games

MDPI

Article

Learning Optimal Strategies in a Duel Game

Angelos Gkekas, Athina Apostolidou, Artemis Vernadou and Athanasios Kehagias *

Department of Electronics and Computer Engineering, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece; gekasaa@ece.auth.gr (A.G.); auhapo@gmail.com (A.A.); artber@hotmail.com (A.V.)
* Correspondence: kehagiat@ece.auth.gr

Abstract: We study a duel game in which each player has incomplete knowledge of the game parameters. We present a simple, heuristically motivated and easily implemented algorithm by which, in the course of repeated plays, each player estimates the missing parameters and consequently learns his optimal strategy.

Keywords: duel game; tree game; backward induction; incomplete information; learning

1. Introduction

We study a two-player zero-sum duel game in which certain game parameters are unknown, and we present an algorithm by which each player can estimate the opponent's parameters and consequently learn his optimal strategy.

The duel game with which we are concerned has been presented in Polak (2007), and a similar one was examined in Prisner (2014). The game is a variation of a duel game and, more generally, games of timing, as presented in Fox and Kimeldorf (1969); Garnaev (2000); Radzik (1988).

We call the two players P_1 and P_2; for $n \in \{1,2\}$, P_n has a *kill probability* vector $\mathbf{p}_n$, with the component $p_{n,d}$ (for $d \in \mathbb{N}$) giving the probability that P_n kills his opponent when he shoots at him from distance d. In our analysis, we make the following assumptions:

1. The kill probabilities are given by a function, f:
$$\forall n, d : p_{n,d} = f(d, \theta_n),$$
where θ_n is a parameter vector.
2. Each P_n knows the general form $f(d,\theta)$ and his own parameter vector θ_n, but not that of his opponent.
3. The duel will be repeated a large number of times.

Each P_n's goal is to compute a strategy that optimizes his payoff, which (as will be seen in Section 2), depends on both $\mathbf{p}_1$ and $\mathbf{p}_2$; since P_n does not know his opponents kill probability, he does not know either his own or his opponent's payoff. In short, we have an *incomplete information* game.

Games of incomplete information are those in which each player has only partial information about the payoff parameters. Clearly, the players must utilize some form of *learning*, namely the use of information (i.e., data) collected from previous plays of a game to adjust their strategies through belief update, imitation, reinforcement etc.

The first work on learning in games was, arguably, the introduction of the fictitious play algorithm in Brown (1949); Robinson (1951). In later years, several approaches to learning were proposed in the literature.

check for updates

Academic Editor: Heinrich H. Nax

Received: 9 November 2024
Revised: 17 December 2024
Accepted: 20 January 2025
Published: 5 February 2025

Citation: Gkekas, A., Apostolidou, A., Vernadou, A., & Kehagias, A. (2025). Learning Optimal Strategies in a Duel Game. *Games*, *16*(1), 8. https://doi.org/10.3390/g16010008

The *Bayesian game* approach introduced by Harsanyi (1962) has led to intensive research on the subject of *Bayesian games*; for a recent review, see Zamir (2020). However, the main corpus of this literature is concerned with Bayesian reasoning for a *single* play of a game, rather than learning from *repeated* plays, and this point of view is not particularly relevant to our approach.

Another approach, which can be understood as a form of "intergenerational" learning, is that of *evolutionary game theory*, as presented in Alexander (2023); Tanimoto (2015); Weibull (1997), but this is not directly concerned with games of incomplete information and hence is also not very relevant to our approach. However, for some interesting remarks on the connection between learning and evolutionary game theory, see the book by Cressman (2003).

The main approach to learning in games is the one exemplified by Fudenberg and Levine (1998)'s seminal book *The Theory Of Learning In Games*. This has spawned a tremendous amount of research; a recent overview of the field appears in Fudenberg and Levine (2016).

Fudenberg and Levine's approach is directly relevant to learning in the duel game, but we believe it is *too* general. In particular, we have not been able to find any works which address the specific case in which the functional form of payoff functions is known to the players and it is only required to estimate parameter values.

This is essentially a problem of *parameter estimation*, and there is another research corpus in which this problem is addressed explicitly: that of *machine learning*. Particularly relevant to our needs is the literature on *multi-agent reinforcement learning*, such as Alonso et al. (2001); Gronauer and Diepold (2022). Reinforcement learning is a basic component of machine learning and, in the machine learning community, the connection between multi-agent problems and game theory has long been recognized (Nowe et al., 2012; Rezek et al., 2008); recent and very extensive reviews can be found in Jain et al. (2024); Yang and Wang (2020).

In particular, the issue of convergence of parameter estimation has been comprehensively studied (Hussain et al., 2023; Leonardos et al., 2021; Martin & Sandholm, 2021; Mertikopoulos & Sandholm, 2016). An important conclusion of these and related works is the trade-off between *exploration* and *exploitation*. Quoting from Hussain et al. (2023): "... *understanding exploration remains an important endeavour as it allows agents to avoid suboptimal, or potentially unsafe areas of their state space ... In addition, it is empirically known that the choice of exploration rate impacts the expected total reward*".

In short, extensive literature on learning and incomplete information games is available but, while some ingenious and highly technical approaches have been proposed, we believe that much simpler techniques suffice for our duel learning problem. This is mainly due to the fact that each player knows the general form of the kill probabilities and only needs to estimate his opponent's parameter vector. Hence, in the current paper, we propose a *simple, heuristically motivated, and easily implemented* algorithm which allows each player to estimate his optimal strategy, using the information collected from repeated plays of the duel.

The paper is organized as follows. In Section 2 we present the rules of the game. In Section 3 we solve the game under the assumption of complete information. In Section 4 we present our algorithm for solving the game when the players have incomplete information. In Section 5 we evaluate the algorithm by numerical experiments. Finally, in Section 6 we summarize our results and present our conclusions.

2. Game Description

The duel with which we are concerned is a zero-sum game played between players P_1 and P_2, under the following rules.

1. It is played in discrete rounds (time steps) $t \in \{1, 2, \dots\}$.
2. In the first turn, the players are at distance D.
3. P_1 (resp. P_2) plays on odd (resp. even) rounds.
4. On his turn, each player has two choices: (i) he can shoot his opponent or (ii) he can move one step forward, reducing their distance by one.
5. If P_n shoots (with $n \in \{1, 2\}$), he has a *kill probability* $p_n(d)$ of hitting (and killing) his opponent, where d is their current distance. If he misses, the opponent can walk right next to him and shoot him for a certain kill.
6. Each player's payoff is 1 if he kills the opponent and -1 if he is killed. Note that a player who misses his shot has payoff -1, since the opponent will approach him at distance one and shoot at him with kill probability one.

