based designs to multi-agent settings, one direct approach is to treat each player as unaware of the others and to regard the remaining players as an unknown component of the environment. In this case, each player $(\mathcal{O}^i, \mathfrak{L}^i_Q, \Upsilon^i)$ still relies on a single type of estimate, such as $\mathcal{M}^i_Q := \{\mathbb{S} \times \mathbb{A}^i \to \mathbb{R}\}$, together with a common behavior Υ^i. However, unlike the single-agent setting, the optimization target of the learning algorithms is no longer fixed by the environment, since the behaviors of other players are themselves evolving.[9]

Without allowing players to explicitly acquire additional estimates about their opponents, one can still couple them through a shared estimate. A large body of the literature studies this centralized-training–decentralized-execution regime, primarily in cooperative settings. Since the learning algorithm is centralized, the training phase is more naturally viewed as a single player $(\vec{\mathcal{O}}, \mathfrak{L}_Q, \vec{\Upsilon})$ with high-dimensional observations and behaviors.

Consider, for example, a shared estimate space $\mathcal{M}_Q := \{\mathbb{S} \times \vec{\mathbb{A}} \to \mathbb{R}\}$ with learning algorithm $\mathfrak{L}_Q : \vec{\mathcal{E}} \times \vec{\mathcal{M}}_Q \times \mathcal{M}_Q \to \vec{\mathcal{M}}_Q \times \mathcal{M}_Q$. As long as Υ^i de-

[9] For simplicity, we write estimates as functions of the state $\mathbb{S}$. In practice, observations are often processed through additional estimates, such as recurrent neural networks, and the output space of those estimates is used instead. We omit such considerations to focus on what is new in the multi-agent case.

pends only on $\mathcal{M}_Q^i$, once training is complete, one can define individual players as $(\mathcal{O}^i, \mathfrak{L}^i, \Upsilon^i)$, where $\mathfrak{L}^i$ yields Q^i as a constant in order to freeze training.

There are several ways to realize such a single-player formulation. For instance, in [74], the authors define Q as the sum of the individual Q^i's. In [75], this approach is generalized by introducing an additional estimate $\varphi : \vec{\mathcal{M}}_Q \to \mathcal{M}_Q$ satisfying $\partial_{Q^i} \varphi \geq 0$, ensuring that the learned estimates Q^i respect each player's local maximization behavior. In [76], the authors further generalize this framework by relaxing the monotonicity constraint and introducing consistency conditions that allow a broader class of decompositions while preserving individual players' behavior. Finally, in [77], the analysis is carried out in the advantage space by introducing two estimates V and A to define Q, while maintaining a general representation.

The same centralization idea can also be applied to actor–critic designs. In this case, the planned behavior ${}^n\vec{\Upsilon} \in \vec{\mathcal{M}}_\Upsilon$ of the central (single) agent is fixed after training, and the individual agents are equipped with mappings Υ^i that constantly yield ${}^n\Upsilon^i \in \mathcal{M}_\Upsilon^i$.

In [78], the authors apply a deterministic policy gradient method with a central player $(\vec{\mathcal{O}}, \vec{\mathfrak{L}}_Q, \vec{\Upsilon})$, where

$$\begin{aligned} \mathfrak{L}_Q^i &: \vec{\mathcal{E}} \times \vec{\mathcal{M}}_Q \times \vec{\Upsilon} \to \mathcal{M}_Q^i, \\ \Upsilon^i &: \mathcal{E}^i \times \mathcal{M}_Q^i \times \mathcal{M}_\Upsilon^i \to \mathcal{M}_\Upsilon^i, \end{aligned}$$

and $\mathcal{M}_\Upsilon^i = \{\mathbb{S}^i \to \mathbb{A}^i\}$, $\mathcal{M}_Q^i = \{\vec{\mathbb{S}} \times \vec{\mathbb{A}} \to \mathbb{R}\}$. In [79], the authors consider stochastic policy gradients, and hence $\mathcal{M}_\Upsilon^i = \{\Omega^u \times \mathbb{S}^i \to \mathbb{A}^i\}$ when written as a random variable. They introduce an advantage estimate that compares the current Q estimate to an average taken over the planned behavior ${}^n\Upsilon^i$, in order to provide individual credit assignment. In [80], the authors demonstrate the effectiveness of proximal policy optimization in this regime. This core method has also been successfully used in large-scale systems such as OpenAI Five for *Dota 2* [81].

