Vrije Universiteit Brussel

Reinforcement learning in path-dependent, non-ergodic contexts

Verbruggen, B.; Vanhoyweghen, A.; Ginis, V.

Published in:
Proceedings of the Royal Society A

DOI:
10.1098/rspa.2024.0415

Publication date:
2025

License:
CC BY

Document Version:
Accepted author manuscript

Link to publication

Citation for published version (APA):
Verbruggen, B., Vanhoyweghen, A., & Ginis, V. (2025). Reinforcement learning in path-dependent, non-ergodic contexts. *Proceedings of the Royal Society A*, *480*(2314), Article 20240415.
https://doi.org/10.1098/rspa.2024.0415

Copyright
No part of this publication may be reproduced or transmitted in any form, without the prior written permission of the author(s) or other rights holders to whom publication rights have been transferred, unless permitted by a license attached to the publication (a Creative Commons license or other), or unless exceptions to copyright law apply.

Take down policy
If you believe that this document infringes your copyright or other rights, please contact openaccess@vub.be, with details of the nature of the infringement. We will investigate the claim and if justified, we will take the appropriate steps.

Download date: 14. Jul. 2026

PROCEEDINGS A

royalsocietypublishing.org/journal/rspa

Research

Article submitted to journal

Subject Areas:
Artificial Intelligence, Mathematical Modelling, Applied Mathematics

Keywords:
reinforcement learning, ergodicity, decision-making, mathematical modelling

Author for correspondence:
Bert Verbruggen
e-mail: bert.verbruggen@vub.be

Reinforcement Learning in path-dependent, non-ergodic contexts

B. Verbruggen[1], A. Vanhoyweghen[1,2], and V. Ginis[1,3]

[1]Data Analytics Lab, Vrije Universiteit Brussel, 1050 Brussel, Belgium
[2]Mobilise, Vrije Universiteit Brussel, 1050 Brussel, Belgium
[3]School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts 02138, USA
ORCID: **BV,** 0000-0001-9776-2420;**AV,** 0000-0003-0103-4715; **VG,** 0000-0003-0063-9608

Over the past decades, humanity has tried to automate decision-making processes, often by optimizing a specific value function. But how do you make a decision when significant risk is involved? Central to decision-making under uncertainty often lies the assumption of ergodicity, which implies that temporal evolution can be predicted from ensemble averages. In this paper, we study what happens to a Reinforcement Learning (RL) agent when ergodicity is broken in a decision-making context. First we show that in a traditional implementation, RL algorithms optimize towards an expected value of a stochastic process, having detrimental consequences for the temporal evolution of the agents in such a setting. Second, we propose a general mechanism that allows RL agents to capture the effects of ergodicity breaking and path dependencies, enabling them to converge in a more optimal strategy in terms of temporal evolution. Lastly, we implement this training mechanism in a variety of RL experiments.

© The Authors. Published by the Royal Society under the terms of the Creative Commons Attribution License http://creativecommons.org/licenses/by/4.0/, which permits unrestricted use, provided the original author and source are credited.

THE ROYAL SOCIETY PUBLISHING

These experiments show that our proposed method works for ergodic dynamics and non-ergodic contexts. We further apply our temporal training method to a simplified portfolio selection problem, and we show that our method allows an RL agent to approximate the Kelly criterion without transforming the value function.

1. Introduction

Historically, an entire scientific discipline is devoted to human decision-making [1,2]. Nowadays, we increasingly look to computers and algorithms for support in decision-making processes. Based on pre-defined rules, we delegate choices to automate the decision-making process efficiently. In recent years, Machine learning (ML) has taken over this purpose from traditional algorithms in various ways [3,4]. One of the ML techniques, Reinforcement Learning (RL) [5], uses an agent's experience to determine how to act in a specific situation. As a result, the agent develops a policy, some virtual decision tree on how it must respond to specific incentives in its world model. RL has proven to be very successful at various tasks [6–9] and seems to be a perfect technique to model human decision-making. Therefore, it is vital to understand how policies are formed during training.

Decisions often have uncertainty about what to expect, which introduces risk into the context. A typical example, as introduced in [10], involves human decision-making under multiplicative dynamics. A key insight in this description is the path dependence of the process. This means that the possible outcomes of the process depend on the specific events that occurred along the trajectories rather than on the current state alone. This leads to ergodicity breaking and results in different values for the ensemble average and the time average [11]. Intuitively, one would assume we benchmark options on expected values as an outcome of a group of independent but identical agents, culminating in distinctly different preferences when ergodicity is broken.

In our study, we aim to answer the question of whether RL develops optimal policies when a context is inherently non-ergodic. Is the expected value of a choice sufficient to formulate optimal policies within the RL formulation? If not, can we enhance classical RL techniques to allow them to pick up on temporal dynamics and thus improve individual agent predictions?

(a) Reinforcement Learning

Reinforcement learning is a popular technique used within the broad framework of Machine Learning. RL techniques stand apart from other methods as agents are presented only with possible actions ($a \in \mathbb{A}$) and positions, states ($s \in \mathbb{S}$), which they can occupy within the environment. These actions determine how the agents interact with the environment and move from one state to another. In each state, a specific reward can be assigned to the agent. As such, an agent moves through the virtual environment, collecting rewards ($r_i \in \mathbb{R}$) and ending with a final total reward. Reinforcement learning then uses an algorithm to update the agent's internal model of the environment based on these cumulated rewards. Hence, an agent learns from experience and updates its knowledge continuously. The origins of reinforcement learning are commonly attributed to the introduction of the Bellman policy evaluation equation [12], shown in Equation 1.1 and the theory of dynamic programming (DP) [13]. The update rule aims to optimize the value function $V(s)$ for the value in the state s. Central to these updates is the explicit dependence on the expected values of the rewards (r_{t+1}) and the value of future states ($V(s_{t+1})$) in its formulation

$$V(s) = \mathbb{E}\left[r_{t+1} + \gamma V(s_{t+1}) \mid s_t = s\right]. \tag{1.1}$$

Watkins later incorporated the Bellman equation and dynamic programming as an update rule in an algorithm. This Q-learning algorithm [14,15] uses the Bellman equation at its core to train an

agent based on the set of states and actions

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]. \tag{1.2}$$

The algorithm assigns a Q-value to each possible combination of states and actions (state-action pairs), indicating the expected quality of taking this specific action from the current state. These Q-values are updated based on the agent's experience during the training process. An alternative view on optimising the Q-values using the Bellman equations under uncertainty is introduced by Bellemare et al. [16]. The authors replace the expected value formulation at the core of the value function with the distribution of the returns. By incorporating distributions rather than the expected value, agents can discern between an average of decisions and its actual overall range of returns. As the authors show, the whole distribution of possible returns might not behave as well as the mean value of the distribution, which converges exponentially fast to the traditional optimal Q-value.