For $n \in \{1, 2\}$, we will denote by $x_n(t)$ the position of P_n at round t. The starting positions are $x_1(0) = 0$ and $x_2(0) = D$, with $D = 2N, N \in \mathbb{N}$. The distance between the players at time t is

$$d = |x_1(t) - x_2(t)|.$$

For $n \in \{1, 2\}$, the kill probability is a *decreasing* function $p_n : \{1, 2, \dots D\} \to [0, 1]$ with $p_n(1) = 1$. It is convenient to describe the kill probabilities as vectors:

$$\mathbf{p}_n = (p_{n,1}, \dots, p_{n,D}) = (p_n(1), \dots, p_n(D)).$$

This duel can be modeled as an *extensive form game* or *tree game*. The game tree is a directed graph $G = (V, E)$ with vertex set

$$V = \{1, 2, \dots, 2D\},$$

where

1. the vertex $v = d \in \{1, 2, \dots, D\}$ corresponds to a game state in which the players are at distance d, and
2. the vertex $v = d + D \in \{D + 1, D + 2, \dots, 2D\}$ is a *terminal* vertex, in which the "active" player has fired at his opponent.

The edges correspond to *state transitions*; it is easy to see that the edge set is

$$E = \{(1, D + 1), (2, 1), (2, D + 2), (3, 2), (3, D + 3), \dots, (D, D - 1), (D, 2D)\}.$$

An example of the game tree, for $D = 6$, appears in Figure 1. The circular (resp. square) vertices are the ones in which P_1 (resp. P_2) is active, and the rhombic vertices are the terminal ones.

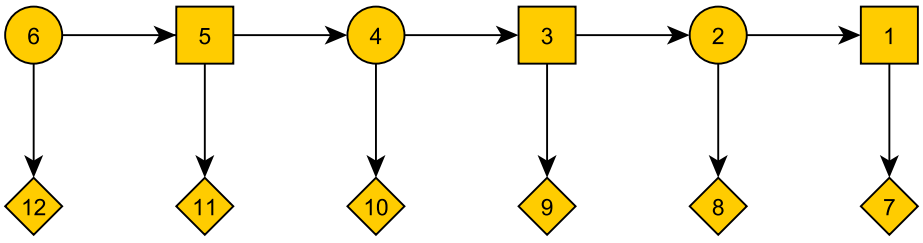

Figure 1. Game tree example.

To complete the description of the game, we will define the *expected* payoff for the terminal vertices. Note that the terminal vertex $d + D$ is the child of the nonterminal vertex d in which:

1. The distance of the players is d and, assuming P_n to be the active player, his probability of hitting his opponent is $p_{n,d}$.
2. The active player is P_1 (resp. P_2) if d is even (resp. odd).

Keeping the above in mind, we see that the payoff (to P_1) of vertex $d + D$ is

$$\forall d \in \{1, \ldots, D\} : Q(d+D) = \begin{cases} p_{1,d} \cdot 1 + (1 - p_{1,d}) \cdot (-1) = 2p_{1,d} - 1, & \text{when } d \text{ is even;} \\ p_{2,d} \cdot (-1) + (1 - p_{2,d}) \cdot 1 = 1 - 2p_{2,d} & \text{when } d \text{ is odd.} \end{cases}$$

Since this is a zero-sum game, the payoff to P_2 at vertex $d + D$ is $-Q(d + D)$. This completes the description of the duel.

3. Solution with Complete Information

It is easy to solve the above duel when each player knows D and *both* $\mathbf{p_1}$ and $\mathbf{p_2}$. We construct the game tree as described in Section 2 and we solve it by backward induction. Since the method is standard, we simply give an example of its application. Suppose that

$$\forall n, d : p_{n,d} = \begin{cases} 1 & \text{when } d = 1, \\ \min\left(1, \frac{c_n}{d^{k_n}}\right) & \text{when } d > 1. \end{cases}$$

We take $c_1 = 1, k_1 = 1, c_2 = 1, k_2 = \frac{1}{2}$. The kill probabilities are as seen in Table 1.

Table 1. Kill probabilities.

d	1	2	3	4	5	6
$p_{1,d}$	1.0000	0.5000	0.3333	0.2500	0.2000	0.1667
$p_{2,d}$	1.0000	0.7071	0.5774	0.5000	0.4472	0.4082

The game tree with terminal payoffs is illustrated in Figure 2.

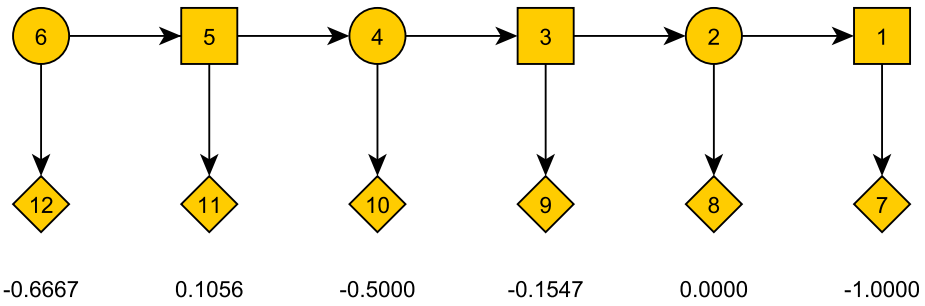

Figure 2. Game tree example with values of terminal vertices.

By the standard backward induction procedure, we select the optimal action at each vertex and also compute the values of the nonterminal vertices. These are indicated in Figure 3 (optimal actions correspond to thick edges). We see that the game value is -0.1547, attained by P_2 shooting when the players are at distance 3 (which happens in round 4).

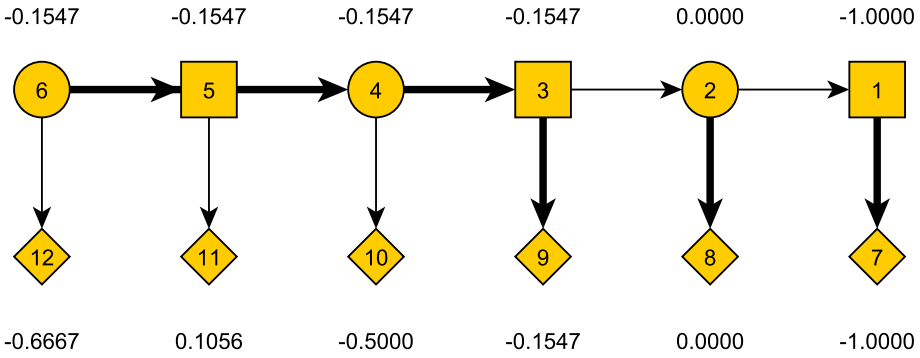

Figure 3. Game tree example with values of all vertices.