In this centralized regime, communication has also been studied extensively in the literature. In this context, communication often refers to the introduction of two additional estimates. First, each agent constructs messages from its individual observations, taking values in a Euclidean space, $C \in \mathcal{M}_C^i := \{\mathcal{E}^i \to \mathbb{R}^k\}$. Second, each player maintains an estimate that aggregates the incoming messages, $D \in \mathcal{M}_D^i := \{\vec{\mathbb{R}}^k \to \mathbb{R}^{k'}\}$. Within this framework, the learning algorithm of the central player can, for example, be formulated as

$$\mathfrak{L}_Q : \vec{\mathcal{E}} \times \vec{\mathcal{M}}_C \times \vec{\mathcal{M}}_D \times \vec{\mathcal{M}}_Q \times \mathcal{M}_Q \to \vec{\mathcal{M}}_C \times \vec{\mathcal{M}}_D \times \vec{\mathcal{M}}_Q \times \mathcal{M}_Q$$

allowing the estimates C and D to be updated jointly with the value estimates, for instance via backpropagation. The value functions may then incorporate the aggregated messages in their domain, $\mathcal{M}_Q^i := \{\mathbb{S} \times \mathbb{A}^i \times \mathbb{R}^{k'} \to \mathbb{R}\}$, or, alternatively, the planned behavior itself may be defined to depend explicitly on the aggregated messages.

In [82], the authors implement communication both in independent Q-learning and in a centralized variant, following the general structure described above. In [83], the authors augment messages with a notion of directionality, allowing each agent to attend to incoming messages along a chosen direction. In [84], the authors adopt a centralized single-agent perspective, and introduce multiple communication layers $D^1, D^2, D^3, \ldots$, in which messages are iteratively averaged. This architecture enables agents to communicate at higher levels, even when they are not directly connected. In [85], the authors extend this line of work to settings with individual agents, including competitive scenarios, and introduce a gating mechanism that allows agents to learn when to exchange messages and when to remain silent. In [86], attention mechanisms are used to aggregate messages across multiple communication layers. Finally, in [87], the attention-based framework is further extended by incorporating a gating mechanism through a learnable adjacency matrix, treating the communication graph itself as a differentiable estimate.

A large-scale project worth mentioning in the context of communication is [88], which studies the turn-based game *Diplomacy*, a setting that allows natural-language communication between players. The authors train a large language model on human gameplay data to generate dialogue and introduce explicit estimates of opponents' intents. Policies are then learned in a manner grounded in human play, integrating strategic decision-making with models of communication and inferred opponent intentions. In effect, this approach constructs a richly structured model of a human player. This example further illustrates that such games cannot be adequately captured by standard external formulations.

We remark that the above example already includes elements of opponent modeling. Let us mention a few important works on opponent modeling. Depending on the structure of the game, it may suffice for a player to observe how other agents influence the state of the game. For example, in another large-scale project [89], the authors train independent agents for the game *StarCraft II*, where players compete through carefully designed self-play as well as against opponents constructed to exploit their strategies, with policies guided in part by human gameplay data. However, a more sophisticated player may need to maintain explicit estimates dedicated to its opponents. In [90], the authors introduce an additional estimate that constructs an embedding of the opponent based on observations. First, they extend the Q-function to incorporate this embedding as an additional input. Second, they consider a collection of Q-functions and use the learned embedding to as-

sign weights to them. In these approaches, the learning algorithm for the opponent embedding is initially embedded within the learning procedure for the standard Q-estimate. Finally, the authors propose introducing a separate learning algorithm specifically for the opponent embedding in order to enhance its expressiveness. In [91], rather than constructing a full model of a player, the authors focus on learning a particular estimate for characterizing an observed opponent. They introduce one estimate that aggregates information across observations from multiple episodes of the observed player, and a second estimate that depends on the first while incorporating partial observations within a single episode. Together with the current observation, the outputs of these two estimates are then used as inputs to a third estimate, whose learning objective is to predict properties of the opponent being observed. In [92], the authors introduce intrinsic rewards for influencing the behavior of other agents, and demonstrate that accounting for such influence can promote coordination and communication. They further show that these intrinsic rewards can be trained in a decentralized manner, provided that agents acquire predictive estimates of other agents' action distributions.