Some of the best-known success stories of RL are achieved in games or gamified settings [17–19]. The technique is, however, not limited to such cases only. Notably, RL techniques provided a leap forward to enhance modelling the protein folding of genomes [20], and in Large Language Models (LLM), the use of reinforcement learning plays a significant part in fine-tuning outputs of the model [21–23].

Modern approaches often use Deep Reinforcement Learning (DRL) by incorporating a neural network to map features of the environment to the action space [24]. Their main advantage is that DRL omits the curse of dimensionality when input spaces become too large to manage for traditional tabular methods or when input spaces are continuous [25,26]. Deep Q-Networks (DQN) [18,24] use a modified version of the traditional Q-learning algorithm, passing the input through a deep neural network before the action selection occurs.

Policy gradient methods such as PPO and TRPO resolve the scalability by using a gradient ascent to optimize policies [27,28]. Due to the complexity of problems for which DRL methods are applied, function approximation has become essential to handle the complexity where value functions or policies are approximated using (non)linear functions [29]. While these methods are state-of-the-art, they introduce new problems like those discussed by Schulman et al. [27]. In our work, we will focus on an underlying dynamic of the problem and study the impact of the essential formulation of the update rules as dictated by the Bellman equations. To be able to study our hypothesis in a clean environment, we present our work in a traditional RL context. By refraining from DQN models and the complexity introduced by neural networks and policy gradient methods, we measure the explicit impact of the update rules on the optimization of underlying dynamics.

In many applications of RL, the context is subject to uncertainty as outcomes depend on stochastic processes. Introducing uncertainty in reinforcement learning is an ongoing field with many interesting applications [30,31]. Several studies incorporate the uncertainty from the underlying processes as a risk factor in formulating the policy optimization for the agents.

Some specific techniques for reinforcement learning under uncertainty are safe RL and risk-aware RL. Safe reinforcement learning [32,33] aims to model agents when a wrong decision or action can lead to catastrophic behaviour. In a technology like autonomous driving [34], there is no room for error where, in general, translations for RL techniques to the real world suffer from certain safety concerns in the policies [35]. Such dynamics are, however, more generally related to a different type of ergodicity breaking and aligning with the scope of ruin or absorbing barriers [36]. The notion of risk-aware RL, on the other hand, aligns nicely with our discussion. Heger et al. [37] introduced a modified Q-learning algorithm to make agents aware of and avoid worst-case scenarios. A more recent study on Risk-averse RL introduces a tunable parameter for risk-taking and risk-averse agents [38]. Yet another interesting study [39] indicates the problem with adapting RL techniques to include variances in the expected rewards or mapping rewards to utility functions. The authors propose transforming the temporal difference (TD) factor in the RL update rules to be more resilient to optimization under different dynamics.

One study, in particular, works with a similar approach to transform the rewards and value function under the ergodic transform. Baumann et al. [40] use the concept of non-ergodic rewards to find a transformation that translates the RL updates to ergodic increments and, as such, maximizes the long-term growth rate.

Our study continues on RL under uncertainty but includes the path-dependent nature of the underlying processes and the inherent ergodicity breaking in the training context. In agreement with the work from Baumann et al., we take the non-ergodic nature into account but use a general method to include path dependence in the agents' updates rather than interventions in the value function. Before turning to our novel approach, we will introduce a general notion of ergodicity and how it is broken.

(b) Ergodicity

Reinforcement learning generally relies on the concept of training many agents in identical conditions, resulting in an arithmetic average or expected value. However, we often take an interest in optimizing a process for a single individual agent. In ergodic situations the optimization of the individual (time average) and a group (ensemble average) result in the same description. However, when ergodicity is broken the optimal results differ for an individual and a group of agents. The ergodic assumption thus states that ensemble averages over the expected outcome for a group of individuals are a proper estimator for the time evolution of one individual,

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T f\left(\omega\left(t\right)\right) dt = \int_\Omega f\left(\omega\right) P\left(\omega\right) d\omega. \tag{1.3}$$

The left-hand side describes the dynamics of the system over time. In an ergodic context, this equals the right-hand side, representing a population average for an optimizer function $f(\omega)$. When equality does not hold, which is often the case, we say ergodicity is broken. Some recent studies incorporate the notion of ergodicity in their research. In finance, the Kelly criterion [41] is a well-known argument for optimizing time growth, which we will also return to later. In [42], price dynamics in financial markets are studied under ergodicity breaking. A strong argument for cooperation and altruism, in general, follows from an Ergodicity Economics (EE) argument [43]. In ecology [44], include an argument on ergodicity breaking to study flight patterns of avian predators. In fractal structures, the breaking of ergodicity is often essential in its descriptions [45, 46]. The ergodic assumption generally does not hold in chaotic systems, where this can be applied to cancer cell population [47] or in general for stochastic processes [48].

The source of ergodicity breaking can have very different underlying processes, of which the most popular examples are multiplicative dynamics [10], absorbing boundaries [36], or positive feedback mechanisms like in Polya's urn [49,50]. A key feature that comes up in most of these processes is the explicit path dependence of the process. Most of our simplified models of economics, human decision-making, and reinforcement learning implicitly assume ergodicity is not broken. This is because we use the expected value as a first-order approach when modelling an individual's temporal evolution.