We next present a proposition which characterizes each player's optimal strategy in terms of a *shooting threshold*.[1] In the following, we use the standard notation by which "$-n$" denotes the "other player", i.e., $p_{-1} = p_2$ and $p_{-2} = p_1$.[2]

Theorem 1. *We define for* $n \in \{1, 2\}$ *the shooting criterion vectors* $\mathbf{K}_n = (K_{n,1}, \ldots, K_{n,D})$ *where*

$$K_{n,1} = 1 \text{ and for } d \geq 2 : K_{n,d} = p_{n,d} + p_{-n,d-1}.$$

Then, the optimal strategy for P_n *is to shoot as soon as it is his turn and the distance of the players is less than or equal to the shooting threshold* d_n*, where:*

$$d_1 = \max\{d : K_{1,d} \geq 1\}, \quad d_2 = \max\{d : K_{2,d} \geq 1\}.$$

Proof. Suppose the players are at distance d and the active player is P_n.

1. If P_{-n} will not shoot in the next round, when their distance will be $d-1$, then P_n must also not shoot in the current round, because he will have a higher kill probability in his next turn, when they will be at distance $d-2$.
2. If P_{-n} will shoot in the next round, when their distance will be $d-1$, then P_n should shoot if $p_{n,d}$ (his kill probability now) is higher than $1 - p_{-n,d-1}$ (P_{-n}'s miss probability in the next round). In other words, P_n must shoot if

$$p_{n,d} \geq 1 - p_{-n,d-1}$$

or, equivalently, if

$$K_{n,d} = p_{n,d} + p_{-n,d-1} \geq 1. \tag{1}$$

Hence, we can reason as follows:

1. At vertex 1, P_2 is active and his only choice is to shoot.
2. At vertex 2, P_1 is active and he knows P_2 will certainly shoot in the next round. Hence, P_1 will shoot if he has an advantage, i.e., if

$$Q_1(2) \geq Q_1(1) \Leftrightarrow 2p_{1,2} - 1 \geq 1 - 2p_{2,1} \Leftrightarrow p_{1,2} + p_{2,1} \geq 1.$$

This is equivalent to $K_{1,1} \geq 1$ and will always be true.
3. Hence, at vertex 3, P_2 is active and he knows that P_1 will certainly shoot in the next round (at vertex d). So, P_2 will shoot if

$$Q_1(3) \leq Q_1(2) \Leftrightarrow 1 - 2p_{2,3} \leq 2p_{1,2} - 1 \Leftrightarrow p_{2,3} + p_{1,2} \geq 1.$$

which is equivalent to $K_{2,3} \geq 1$. Also, if $K_{2,3} < 1$, then P_1 will know, when the game is at vertex 4, that P_2 will not shoot when at 3. So, P_1 will not shoot when at 4. But then,

when at 5, P_2 knows that P_1 will not shoot when at 4. Continuing in this manner, we see that $K_{2,3} < 1$ implies that firing will take place exactly at the vertex 2.

4. On the other hand, if $K_{2,3} \geq 1$, then P_1 knows when at 4 that P_2 will shoot at the next round. So, when at 4, P_1 should shoot if $K_{1,4} \geq 1$. If, on the other hand, $K_{1,4} < 1$, then P_1 will not shoot when at 4 and P_2 will shoot when at 3.
5. We continue in this manner for increasing values of d. Since both $K_{1,d}$ and $K_{2,d}$ are decreasing with d, there will exist a maximum d_n value (it could equal D) in which some $K_{n,d}$ will be greater than one and P_n will be active; then, P_n must shoot as soon as the game reaches or passes vertex d_n and he "has the action".

This completes the proof. □

Returning to our previous example, we compute the vectors K_n for $n \in \{1, 2\}$ and list them in the following Table 2.

Table 2. Shooting criterion.

d	**1**	**2**	**3**	**4**	**5**	**6**
Round	**6**	**5**	**4**	**3**	**2**	**1**
$K_{1,d}$		1.5000	1.0404	0.8274	0.7000	0.6139
$K_{2,d}$		1.7071	1.0774	0.8333	0.6972	0.6082

For P_1 the shooting criterion is last satisfied when the distance is $d = 3$; this happens at round 4, in which P_1 is inactive, so he should shoot at round 5. However, for P_2 the shooting criterion is also last satisfied at distance $d = 3$ and round 4, in which P_2 is active; so, he should shoot at round 4. This is the same result we obtained with backward induction.

4. Solution with Incomplete Information

As already mentioned, the implementation of either the backward induction or the shooting criterion requires *complete information*, i.e., knowledge by both players of all game parameters: D, $\mathbf{p}_1$, and $\mathbf{p}_2$.

In what follows, we will assume that the players' kill probabilities have the functional form (where θ is a parameter vector):

$$\forall d : p_{1,d} = f(d; \theta_1), \quad p_{2,d} = f(d; \theta_2)$$

Suppose now that both players know D and $f(d; \theta)$ but, for $n \in \{1, 2\}$, P_n only knows his own parameter vector $\theta = \theta_n$ and is ignorant of the opponent's θ_{-n}. Hence, each P_n knows $\mathbf{p}_n$, but not $\mathbf{p}_{-n}$. Consequently, neither player knows his payoff function, which depends on both $\mathbf{p}_1$ and $\mathbf{p}_2$.

In this case, obviously, the players cannot perform the computations of either backward induction or the shooting criterion. Instead, *assuming that the players will engage in multiple duels*, we propose the use of an heuristic "*exploration-and-exploitation*" approach in which each player initially adopts a random strategy and, using information collected from played games, he gradually builds and refines an estimate of his optimal strategy. Our approach is implemented by Algorithm 1, presented below in pseudocode.

Algorithm 1 Learning the Optimal Duel Strategy.