We have aimed to emphasize the importance of player design. The literature is extensive and spans many different directions. To illustrate how broad the design space can be, we conclude by pointing to recent works in which players incorporate large language models as part of their estimates. For example, [93] demonstrates the emergence of believable human behavior, while [94, 95] introduce teams of agents designed for collaborative software development.

6 Conclusion

We have reframed the problem of decision-making from the perspective of the player and, in essence, abstracted the constructions in reinforcement learning. Traditional approaches adopt a viewpoint external to the players. Even when there is a single decision-maker, the external setting may guide the design but does not suffice to capture the required complexity. When many players interact, considering only compatible strategies is not a sufficiently rich objective, particularly for competing players.

Our broader vision is to emphasize the inherent complexity of intelligent behavior. We do not act through a single estimate, but through dynamically evolving layers of estimations and behaviors: structures capable of working with any stream of observations, adapting to novel situations, reconstructing, planning, and predicting; forming diverse values both individually and collectively. Understanding how a dynamical system achieves such complexity remains far beyond our reach, but this work aims to provide the foundational definitions for initiating a formal approach.

References

[1] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton, NJ: Princeton University Press, 1944.

[2] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.

[3] M. Maschler, E. Solan, and S. Zamir, *Game Theory*. Cambridge: Cambridge University Press, 2013.

[4] R. J. Aumann, "Correlated equilibrium as an expression of bayesian rationality," *Econometrica*, vol. 55, no. 1, pp. 1–18, 1987.

[5] J. B. Kadane and P. D. Larkey, "Subjective probability and the theory of games," *Management Science*, vol. 28, no. 2, pp. 113–120, 1982.

[6] Y. Shoham, R. Powers, and T. Grenager, "Multi-agent reinforcement learning: a critical survey," technical report, Computer Science Department, Stanford University, 2003. Unpublished manuscript.

[7] Y. Shoham, R. Powers, and T. Grenager, "If multi-agent learning is the answer, what is the question?," *Artificial Intelligence*, vol. 171, no. 7, pp. 365–377, 2007.

[8] J. R. Wright and K. Leyton-Brown, "Predicting human behavior in unrepeated, simultaneous-move games," *Games and Economic Behavior*, vol. 106, pp. 16–37, 2017.

[9] J. S. Hartford, J. R. Wright, and K. Leyton-Brown, "Deep learning for predicting human strategic behavior," in *Advances in Neural Information Processing Systems*, vol. 29, (Red Hook, NY, USA), pp. 2424–2432, Curran Associates, Inc., 2016.

[10] G. W. Brown, "Iterative solutions of games by fictitious play," in *Activity Analysis of Production and Allocation* (T. C. Koopmans, ed.), pp. 374–376, New York: Wiley, 1951.

[11] C. Daskalakis, R. Frongillo, C. H. Papadimitriou, G. Pierrakos, and G. Valiant, "On learning algorithms for nash equilibria," in *Algorithmic Game Theory* (S. Kontogiannis, E. Koutsoupias, and P. G. Spirakis, eds.), pp. 114–125, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[12] S. Hart and A. Mas-Colell, "A simple adaptive procedure leading to correlated equilibrium," *Econometrica*, vol. 68, no. 5, pp. 1127–1150, 2000.

[13] D. Fudenberg and D. K. Levine, *The Theory of Learning in Games*, vol. 2 of *Economic Learning and Social Evolution*. Cambridge, MA: MIT Press, 1998.