O. Peters sheds new light on the underlying dynamics behind human decision-making by calling the ergodic assumption into question for human decision-makers [10]. Peters suggests that humans value their time average rather than the ensemble average. This new view in the field of EE illustrates this effect for wealth optimization in a financial context. This new view in the field of EE illustrates the effect for wealth optimization in a financial context. When agents repeatedly take a gamble in a non-ergodic context, the expectation value of the wealth over an infinite ensemble of agents ($\mathbb{E}[W]$) is not a reliable estimate for an individual's wealth (W). A better description relies on the time average growth rate of the process (g), as indicated in Figure 1, which represents the average rate of change for a quantity (such as wealth) for an individual agent along a trajectory. Interestingly, by providing a rigorous mathematical basis for interpreting human behaviour, the discrepancies from expected value optimization no longer need to be a sole consequence of psychologically motivated biases and heuristics [51,52].

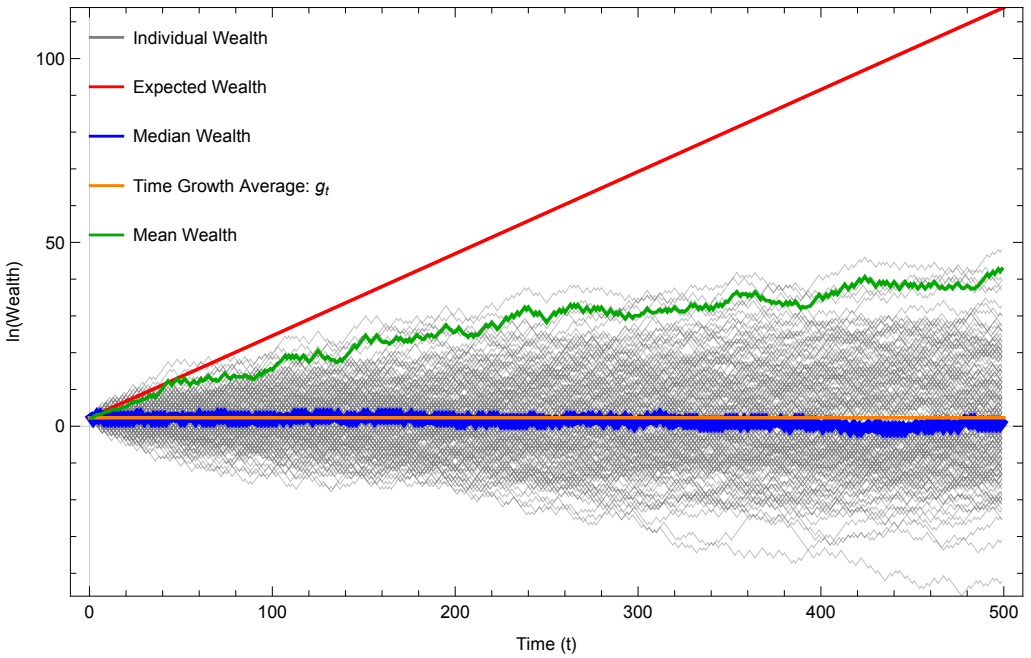

Figure 1. Graphical representation of wealth trajectories in a binomial process illustrating ergodicity breaking. We model the evolution of wealth for 200 agents with starting wealth $W = 10$ subjected to a binomial process with multiplicative dynamics, where each agent's wealth either doubles or halves with a 50% probability at each time step. Over 500 iterations, individual wealth trajectories (grey lines) diverge significantly from the expected wealth (red line), as does the group mean wealth (green line). The median wealth (blue line) aligns more closely with the average time growth (g, orange line), highlighting the discrepancy between expected outcomes and individual experiences in a non-ergodic context.

For the process illustrated in Figure 1, it can be shown [10] that the average time growth rate (g) is given by

$$g = \mu - \frac{1}{2}\sigma^2. \tag{1.4}$$

To maximize growth rates, it is clear that not only maximizing the drift (μ) matters but also minimizing the volatility of the process (σ^2). This idea can be introduced in the world of reinforcement learning by introducing an agent to a non-ergodic dynamic, where rewards are cumulated multiplicatively. In his tutorial on random multiplicative processes, Redner [11] explains how there is no equivalent for the Central Limit Theorem when describing multiplicative dynamics as opposed to additive increments. The author shows that the average is dominated by extreme events for multiplicative dynamics, which become exponentially rare for longer sequences. Therefore, the process's most likely value gets increasingly far from the theoretical average. An important result from the discussion shows that for binomial multiplicative processes, the optimal value is the most probable value of the process rather than the expected value.

In our work, we first create a generic environment to test the optimization hypothesis by RL within a non-ergodic context and evaluate standard RL models in such environments. Additionally, we present a technique that introduces the path dependence of random multiplicative processes by repeating the agent's decision within one training step before updating value functions and moving to the next episode. In short, the nature of our experimental design allows the RL agents to exploit the temporal dynamics inherent in path-dependent processes without directly interfering with reward or value functions.

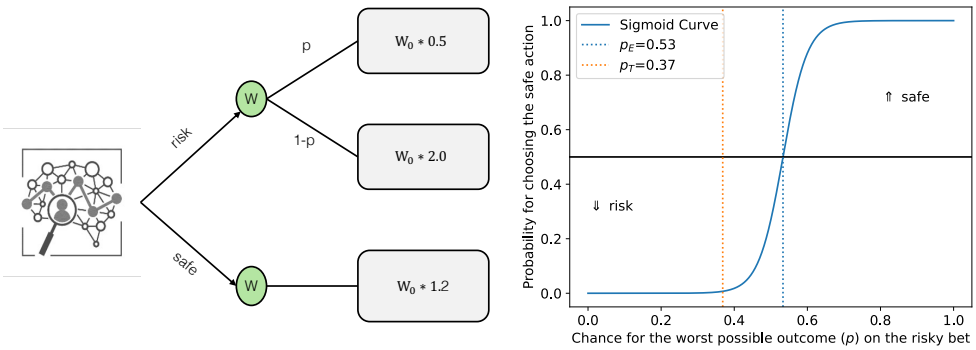

Figure 2. Decision-making model of an agent facing a binary choice between a risky and a safe option in a Bernoulli process, depicted through a coin flipping game (left diagram). The risk involves the agent's wealth being either halved ($W_0 * 0.5$) with probability p, or doubled ($W_0 * 2$) with probability $(1 - p)$. In contrast, the safe option yields a consistent growth ($W_0 * 1.2$). The right graph shows the agent's policy shift from risk-taking to risk-averse based on the varying probability of the worst outcome (p). The logistic curve represents the normalized probability of choosing the safe bet over the risky bet, with the inflection point p_0 indicating the threshold at which the agent's strategy shifts from risk-seeking to risk-averse behaviour. We show a perfect expected value optimizer as the inflection point (p_0) is located at the point where an agent flips policies for expected value optimization (p_E).