1: **Input**: Duel parameters D, θ_1, θ_2; Learning parameters λ, σ_0; Number of plays R
2: $\mathbf{p}_1 = \text{CompKillProb}(\theta_1)$
3: $\mathbf{p}_2 = \text{CompKillProb}(\theta_2)$
4: Randomly initialize parameter estimates θ_1^0, θ_2^0
5: **for** $r \in \{1, 2, \ldots, R\}$ **do**
6: $\quad \mathbf{p}_1^r = \text{CompKillProb}(\theta_1^{r-1})$
7: $\quad \mathbf{p}_2^r = \text{CompKillProb}(\theta_2^{r-1})$
8: $\quad d_1^r = \text{CompShootDist}(\mathbf{p}_1, \mathbf{p}_2^r)$
9: $\quad d_2^r = \text{CompShootDist}(\mathbf{p}_1^r, \mathbf{p}_2)$
10: $\quad \sigma_r = \sigma_{r-1}/\lambda$
11: $\quad X = \text{PlayDuel}(\mathbf{p}_1, \mathbf{p}_2, d_1^r, d_2^r, \sigma_r, X)$
12: $\quad (\widehat{\mathbf{p}}_1^r, \widehat{\mathbf{p}}_2^r) = \text{EstKillProb}(X)$
13: $\quad \theta_1^r = \text{EstPars}(\widehat{\mathbf{p}}_1^r, 1)$
14: $\quad \theta_2^r = \text{EstPars}(\widehat{\mathbf{p}}_2^r, 2)$
15: **end for**
16: **return** d_1^R, d_2^R, θ_1^R, θ_2^R

The following remarks explain the operation of the algorithm.

1. In line 1, the algorithm takes as input: (i) the duel parameters D, θ_1, and θ_2; (ii) two *learning parameters* λ, σ_0; and (iii) the number R of duels used in the learning process.
2. Then, in lines 2–3, the true kill probability $\mathbf{p}_n$ (for $n \in \{1, 2\}$) is computed by the function $\text{CompKillProb}(\theta_n)$, which simply computes

$$\forall n, d : p_n(d) = f_n(d; \theta_n).$$

   We emphasize that these are the *true* kill probabilities.
3. In line 4, randomly selected parameter vector estimates θ_1^0, θ_2^0 are generated.
4. Then, the algorithm enters the loop of lines 5–15 (executed for R iterations), which constitutes the main learning process.
   (a) In lines 6–7 we compute new *estimates* of the kill probabilities $\mathbf{p}_n^r$, by function CompKillProb, based on the estimates of parameters θ_n^{r-1}:

$$\forall n, d : p_n^r(d) = f_n\left(d; \theta_n^{r-1}\right).$$

   We emphasize that these are *estimates* of the kill probabilities, based on the parameter estimates θ_n^{r-1}.
   (b) In lines 8–9 we compute new *estimates* of the shooting thresholds d_n^r, by function CompShootDist. For P_n, this is achieved by computing the shooting criterion $\mathbf{K}_n$ using the (known to P_n) $\mathbf{p}_n$ and the (estimated by P_n) $\mathbf{p}_{-n}^r$.
   (c) In line 10, σ_r (which will be used as a standard deviation parameter) is divided by the factor $\lambda > 1$.
   (d) In line 11, the result of the duel is computed by the function PlayDuel. This is achieved as follows:
      i. For $n \in \{1, 2\}$, P_n selects a random shooting distance $\widehat{d}_n$ from the *discrete normal distribution* (Roy, 2003) with mean d_n^r and standard deviation σ_r.
      ii. With both $\widehat{d}_1, \widehat{d}_2$ selected, it is clear which player will shoot first; the outcome of the shot (hit or miss) is a Bernoulli random variable with success probability $p_{n,\widehat{d}_n}$, where P_n is the shooting player. Note that $\mathbf{p}_n$ is the *true* kill probability.

The result is stored in a table X, which contains the data (shooting distance, shooting player, hit or miss) of every duel played up to the r-th iteration.

(e) In line 12, the entire game records X are used by EstKillProb(X) to obtain empirical estimates of the kill probabilities $\widehat{\mathbf{p}}_1^r, \widehat{\mathbf{p}}_2^r$. These estimates are as follows:

$$\forall n, \forall d \in \mathcal{D}_n \colon \widehat{p}_{n,d}^r = \frac{\sum_{r \in \mathcal{R}_{n,d}} Z_r}{|\mathcal{R}_{n,d}|}$$

where

$$\begin{aligned} \mathcal{D}_n &= \{d : d \text{ such that } P_n \text{ may shoot}\} \\ \mathcal{R}_{n,d} &= \{r : \text{ in the } r\text{-th game } P_n \text{ actually shot from distance } d\}, \\ Z_r &= \begin{cases} 1 & \text{if the shot in the } r\text{-th game hit the target,} \\ 0 & \text{if the shot in the } r\text{-th game missed the target} \end{cases} \end{aligned}$$

(f) In lines 13–14 the function EstPars uses a least squares algorithm to find (only for the P_n who currently has the action) θ_n^r values which minimize the squared error

$$J(\theta_n) = \sum_{d \in \mathcal{D}_n} \left(f_n(d; \theta_n) - \widehat{p}_{n,d}^r \right)^2.$$

(g) In line 16, the algorithm returns the final estimates of optimal shooting distances d_1^R, d_2^R and parameters θ_1^R, θ_2^R.

The core of the algorithm is the exploration-exploitation approach, which is implemented by the gradual reduction of σ_r in line 10. Since $1/\lambda \in (0,1)$, we have $\lim_{r \to \infty} \sigma_r = 0$.

Exploration is predominant in the initial iterations of the algorithm, when the relatively large value of σ_r implies that the players use essentially random shooting thresholds. In this phase, P_n's shooting threshold and, consequently, his payoff, is suboptimal. However, since P_{-n} is also using various, randomly selected shooting thresholds, P_n collects information about P_{-n}'s kill probability at various distances. This information can be used to estimate θ_n; the fact that the functional form of the kill probabilities is known results in a tractable parameter estimation problem.

As r increases, we gradually enter the exploitation phase. Namely, σ_r tends to zero, which means that each P_n uses a shooting threshold that is, with high probability, very close to the one which is optimal with respect to $\widehat{\mathbf{p}}_{-n}^r$, and his current estimate of $\mathbf{p}_{-n}$, the opponent's kill probability. Provided a sufficient amount of information was collected in the exploration phase, $\widehat{\mathbf{p}}_{-n}^r$ is sufficiently close to $\mathbf{p}_{-n}$ to ensure that the used shooting threshold is close to the optimal one. The key issue is to choose a *learning rate* λ that is "sufficiently slow" to ensure that the exploration phase is long enough to provide enough information for convergence of the parameter estimates to the true parameter values, but not "too slow", because this will result in slow convergence. This is the *exploration-exploitation dilemma*, which has been extensively studied in the reinforcement learning literature. For several Q-learning algorithms it has been established that convergence to the optimal policy is guuaranteed, provided the learning rate is sufficiently slow (Ishii et al., 2002; Osband & Van Roy, 2017; Singh et al., 2000); in this manner, a balance is achieved between exploration, which collects data on long-term payoff, and exploitation, which maximizes near-term payoff. Similar conclusions have been established for *multi-agent* reinforcement learning (Hussain et al., 2023; Martin & Sandholm, 2021).