[14] J.-M. Lasry and P.-L. Lions, "Mean field games," *Japanese Journal of Mathematics*, vol. 2, no. 1, pp. 229–260, 2007.

[15] M. Huang, R. P. Malhamé, and P. E. Caines, "Large-population stochastic dynamic games: Closed-loop mckean–vlasov systems and the nash certainty equivalence principle," *Communications in Information and Systems*, vol. 6, no. 3, pp. 221–252, 2006.

[16] R. Carmona and F. Delarue, *Probabilistic Theory of Mean Field Games with Applications I: Mean Field FBSDEs, Control, and Games*, vol. 83 of *Probability Theory and Stochastic Modelling*. Cham: Springer, 2018.

[17] R. Carmona and F. Delarue, *Probabilistic Theory of Mean Field Games with Applications II: Mean Field Games with Common Noise and Master Equations*, vol. 84 of *Probability Theory and Stochastic Modelling*. Cham: Springer, 2018.

[18] L. L. Thurstone, "A law of comparative judgment," *Psychological Review*, vol. 34, no. 4, pp. 273–286, 1927.

[19] R. D. Luce, *Individual Choice Behavior: A Theoretical Analysis*. New York: John Wiley and Sons, 1959.

[20] H. D. Block, "Random orderings and stochastic theories of responses (1960)," in *Economic Information, Decision, and Prediction: Selected Essays: Volume I, Part I Economics of Decision*, pp. 172–217, Dordrecht: Springer Netherlands, 1974.

[21] D. McFadden, "Conditional logit analysis of qualitative choice behavior," in *Frontiers in Econometrics* (P. Zarembka, ed.), pp. 105–142, New York: Academic Press, 1974.

[22] K. E. Train, *Discrete Choice Methods with Simulation*. Cambridge: Cambridge University Press, 2 ed., 2009.

[23] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[24] O. Chapelle and L. Li, "An empirical evaluation of Thompson sampling," in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, eds.), vol. 24, (Red Hook, NY, USA), pp. 2249–2257, Curran Associates, Inc., 2011.

[25] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multi-armed bandit problem," *Machine Learning*, vol. 47, no. 2–3, pp. 235–256, 2002.

[26] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, vol. 70 of *Proceedings of Machine Learning Research*, (Sydney, NSW, Australia), pp. 449–458, PMLR, 2017.

[27] M. İşeri, "Two player game." GitHub repository, 2025. Available at: https://github.com/melihiseri/TwoPlayerGame.

[28] E. Bayraktar, M. İşeri, and N. Mascarenhas, "Algorithmic collusion of strategic firms." Work in progress, 2025.

[29] E. Calvano, G. Calzolari, V. Denicolò, and S. Pastorello, "Artificial intelligence, algorithmic pricing, and collusion," *American Economic Review*, vol. 110, pp. 3267–3297, October 2020.

[30] M. İşeri and J. Zhang, "Set values for mean field games," *Transactions of the American Mathematical Society*, vol. 377, no. 10, pp. 7117–7174, 2024.

[31] P. Cardaliaguet and S. Hadikhanloo, "Learning in mean field games: The fictitious play," *ESAIM: Control, Optimisation and Calculus of Variations*, vol. 23, no. 2, pp. 569–591, 2017.

[32] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] N. Vieillard, O. Pietquin, and M. Geist, "Munchausen reinforcement learning," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, (Red Hook, NY, USA), pp. 4235–4246, Curran Associates, Inc., 2020.

[34] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*, (Phoenix, Arizona), pp. 2094–2100, AAAI Press, 2016.

[35] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, vol. 80, (Stockholm, Sweden), pp. 1587–1596, PMLR, 2018.

[36] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, vol. 80, (Stockholm, Sweden), pp. 1861–1870, PMLR, 2018.

[37] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992.

[38] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems* (S. A. Solla, T. K. Leen, and K.-R. Müller, eds.), vol. 12, (Cambridge, MA), pp. 1057–1063, MIT Press, 2000.

[39] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, NY, USA), pp. 1928–1937, PMLR, 2016.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[41] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations (ICLR)*, 2016.