2. Methods

In our first experiment, we consider a binomial process which uses a multiplicative procedure to generate its outcome: $W_0 * 0.5$ with probability p and $W_0 * 2.0$ with probability $(1 - p)$. This dynamic is inherently non-ergodic, as discussed by [10] and [11], which is also shown in Figure 1. We take this bet and provide the agent with an alternative where we multiply the initial wealth with a small factor, say $(W_0 * 1.2)$. We can call this bet the *safe bet* and the original Binomial coin flip the *risky bet*. This process can be visualized as in Figure 2.

(a) Measuring policies

In Reinforcement Learning (RL), we seek to identify a policy for an agent that indicates the best possible choice an agent can make. An optimal choice for an agent in our environment means we seek to optimize the agent's time-average growth rate rather than its expected value, as introduced in the previous section. In our experiment, the policy results in a preference level for making one of two choices. The safe choice entails no risk of loss over the choice with a possibility of a larger gain but, with some chance p, a loss in wealth. To evaluate this in the context of ergodicity breaking, we must vary the value of the parameter p, which determines the probability of loss in the risky situation, the binomial process in Figure 2. We translate the experiment into a traditional RL setting where the agent has a single state with two possible actions: risk and safe.

When the agent takes the safe bet, we multiply its initial wealth (W_t) with the multiplicative factor for the safe action,

$$W_{t+1} = W_t * r_3. \tag{2.1}$$

When the agent chooses the risky bet, the outcome depends on the outcome of a binomial as indicated in Equation 2.2.

$$W_{t+1} = \begin{cases} r_1 \cdot W_t & \text{probability } p \\ r_2 \cdot W_t & \text{probability } 1 - p. \end{cases} \tag{2.2}$$

This setup leads to a straightforward experiment wherein an agent needs to maximize its wealth by choosing the most optimal bet. First, we assign a value to parameter p of the risky gamble and the update rule to be used by the agent. We show the expected value in Equation 2.3 for the risky bet and Equation 2.4 for the safe bet.

$$\mathbb{E}_{risk}\left[W_{t+1}\right] = (p \cdot r_1 + (1-p) \cdot r_2) \cdot W_t \tag{2.3}$$

$$\mathbb{E}_{safe}\left[W_{t+1}\right] = r_3 \cdot W_t \tag{2.4}$$

We sweep the value of the parameter p to evaluate the agent's sensitivity to the specific context in determining its policy. Given a specific value for p, we can evaluate how likely the agent is to take the safe bet compared to the risky bet. The agent is now trained over several episodes, and in each episode, they have to make a decision between the safe and risky action. Each episode ends by updating the Q-value for the respective actions with the resulting wealth after the decision. The policy is determined using these two Q-values after finishing the final training episode. As the policy reflects the preference of the agent to choose the safe action over the risky action, we use a softmax function with both Q-values, in which we subtract the maximum Q-values from each one to have a normalized view of the preference. The $Q(s, safe)$ state-action pair will yield a value between 0 and 1 after the softmax operation, indicating the agent's preference for playing safe from the final policy.

We repeat this process for a set of independent identical agents to get a statistical interpretation of the estimate ($\hat{p}_0$) for the inflection point (p_0). A proper policy will start asymptotically at 0 preference when the agent cannot experience the worst possible outcome from the risk action, $p = 0$. When the agent is bound only to receive the worst outcome, $p = 1$, the policy should reflect a strict preference for the safe action, asymptotic to 1 as illustrated in Figure 2. Between these two extremes, the agent should shift from a convex to a concave function with a specific inflection point, where the policy has no preference for either action. One standard function that describes such behaviour is the sigmoid function

$$\sigma(p) = \frac{1}{1 + e^{-k(p-p_0)}}. \tag{2.5}$$

The inflection point indicated as p_0 dictates the value for parameter p where the agent switches its policy from risk-taking to risk-averse. The parameter k determines the slope of the regression function and stretches the sigmoid function to change policy within the interval $[0, 1]$. The slope measures how fast the agent changes policies but does not change the location of the inflection point itself. It is this exact location of the inflection point, i.e. the value of parameter p, where the RL agent decides to switch from preferring the risky gamble to the safe gamble.

This point can theoretically be predicted by using expected rewards. For an expected reward optimizing agent, the inflection point occurs when the expected reward for the risk and safe actions, Equations 2.3 and 2.4 equal to each other; we find the prediction on the location of the inflection point (p_E).

$$\begin{aligned} \mathbb{E}_{safe}\left[W_t\right] &= \mathbb{E}_{risk}\left[W_t\right] \\ \Rightarrow r_3 &= p_E \cdot r_1 + (1-p_E) \cdot (r_2) \\ \Rightarrow p_E &= \frac{r_3 - r_2}{r_1 - r_2} \end{aligned}$$

We refer to this value as the expected value estimate of the inflection point (p_E). As we derived it from the expected wealth, this inflection point indicates where the agent should change policies according to expected value optimization. When RL agents are in line with optimizing towards expected rewards, the regression of our logistic function should estimate the inflection point at this location. On the other hand, as the system has non-ergodic dynamics, a different time-optimal value of the inflection point exists for the agent. Considering the process from a time-growth perspective, the probability for a specific outcome is modelled as a Bernoulli process. For time

growth averages, we can determine this inflection point (p_T) where the cumulated outcomes according to the probability of receiving the worst possible outcome in the risky bet (p) are equal to the outcome for the safe bets.

$$\begin{aligned} r_3 &= (r_1)^{p_T} \cdot (r_2)^{(1-p_T)} \\ \Rightarrow \ln(r_3) &= \ln\left[(r_1)^{p_T} \cdot (r_2)^{(1-p_T)}\right] \\ \Rightarrow \ln(r_3) &= p_T \cdot \ln(r_1) + (1-p_T) \cdot \ln(r_2) \\ \Rightarrow p_T &= \frac{\ln(r_3) - \ln(r_2)}{\ln(r_1) - \ln(r_2)} \end{aligned}$$

An agent optimizing for the time growth driving the process would learn to switch preferences around this time-growth value (p_T) rather than the one based on expected rewards (p_E).