Since our approach is heuristic, the evaluation of an "appropriately slow" learning rate is performed by the numerical experiments in Section 5. Before proceeding, let us note

that "methods with the best theoretical bounds are not necessarily those that perform best in practice" (Martin & Sandholm, 2021).

5. Experiments

Since the motivation for our proposed algorithm is heuristic, in this section, we give an empirical evaluation of its performance. In the following numerical experiments, we will investigate the influence of both the duel parameters (D, $\mathbf{p}_1$, $\mathbf{p}_2$) and the algorithm parameters (λ, σ_0, R).

5.1. Experiments Setup

In the following subsection, we present several experiment groups, all of which share the same structure. Each experiment group corresponds to a particular form of the kill probability functions. In each case, the kill probability parameters, along with the initial player distance D, are the *game parameters*. For each choice of game parameters we proceed as follows.

First, we select the learning parameters λ, σ_0 and the number of learning steps R. These, together with the game parameters, are the *experiment parameters*. Then, we select a number J of estimation experiments to run for each choice of experiment parameters. For each of the J experiments, we compute the following quantities:

1. The relative error of the final kill probability parameter estimates. For a given parameter $\theta_{n,i}$, this error is defined to be
$$\Delta\theta_{n,i} = \left| \frac{\theta_{n,i} - \theta^R_{n,i}}{\theta_{n,i}} \right|.$$
2. The relative error of the shooting threshold estimates. Letting d^R_n be the estimate of the shooting threshold based on the true kill probability vector p_n and kill probability vector estimate p^R_{-n}, this error is defined to be
$$\Delta d_n = \left| \frac{d_n - d^R_n}{d_n} \right|.$$
3. The relative error of the optimal payoff estimates. Letting Q^R_n be the estimate of the optimal payoff (computed from the estimated shooting thresholds d^R_1, d^R_2), this error is defined to be
$$\Delta Q_n = \begin{cases} \left| \frac{Q_n - Q^R_n}{Q_n} \right| & \text{iff } Q_n \neq 0 \\ 0 & \text{iff } Q_n = 0 \text{ and } Q^R_n = Q_n \\ 1 & \text{iff } Q_n = 0 \text{ and } Q^R_n \neq Q_n \end{cases}$$
Note that $\Delta Q_2 = \Delta Q_1$, because the game is zero-sum.

5.2. Experiment Group A

In this group, the kill probability function has the form:

$$n \in \{1, 2\} : p_{n,d} = \min\left(\frac{c_n}{d^{k_n}}, 1\right).$$

Let us look at the final results of a representative run of the learning algorithm. With $c_1 = 1$, $k_1 = 0.5$, $c_2 = 1$, $k_2 = 1$ and $D = 10$, we run the learning algorithm with $R = 1500$, $\sigma_0 = 6D$, and for three different values $\lambda \in \{1.001, 1.01, 1.05\}$. In Figure 4 we plot the logarithm (with base 10) of the relative payoff error $\Delta Q_1 + \epsilon$ (we have added

$\epsilon = 10^{-3}$ to deal with the logarithm of zero error). The three curves plotted correspond to the λ values 1.001, 1.01, and 1.05. We see that, for all λ values, the algorithm achieves zero relative error; in other words, it learns the optimal strategy for both players. Furthermore, convergence is achieved by the 1500th iteration of the algorithm (1500th duel played), as seen by the achieved logarithm value -3 (recall that we have added $\epsilon = 10^{-3}$ to the error, hence the true error is zero). Convergence is fastest for the largest λ value, i.e., $\lambda = 1.05$, and slowest for the smallest value, $\lambda = 1.001$.

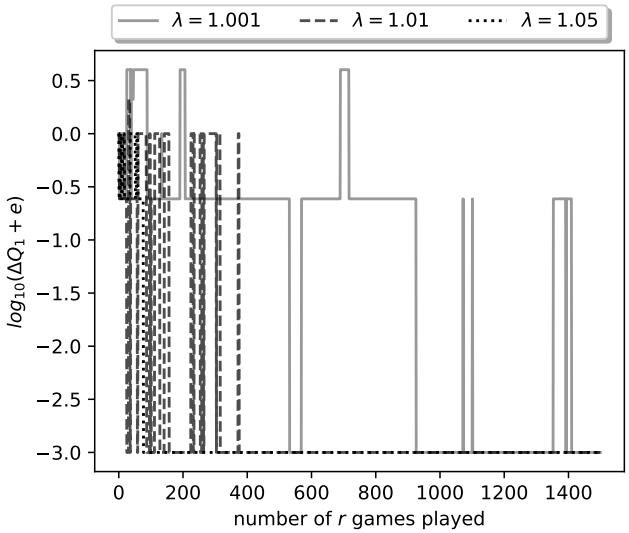

Figure 4. Plot of logarithmic relative error $\log_{10} \Delta Q_1$ of P_1's payoff for a representative run of the learning process.

In Figure 5, we plot the logarithmic relative errors $\log_{10} \Delta d_n$. These also, as expected, have converged to zero by the 1500th iteration.

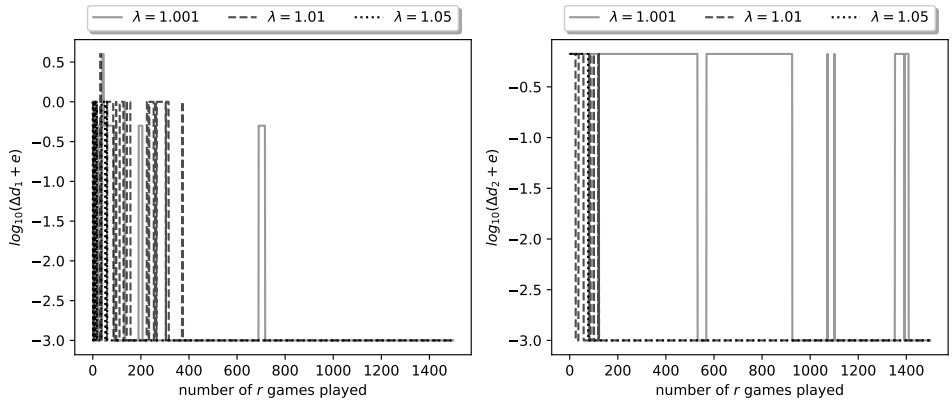

Figure 5. Plot of logarithmic relative errors $\log_{10} \Delta d_1$ and $\log_{10} \Delta d_2$ for a representative run of the learning process.