[42] W. Dabney, M. Rowland, M. G. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, (New Orleans, Louisiana, USA), pp. 2892–2901, AAAI Press, 2018.

[43] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm, Sweden), pp. 1096–1105, PMLR, 2018.

[44] D. Yang, L. Zhao, Z. Lin, T. Qin, J. Bian, and T.-Y. Liu, "Fully parameterized quantile function for distributional reinforcement learning," in *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, (Red Hook, NY, USA), pp. 556–566, Curran Associates, Inc., 2019.

[45] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in Neural Information Processing Systems*, vol. 29, (Red Hook, NY, USA), pp. 4026–4034, Curran Associates, Inc., 2016.

[46] D. Ha and J. Schmidhuber, "World models," *arXiv preprint arXiv:1803.10122*, 2018.

[47] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, (Long Beach, California, USA), pp. 2555–2565, PMLR, 2019.

[48] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, 2020.

[49] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, (Brisbane, Australia), pp. 7559–7566, IEEE, 2018.

[50] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems*, vol. 31, (Red Hook, NY, USA), pp. 4754–4765, Curran Associates, Inc., 2018.

[51] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman,

D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[52] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[53] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[54] P. Bacon, J. Harb, and D. Precup, "The option-critic architecture," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, (San Francisco, California, USA), pp. 1726–1734, AAAI Press, 2017.

[55] O. Nachum, S. Gu, H. Lee, and S. Levine, "Data-efficient hierarchical reinforcement learning," in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, (Montréal, Canada), pp. 3307–3317, Curran Associates, Inc., 2018.

[56] A. Levy, G. Konidaris, and R. Platt, "Learning multi-level hierarchies with hindsight," in *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, (Long Beach, California, USA), pp. 3846–3855, PMLR, 2019.

[57] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman, "Data-efficient reinforcement learning with self-predictive representations," in *Proceedings of the 9th International Conference on Learning Representations (ICLR 2021)*, 2021.

[58] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[59] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)* (H. D. III and

A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, (Virtual Event), pp. 5639–5650, PMLR, 2020.

[60] A. Stooke, K. Lee, M. Laskin, and P. Abbeel, "Decoupling representation learning from reinforcement learning," in *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, (Virtual Event), pp. 9870–9879, PMLR, 2021.

[61] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (Sydney, NSW, Australia), pp. 2778–2787, PMLR, 2017.

[62] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*, 2019.

[63] Z. D. Guo, S. Thakoor, M. Pîslar, B. A. Pires, F. Altché, C. Tallec, A. Saade, D. Calandriello, J.-B. Grill, Y. Tang, M. Valko, R. Munos, M. G. Azar, and B. Piot, "BYOL-explore: Exploration by bootstrapped prediction," in *Advances in Neural Information Processing Systems*, vol. 35, (Red Hook, NY, USA), pp. 36965–36978, Curran Associates, Inc., 2022.

[64] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell, "Never give up: Learning directed exploration strategies," in *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*, 2020.

[65] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell, "Agent57: Outperforming the atari human benchmark," in *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, (Virtual Event), pp. 507–517, PMLR, 2020.

[66] A. Saade, S. Kapturowski, D. Calandriello, C. Blundell, P. Sprechmann, L. Sarra, O. Groth, M. Valko, and B. Piot, "Unlocking the power of representations in long-term novelty-based exploration (recode)," *arXiv preprint arXiv:2305.01521*, 2023.

[67] M. İşeri, "Cartpole toy model." GitHub repository, 2025. Available at: https://github.com/melihiseri/CartPole_ToyModel.

[68] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Advances in Neural Information Processing Systems*, vol. 20, (Red Hook, NY, USA), pp. 1729–1736, Curran Associates, Inc., 2007.

[69] M. Bowling, N. Burch, M. Johanson, and O. Tammelin, "Heads-up limit hold'em poker is solved," *Science*, vol. 347, no. 6218, pp. 145–149, 2015.