(b) Model selection

Previously, the concept of an update rule for RL was mentioned, but many different methods exist. Our experimental setup has just one state and two actions, and we study the changing dynamics using several RL algorithms. Table 1 introduces the update rules we use to train our agents, where s' signifies future states for the agent to occupy. As we only have one state, future states refer back to the original state from which the agent decides between the two actions, risk or safe, but with updated wealth. In what follows, we refer to different models based on the update rule used to train identical agents during the process. An additional model is introduced in this

Model	Update rule
Q-Learning	$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
SARSA	$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s',a') - Q(s,a)\right]$
Monte Carlo (MC)	$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + G - Q(s,a)\right]$
PTD	$Q(s,a) \leftarrow Q(s,a) + \alpha\left[\gamma \max_{a'} Q(s',a') - Q(s,a)\right]$

Table 1. Different update rules used for training the models.

list, the "pure temporal difference" or PTD model, which focuses only on the advantage of the temporal difference in its formulation. This model only uses the final wealth during its update process. The main idea of this model is based on removing intermediate rewards (r) from the update rule that are assigned during the agent's decisions [52]. In some contexts, we don't have direct access to intermediate rewards from the environment, and our agent needs to work with the terminal wealth instead [51]. The experiment describes an environment with finite state-action pairs, allowing for discrete RL models. Training our agents means we update the corresponding Q-values for these state-action pairs ($Q(s,a)$). The learning rate (α) ensures our Q-values updates are proportional to be resilient to outlier cases.

Discounting (γ) manages how much value we want to put on future values rather than immediate updates. Except for the MC model, all our update rules rely on Temporal Difference (TD) learning. This means the agents can learn from incomplete episodes by using estimated values of successive states or prospects of future states ($Q(s',a')$). In contrast to the TD models, the MC model relies on the average of all explored rewards during an episode (G) before applying the update rule.

3. Results

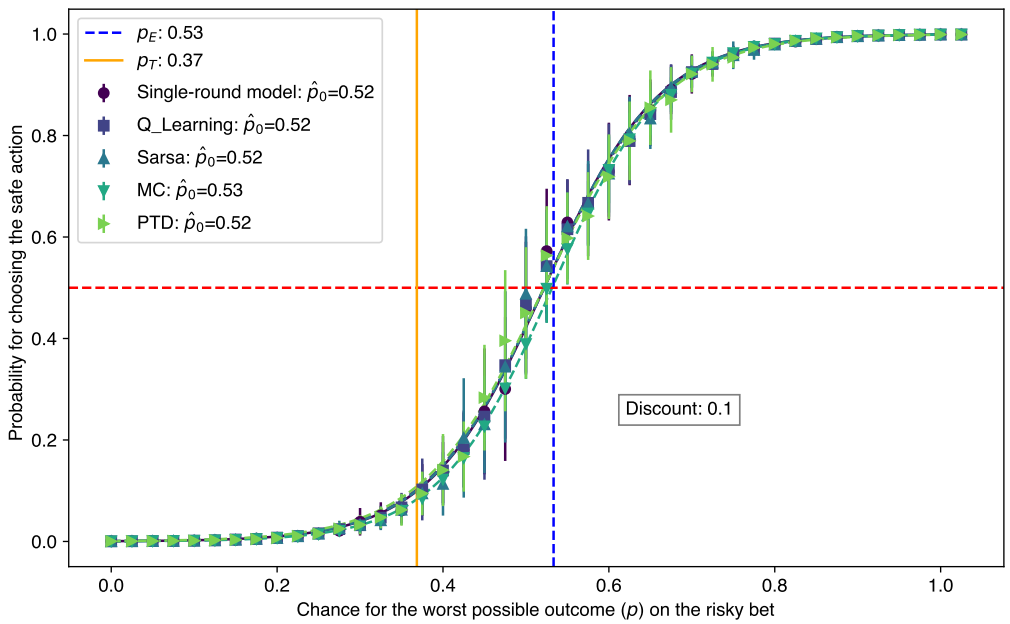

Figure 3. Comparative analysis of the final policies of agents in a single-round betting scenario across various RL models. Agents are trained to choose between a risky bet, where initial wealth (W_0) is doubled with probability $(1-p)$ or halved with probability p, and a safe bet yielding a constant increase ($W_0 * 1.2$). The graph plots the normalized preference for the safe bet against the probability of the worst outcome for the risky bet (p). Empirical estimates for inflection points ($\hat{p}_0$) from different models are compared to the expected value-based prediction (p_E) and average time growth prediction (p_T).

(a) One-round model training

As indicated in Figure 3, we can evaluate the different models in the non-ergodic context by calculating the inflection point of the policies for varying values of p. Before starting this experiment with all the models introduced in Table 1, we include one reference model that only adopts a single choice (single-round model). The model is a simplified version of the standard Q-learning algorithm where we only introduce a single update from the final wealth. We use this model as a benchmark for the perfect expected value estimate, as the optimization of this model can only rely on the expected value of one single decision. Figure 3 indicates that agents flip their policies around estimates for the inflection point ($\hat{p}_0$) that align with the expected value (p_E), regardless of the model by which the agent is trained. Consequently, standard RL models all work as expected value optimizers. Regardless of the update rule, the model's training results in a policy that prefers risk over safe bets until expected rewards for the safe bet become higher ($\hat{p}_0 = p_E$). In Figure 3, we also show the alternative optimizer (p_T), which is not considered an optimal value by any of the models in this traditional training setup. We refer the reader to Appendix B for a detailed description of the model parameters. We further present our hyperparameter sweep methodology in Appendix C.