The fact that the estimates of the optimal shooting thresholds and strategies achieve zero error does not imply that the same is true of the kill probability parameter estimates. In Figure 6 we plot the relative errors Δc_1 and Δk_1.

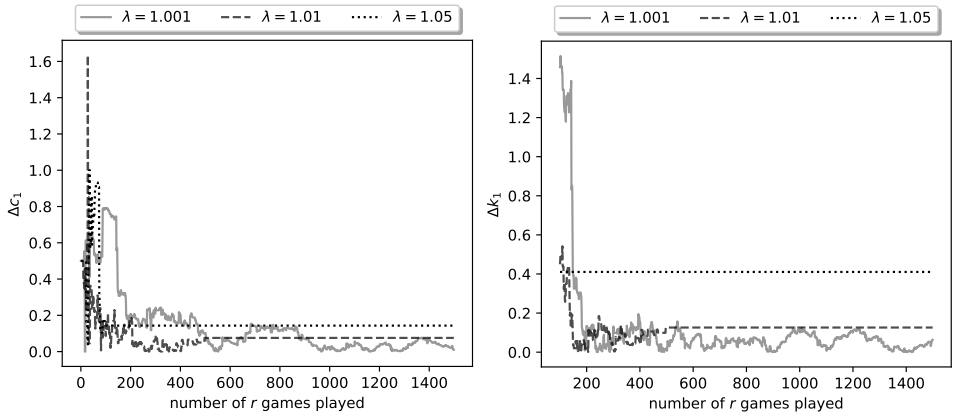

Figure 6. Plot of relative parameter errors Δc_1 and Δk_1 for a representative run of the learning process.

It can be seen that these errors do not converge to zero; in fact, for $\lambda \in \{1.01, 1.05\}$, the errors converge to fixed nonzero values, which indicates that the algorithm obtains wrong estimates. However, the error is sufficiently small to still result in zero-error estimates of the shooting thresholds. The picture is similar for the errors Δc_2 and Δk_2; hence, their plots are omitted.

In the above, we have given results for a particular run of the learning algorithm. This was a successful run, in the sense that it obtained zero-error estimates of the optimal strategies (and shooting thresholds). However, since our algorithm is stochastic, it is not guaranteed that every run will result in zero-error estimates. To better evaluate the algorithm, we ran it $J = 10$ times and averaged the obtained results. In particular, in Figure 7, we plot the average of ten curves of the type plotted in Figure 4. Note that now we plot the curve for $R = 5000$ plays of the duel.

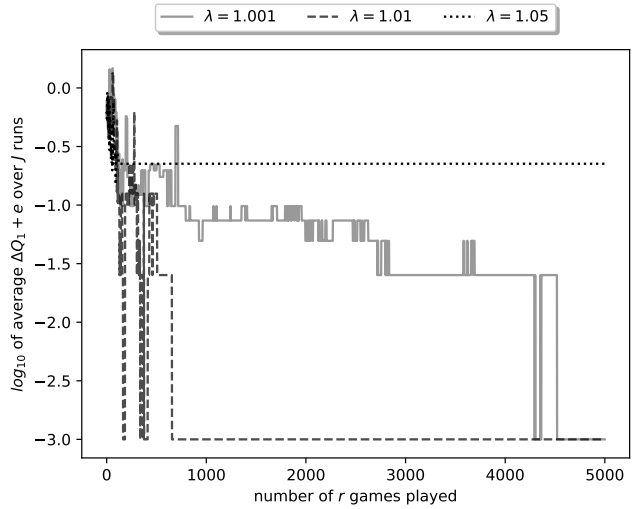

Figure 7. Plot of Q_1 (P_1's payoff) for a representative run of the learning process.

Several observations can be made regarding Figure 7.

1. For the smallest λ value, namely, $\lambda = 1.001$, the respective curve reaches -3 at $r = 4521$. This corresponds to zero *average* error, which means that, in some algorithm runs, it took more than 4500 iterations (duel plays) to reach zero error.
2. For $\lambda = 1.01$, all runs of the algorithm reached zero-error after $r = 656$ runs.
3. Finally, for $\lambda = 1.05$, the average error never reached zero; in fact, 3 out of 10 runs converged to nonzero-error estimates, i.e., to non-optimal strategies.

The above observations corroborate the remarks at the end of Section 4 regarding reinforcement learning. Namely, a small learning rate (in our case small λ) results in higher probability of converging to the true parameter values, but also in slower convergence. This can be explained as follows: a small λ results in higher σ_r values for a large proportion of duels played by the algorithm; i.e., in more extensive *exploration*, which however results in slower *exploitation* (convergence).

We concluded this group of experiments by running the learning algorithm for various combinations of game parameters; for each combination, we recorded the average error attained at the end of the algorithm (i.e., at $r = R = 5000$). The results are summarized in the following Tables 3–5.

Table 3. Values of final average relative error ΔQ_1 for $c_2 = 1$, $k_2 = 1$ and various values of c_1, k_1, λ. D is fixed at $D = 10$.

λ		**1.001**			**1.01**			**1.05**	
c_1	k_1	**0.50**	**1.00**	**1.50**	**0.50**	**1.00**	**1.50**	**0.50**	**1.00**	**1.50**
1.00		0.000	0.000	0.000	0.000	0.100	0.482	0.224	0.500	1.207
1.50		0.000	0.000	0.000	0.059	0.049	1.748	0.215	0.480	3.777
2.00		0.000	0.000	0.048	0.074	0.199	0.097	0.144	0.980	0.072

Table 4. Round at which ΔQ_1 converged to zero for all $J = 10$ sessions for $c_2 = 1$, $k_2 = 1$ and various values of c_1, k_1, λ. D is fixed at $D = 10$. If ΔQ_1 did not converge for all sessions, we note for how many sessions it converged.

λ	**1.001**			**1.01**			**1.05**		
c_1 k_1	**0.50**	**1.00**	**1.50**	**0.50**	**1.00**	**1.50**	**0.50**	**1.00**	**1.50**
1.00	4521	1456	2983	656	9/10	8/10	7/10	5/10	5/10
1.50	3238	2939	1754	7/10	9/10	9/10	0/10	8/10	6/10
2.00	4153	423	8/10	2/10	9/10	6/10	3/10	4/10	6/10

Table 5. Fraction of learning sessions that converged to $\Delta Q_1 - 0$ for different values of D and λ.