[70] N. Brown, A. Lerer, S. Gross, and T. Sandholm, "Deep counterfactual regret minimization," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, vol. 97, (Long Beach, CA, USA), pp. 793–802, PMLR, 2019.

[71] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong, "Combining deep reinforcement learning and search for imperfect-information games," in *Advances in Neural Information Processing Systems*, vol. 33, (Red Hook, NY, USA), pp. 13638–13650, Curran Associates, Inc., 2020.

[72] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, vol. 4, no. Nov, pp. 1039–1069, 2003.

[73] D. Balduzzi, S. Racanière, J. Martens, J. Foerster, K. Tuyls, and T. Graepel, "The mechanics of n-player differentiable games," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, vol. 80, (Stockholm, Sweden), pp. 354–363, PMLR, 2018.

[74] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2085–2087, 2018.

[75] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, vol. 80, (Stockholm, Sweden), pp. 4295–4304, PMLR, 2018.

[76] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proceedings of the 36th International Conference on Machine*

Learning (ICML), vol. 97, (Long Beach, CA, USA), pp. 5887–5896, PMLR, 2019.

[77] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "Qplex: Duplex dueling multi-agent q-learning," in *International Conference on Learning Representations (ICLR)*, 2021.

[78] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[79] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, pp. 2974–2982, 2018.

[80] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," in *Advances in Neural Information Processing Systems*, vol. 35, pp. 24611–24624, 2022.

[81] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," 2019.

[82] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 29, pp. 2137–2145, 2016.

[83] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," in *Proceedings of the 36th International Conference on Machine Learning*, pp. 1538–1546, 2019.

[84] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems*, vol. 29, (Red Hook, NY, USA), pp. 2244–2252, Curran Associates, Inc., 2016.

[85] A. Singh, T. Jain, and S. Sukhbaatar, "Learning when to communicate at scale in multiagent cooperative and competitive tasks," in *International Conference on Learning Representations (ICLR)*, 2019.

[86] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *International Conference on Learning Representations*, 2020.

[87] J. Ryu, H. Zhou, J. Park, and A. Iosifidis, "Multi-agent graph-attention communication and teaming," in *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 964–973, 2021.

[88] Meta Fundamental AI Research Diplomacy Team (FAIR), A. Bakhtin, N. Brown, E. Dinan, G. Farina, C. Flaherty, D. Fried, A. Goff, J. Gray, H. Hu, A. P. Jacob, M. Komeili, K. Konath, M. Kwon, A. Lerer, M. Lewis, A. H. Miller, S. Mitts, A. Renduchintala, S. Roller, D. Rowe, W. Shi, J. Spisak, A. Wei, D. Wu, M. Yates, H. Zhang, M. Zijlstra, M. Letychevsky, *et al.*, "Human-level play in the game of diplomacy by combining language models with strategic reasoning," *Science*, vol. 378, no. 6624, pp. 1067–1074, 2022.

[89] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[90] H. He, J. Boyd-Graber, K. Kwok, and H. Daumé, III, "Opponent modeling in deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning* (M. F. Balcan and K. Q. Weinberger, eds.), vol. 48 of *Proceedings of Machine Learning Research*, (New York, New York, USA), pp. 1804–1813, PMLR, 20–22 Jun 2016.

[91] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine theory of mind," in *Proceedings of the 35th International Conference on Machine Learning (ICML)* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholm, Sweden), pp. 4218–4227, PMLR, 10–15 Jul 2018.

[92] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas, "Social influence as intrinsic motivation for multi-agent deep reinforcement learning," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, vol. 97, (Long Beach, CA, USA), pp. 3040–3049, PMLR, 2019.

[93] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Pro-*

ceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, (New York, NY, USA), pp. 1–22, Association for Computing Machinery, 2023.

[94] C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu, and M. Sun, "ChatDev: Communicative agents for software development," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Bangkok, Thailand), pp. 15174–15186, Association for Computational Linguistics, Aug. 2024.

[95] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, and J. Schmidhuber, "MetaGPT: Meta programming for a multi-agent collaborative framework," in *The Twelfth International Conference on Learning Representations*, 2024.