(b) Multiple Rounds: Including time dynamics

After confirming that RL agents typically optimize their decision-making process by maximizing the expected reward in our setup, we now include the path dependency in the training setup. This new methodology is where our work introduces the novel approach to make the agent aware of the broken ergodicity. To do so, we extend the game within a single training episode.

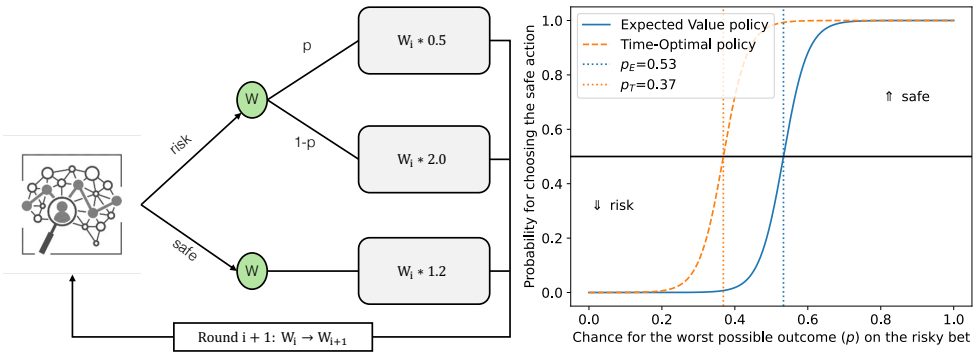

Figure 4. Adapted game layout for the explicit time extension of the RL training environment. The left panel demonstrates the bets available at each round (i), impacting the wealth in the subsequent round (W_{i+1}). The right graph plots the agent policy as a function of the probability p for the worst outcome on the risky bet. Two distinct policy curves are depicted: the Expected Value policy (blue line) and the Time-Optimal policy (orange dashed line), with respective inflection, points $p_E = 0.53$ and $p_T = 0.37$. These indicate the predictions where the agents' strategies pivot from risk preference to aversion, depending on the underlying optimizer.

For a fixed number of repetitions, the agent must make the same decision between the two actions, returning to the original state with the updated wealth each time. These repetitions, or rounds, are all stacked within one training episode. As such, the update rule is only applied after finishing all these repetitions and is then traced back to the single state with respective actions and intermediate cumulated wealth as rewards ($r(i)$).

$$Q(s, a, i+1) \leftarrow Q(s, a, i) + \alpha \left[r(i) + \gamma \max_{a'} Q(s', a', i) - Q(s, a, i) \right]. \tag{3.1}$$

The training still uses the same concept for episodes, but now the agent builds up a trajectory of decisions within one episode before updating the Q-values. The variable i denotes the round within one episode the agent has completed. Including this formal time step explicitly introduces the path dependence into the training context.

As discussed, the ergodic assumption is central to the question of optimizing policies. In its standard formulation, RL fails to capture time dependency in an ergodically broken context. Figure 4 illustrates the extension of the game in the virtual environment. An agent who is aware of the non-ergodic nature of the underlying process will learn to optimize for its time growth (p_T) rather than the expected value prediction (p_E).

Figure 5 shows the results for the different game repetitions ($[1, 2, 5, 10, 15, 20, 25, 30, 50, 100]$), each with regression function and resulting estimates of the inflection point $\hat{p}_0$. For a proper evaluation, one can visualize the evolution of the estimates $\hat{p}_0$. As the number of consecutive rounds increases, the expected value prediction (p_E) is not a good predictor of an individual's expected outcome. Hence, the agents need to find the trade-off to sacrifice some ideal prospect of high rewards for stability in their expected final reward. This trade-off for low variance is one of the fundamental predictions from optimizing time averages rather than expected values as proposed from an ergodicity economics (EE) point of view. An overview of this evolution is shown in Figure 6 for each model in the experiment. This result is consistent with various reward vectors and different hyperparameter settings. Our enhanced setup allows agents to optimize towards average time growth instead. With the proposed training method, RL techniques can be used without changes to the value function or transformations of the rewards, resulting in better predictions of expected outcomes for individual agents and time evolutions of the entire process. In Appendix C(d), we introduce a different visualization of the results to evaluate the

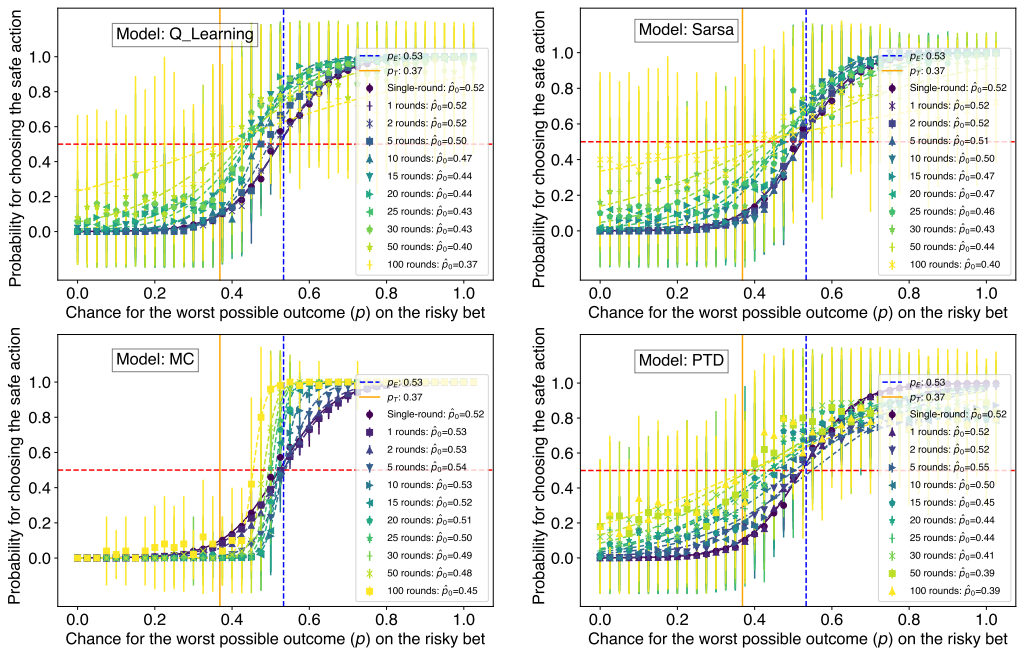

Figure 5. Visualization of the extended experiments to include multiple rounds of choices between safe and risky bets before policy updates, reflecting a non-ergodic process where outcomes accumulate over time. Each figure compares the estimated inflection points ($\hat{p}_0$) of different models with both the theoretical prediction based on expected values (p_E) and on average time growths (p_T). As the number of decision rounds increases, all models shift their policies towards those that factor in average time growth. This shift indicates that the agents are changing their policy towards a more optimal one, emphasizing the importance of considering path dependence in policy estimation to account for the non-ergodic nature of the simulated environment. All models are trained using the model parameters as presented in B.

policy change based on the average time growth rate in more detail.