λ	**1.001**	**1.01**	**1.05**
D			
8	163/170	144/170	89/170
10	165/170	139/170	81/170
12	161/170	123/170	68/170
14	164/170	134/170	73/170
total	653/680	540/680	311/680

From Tables 6 and 7, we see that for $\lambda = 1.001$, almost all learning sessions conclude in zero ΔQ_1, while increasing the value of λ results in more sessions concluding with non-zero error estimates. Furthermore, we observe that when the average ΔQ_1 converges to zero for multiple values of λ, the convergence is faster for bigger λ. These results highlight the trade off between *exploration* and *exploitation* discussed above.

Finally, in Table 8, we see how many learning sessions were run and how many converged to the zero error estimate ΔQ_1 for different values of D and λ.

Table 6. Values of final average relative error ΔQ_1 for $d_{21} = 1$, $d_{22} = D$ and various values of d_{11}, d_{12}, λ. D is fixed at $D = 10$.

d_{11}	λ / d_{12}	**1.001**	**1.01**	**1.05**
1	$D/2$	0.000	0.233	1.000
1	$2D/3$	0.000	0.000	0.799
1	D	$\sim 6{\cdot}10^{-16}$	$\sim 2{\cdot}10^{-16}$	0.800
$D/3$	$2D/3$	0.000	0.000	2.799
$D/3$	D	0.000	0.000	0.777
$D/2$	D	0.000	0.000	0.480

Table 7. Round at which ΔQ_1 converged to zero for all $J = 10$ sessions for $d_{21} = 1$, $d_{22} = D$ and various values of d_{11}, d_{12}, λ. D is fixed at $D = 10$. If ΔQ_1 did not converge for all sessions we note for how many sessions it converged.

d_{11}	λ / d_{12}	**1.001**	**1.01**	**1.05**
1	$D/2$	287	7/10	0/10
1	$2D/3$	812	333	9/10
1	D	7/10	9/10	9/10
$D/3$	$2D/3$	326	139	5/10
$D/3$	D	225	397	5/10
$D/2$	D	711	464	6/10

Table 8. Fraction of learning sessions that converged to $\Delta Q_1 - 0$ for different values of D and λ.

D / λ	**1.001**	**1.01**	**1.05**
8	76/110	80/110	49/110
10	97/110	95/110	56/110
12	94/110	78/110	39/110
14	100/110	84/110	40/110
total	367/440	337/440	184/440

5.3. Experiment Group B

In this group, the kill probability function is piecewise linear:

$$p_{n,d} = \begin{cases} 1 & \text{when } d \in [1, d_{n1}] \\ -\frac{1}{d_{n2}-d_{n1}} d + \frac{d_{n2}}{d_{n2}-d_{n1}} & \text{when } d \in [d_{n1}, d_{n2}] \\ 0 & \text{when } d \in [d_{n2}, D] \end{cases}$$

Let us look again at the final results of a representative run of the learning algorithm. With $d_{11} = D$, $d_{12} = D/3$, $d_{21} = 1$, $d_{22} = D$, and $D = 8$, we run the learning algorithm with $R = 500$, $\sigma_0 = 6D$, and for the values $\lambda \in \{1.001, 1.01, 1.05\}$. In Figure 8 we plot the logarithm (with base 10) of the relative payoff error $\Delta Q_1 + \epsilon$. We see similar results as in Group A, for all λ values, the algorithm achieves zero relative error. Convergence is achieved by the 300th iteration of the algorithm and it is fastest for $\lambda = 1.05$, and slowest for $\lambda = 1.001$. The plots of the errors $\log_{10} \Delta d_1$, $\log_{10} \Delta d_2$ are omitted, since they are similar to the ones given in Figure 5.

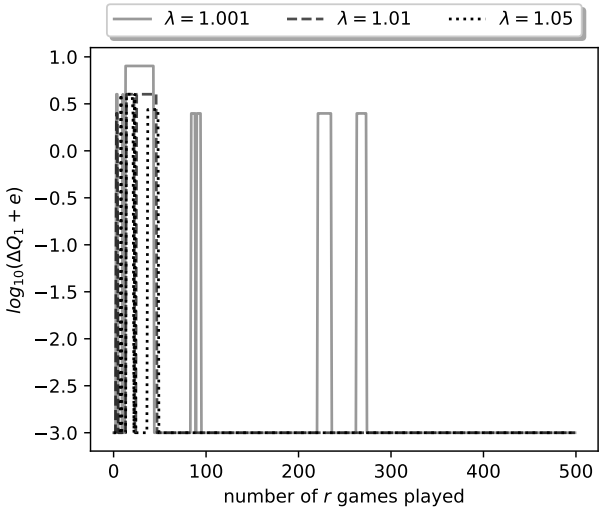

Figure 8. Plot of logarithmic relative error $\log_{10} \Delta Q_1$ of P_1's payoff for a representative run of the learning process.

As in Group A, the fact that the estimates of the optimal shooting thresholds and strategies achieve zero error does not imply that the same is true for the kill probability parameter estimates. For example, in this particular run, the relative error Δd_{11} converges to a nonzero value, i.e., the algorithm obtains a wrong estimate of d_{11}. However, the error is sufficiently small to still result in a zero-error estimate of the shooting thresholds.

As in Group A, to better evaluate the algorithm, we ran it $J = 10$ times and averaged the obtained results. In particular, in Figure 9, we plot the average of ten curves of the type plotted in Figure 4. Note that we now plot the curve for $R = 500$ plays of the duel.

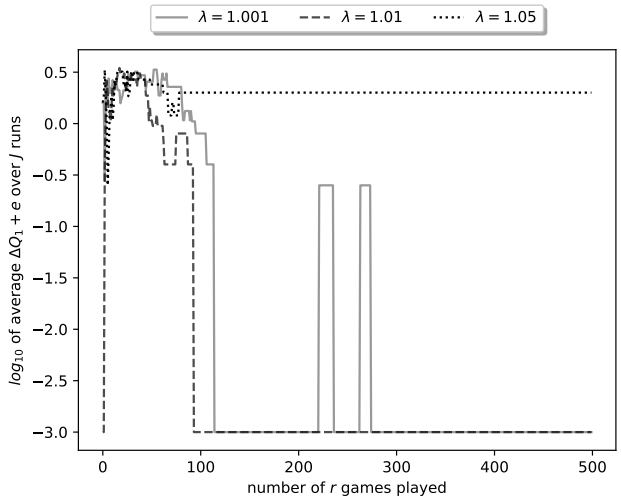

Figure 9. Plot of Q_1 (P_1's payoff) for a representative run of the learning process.