(c) Simulation of optimal policy effects

Our new method to train RL agents to learn an optimal policy in a non-ergodic context allows us to simulate the difference for an ensemble of agents trained on both optimizers. We simulate trajectories for 500 agents for 500 time steps with a random selection of the parameter $p \in [0, 1]$. Each agent uses two policies to determine which bet to take. One policy is optimized using the average time growth (p_T), and the other one uses the expected value optimizer (p_E). This selection of values for the parameter p is preferred over a uniform distribution to reflect the main regime where the optimization differs ($p \in [p_T, p_E]$). Alternative choices can be made as discussed in Appendix C(e).

The resulting trajectories of the cumulated wealth for these agents are shown in Figure 7. The median values for all agents, distinguished by their policy, are added as dashed lines. Agents' cumulated wealth is shown on the vertical axis on a log scale (base 10) as these values increase exponentially.

(d) Policy evolution in an ergodic dynamic

In section 3(b), we propose an alternative approach to train reinforcement learning agents to capture the path dependence of agents in a random multiplicative process. The underlying breaking of ergodicity is paramount in this process and requires a change in perspective for the

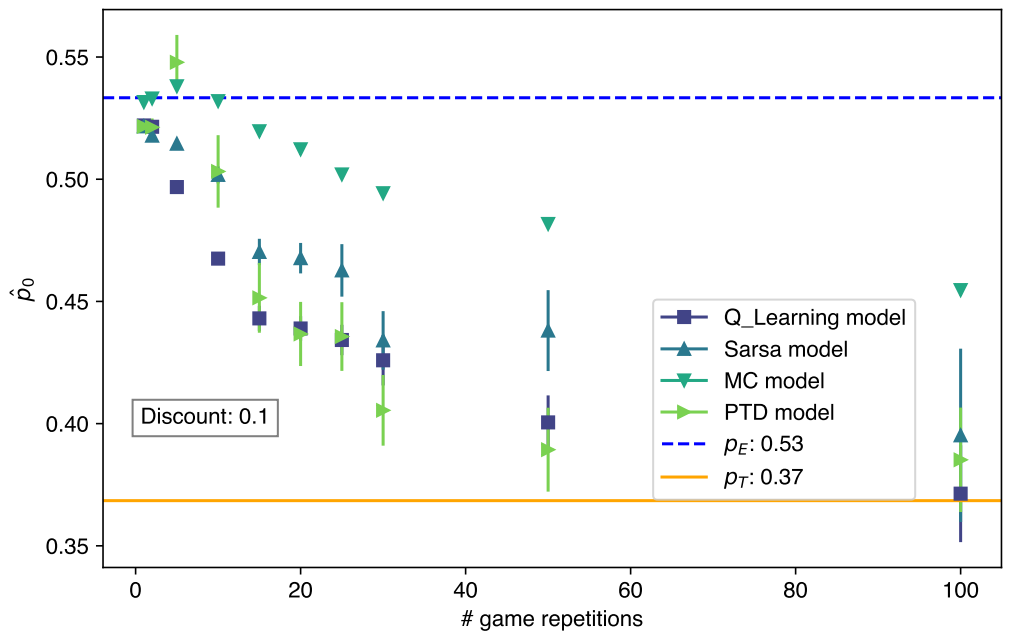

Figure 6. Adaptive response of reinforcement learning models to increasing game repetitions, illustrating a shift in preferred optimizer for the final policy. Our results show the change in the estimated policy inflection point ($\hat{p}_0$) as the number of decision repetitions grows. The dashed blue line represents the expected value estimate (p_E), where initial single-round training estimates of all models are clustered. As game repetitions increase, a clear trend emerges: models progressively abandon the expected value prediction and adapt towards a more risk-averse policy, aligning with the predictions based on average time growth (p_T). This shift towards risk aversion occurs rapidly across most models and intensifies with the number of repetitions.

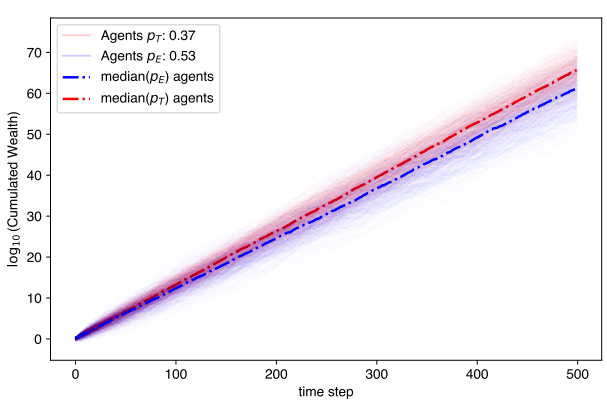

Figure 7. Simulation of 500 agents choosing between the risky or safe bet for 500 time steps. Agents are modelled using both the policy from our new method with optimization for average time growth (p_T, red curves) as for traditional RL agents optimized for expected values (p_E, blue curves). We evaluate the agents cumulated wealth at each time step on a log_{10}-scale on the vertical axis. For both groups of agents, the median of the agent's cumulated wealth is shown as an intermittent line on the graph. The advantage of optimizing time growth is several orders higher than the traditional results.

agent to reach an optimal policy. However, not every process experiences ergodicity breaking. Sometimes, we are not aware whether the context is ergodic or not. In other cases, the underlying process is known to be ergodic.