We again ran the learning algorithm for various combinations of game parameters and recorded the average error attained at the end of the algorithm (again at $r = R = 5000$) for each combination. The results are summarized in the following tables.

From Tables 6 and 7, we observe that for most parameter combinations, all learning sessions concluded with a zero ΔQ_1 for the smaller λ values. However, for $\lambda = 1.05$, the algorithm failed to converge and exhibited a high relative error. Notably, increasing λ from 1.001 to 1.01 generally accelerated convergence, although this is not guaranteed in every case. In one instance, a higher λ (specifically 1.01) led to a slower average convergence of ΔQ_1 to zero, indicating the importance of initial random shooting choices. We also

observed that the results for the highest λ were suboptimal, with the algorithm failing to converge in varying numbers of sessions, such as in 1 out of 10 or even 5 out of 10 cases. Notably, when convergence did occur, it happened relatively quickly, with sessions typically completing in under 1000 iterations. These findings also highlight the trade-off between *exploration* and *exploitation*, as discussed earlier.

Finally, in Table 8, we see how many learning sessions were run and how many converged to the zero-error estimate ΔQ_1 for different values of D and λ.

6. Discussion

We proposed an algorithm for estimating unknown game parameters and optimal strategies for a duel game through the course of repeated plays. We tested an algorithm for two models of the kill probability function and found that it converged for the majority of tests. Furthermore, we observed the established relationship between higher learning rates and reduced convergence quality, underscoring the trade-off between learning speed and stability in convergence. Future research could investigate additional models of the accuracy probability function, including scenarios where the two players employ distinct models, and work toward establishing theoretical bounds on the algorithm's probabilistic convergence.

Author Contributions: All authors contributed equally to all parts of this work, namely conceptualization, methodology, software, validation, formal analysis, writing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Notes

1 This proposition is stated informally in Polak (2007).

2 We use the same notation for several other quantities as will be seen in the sequel.

References

Alexander, J. M. (2023). *Evolutionary game theory*. Cambridge University Press.

Alonso, E., D'Inverno, M., Kudenko, D., Luck, M., & Noble, J. (2001). Learning in multi-agent systems. *The Knowledge Engineering Review*, *16*, 277–284. [CrossRef]

Brown, G. W. (1949). *Some notes on computation of games solutions* (Rand Corporation Report). Rand Corporation.

Cressman, R. (2003). *Evolutionary dynamics and extensive form games*. MIT Press.

Fox, M., & Kimeldorf, G. S. (1969). Noisy duels. *SIAM Journal on Applied Mathematics*, *17*, 353–361. [CrossRef]

Fudenberg, D., & Levine, D. K. (1998). *The theory of learning in games*. MIT Press.

Fudenberg, D., & Levine, D. K. (2016). Whither game theory? Towards a theory of learning in games. *Journal of Economic Perspectives*, *30*, 151–170. [CrossRef]

Garnaev, A. (2000). Games of Timing. In *Search games and other applications of game theory* (pp. 81–120). Springer.

Gronauer, S., & Diepold, K. (2022). Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review*, *55*, 895–943. [CrossRef]

Harsanyi, J. C. (1962). Bargaining in ignorance of the opponent's utility function. *Journal of Conflict Resolution*, *6*, 29–38. [CrossRef]

Hussain, A., Belardinelli, F., & Paccagnan, D. (2023). The impact of exploration on convergence and performance of multi-agent Q-learning dynamics. In *International conference on machine learning* (Vol. 1). PMLR.

Ishii, S., Yoshida, W., & Yoshimoto, J. (2002). Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural Networks*, *15*, 665–687. [CrossRef] [PubMed]

Jain, G., Kumar, A., & Bhat, S. A. (2024). Recent developments of game theory and reinforcement learning approaches: A systematic review. *IEEE Access*, *12*, 9999–10011. [CrossRef]

Leonardos, S., Piliouras, G., & Spendlove, K. (2021). Exploration-exploitation in multi-agent competition: Convergence with bounded rationality. *Advances in Neural Information Processing Systems*, *34*, 26318–26331.

Martin, C., & Sandholm, T. (2021, February 8–9). *Efficient exploration of zero-sum stochastic games* [Conference session]. AAAI Workshop on Reinforcement Learning in Games, Virtual Workshop.

Mertikopoulos, P., & Sandholm, W. H. (2016). Learning in games via reinforcement and regularization. *Mathematics of Operations Research*, *41*, 1297–1324. [CrossRef]

Nowe, A., Vrancx, P., & De Hauwere, Y.-M. (2012). Game theory and multi-agent reinforcement learning. In *Reinforcement learning: State-of-the-art* (pp. 441–470). Springer.

Osband, I., & Van Roy, B. (2017). Why is posterior sampling better than optimism for reinforcement learning? In *International conference on machine learning* (pp. 2701–2710). PMLR.

Polak, B. (2007). *Backward induction: Reputation and duels*. Open Yale Courses. Available online: https://oyc.yale.edu/economics/econ-159/lecture-16 (accessed on 19 January 2025).

Prisner, E. (2014). *Game theory through examples* (Vol. 46). American Mathematical Society.

Radzik, T. (1988). Games of timing related to distribution of resources. *Journal of Optimization Theory and Applications*, *58*, 443–471. [CrossRef]

Rezek, I., Leslie, D. S., Reece, S., Roberts, S. J., Rogers, A., Dash, R. K., & Jennings, N. R. (2008). On similarities between inference in game theory and machine learning. *Journal of Artificial Intelligence Research*, *33*, 259–283. [CrossRef]

Robinson, J. (1951). An iterative method of solving a game. *Annals of Mathematics*, *54*, 296–301. [CrossRef]

Roy, D. (2003). The discrete normal distribution. *Communications in Statistics-Theory and Methods*, *32*, 1871–1883. [CrossRef]

Singh, S., Jaakkola, T., Littman, M. L., & Szepesvari, C. (2000). Convergence results for single-step on-policy reinforcement learning algorithms. *Machine Learning*, *38*, 287–308. [CrossRef]

Tanimoto, J. (2015). *Fundamentals of evolutionary game theory and its applications*. Springer.

Weibull, J. W. (1997). *Evolutionary game theory*. MIT Press.

Yang, Y., & Wang, J. (2020). An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv*, arXiv:2011.00583.

Zamir, S. (2020). *Bayesian games: Games with incomplete information*. Springer.