Our model should be resilient in finding the optimal parameter for optimising the process, regardless of the validity of the ergodic assumption. We can test the application of our approach by training our agents in the same experimental setup we introduced earlier but replacing the underlying path-dependent process with an ergodic one. As mentioned, an additive increment of

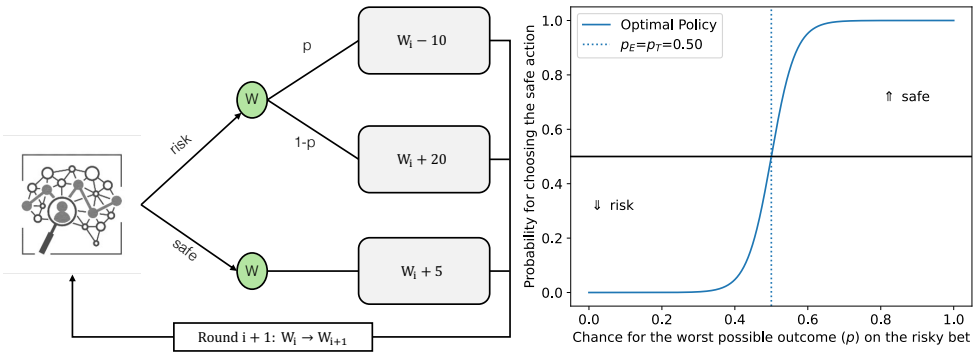

Figure 8. Adapted game layout for the training of the agents in an ergodic environment with additive increments on the agent's wealth. The left panel demonstrates the bets available at each round (i), impacting the wealth in the subsequent round (W_{i+1}). The right graph plots the agent policy as a function of the probability p for the worst outcome on the risky bet. The dynamic is ergodic, so the policy aligns with both the expected value (p_E) and time growth optimizer (p_T) as these provide the same prediction.

an agent's wealth provides an ergodic dynamic. The increments in our previous experiments were non-ergodic due to the multiplicative dynamic, but we can alter this by changing the multiplication of the wealth by an additive increment. For fixed increments, the underlying dynamic for how wealth changes is inherently ergodic [11,53]. The experimental setup of section 3(b) remains unchanged except for the update for the agents' wealth by additive increments, as illustrated in Figure 8. As the average time growth and expected value predictions are the same for these processes, i.e. $p_E = p_T$, we expect the agent not to deviate from its original prediction, regardless of the repetitions. We now train 20 agents over 2000 episodes with a discount factor ($\gamma = 0.8$) and decaying strategy on both the learning rate and exploration-exploitation trade-off (ϵ). The additive increments, represented by the rewards ($r_1 = +20, r_2 = -10, r_3 = +5$). The main results for each update rule are shown in Figure 9 and confirm that our training does not alter the optimizer for ergodic dynamics. Agents trained on a single round or multiple repetitions all result in the same optimal policies based on expected values. With repeated games, the path-dependency is introduced but does not change the process's optimal value due to the ergodicity of the additive increments. As agents do not change their policy over game repetitions, it shows our proposed method's applicability to ergodic processes.

(e) Time growth maximization: the Kelly criterion

In finance, the problem of portfolio management presents unique challenges due to its high path dependence and low signal-to-noise ratio [54]. Fortunately, the Kelly criterion [41] provides an optimal target for its portfolio allocation.

Although theoretically optimal, implementing the Kelly criterion in practice poses significant challenges for RL models. The high variability of real-world financial data and the difficulty of accurately estimating parameters like expected return and variance make it hard for RL agents to learn and generalize effectively. The continuous and dynamic nature of investment decisions compounds this complexity. RL solutions to these distinctive challenges typically fall into two categories. The first employs DRL to handle the continuous state space of investment fractions and integrate multi-type financial data [55]. The second type of solution exists at the value function level, which can be modified to address the challenges of financial returns [56]. However, determining or deriving an appropriate value function presents its computational challenges [55].

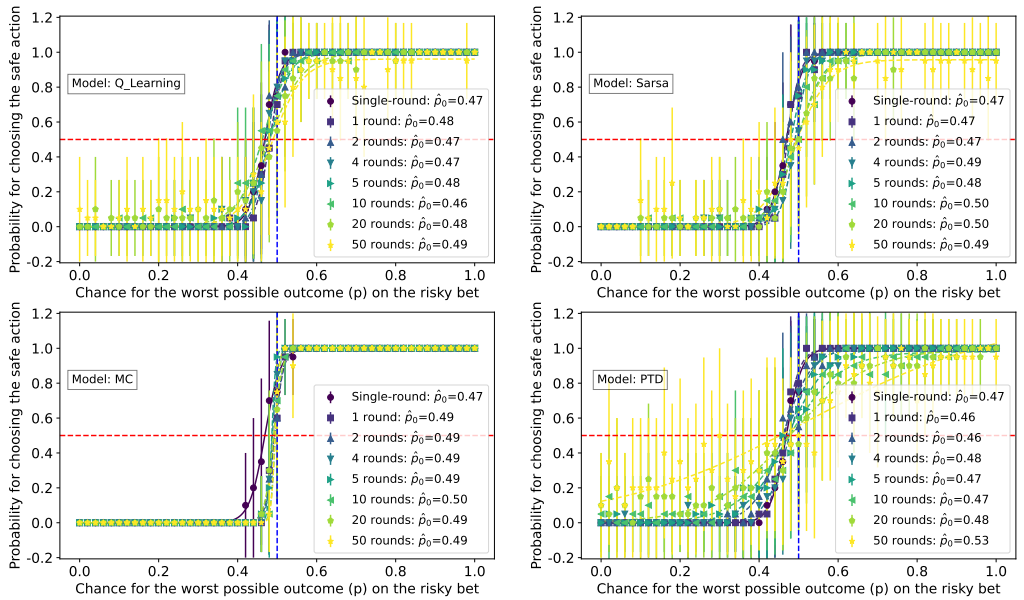

Figure 9. Visualization of the ergodic experiment to include multiple rounds of choices between safe and risky bets before policy updates with additive increments for the agent's wealth. Each figure compares the estimated inflection points ($\hat{p}_0$) of different models with both the theoretical prediction based on expected values (p_E) and on average time growths (p_T). As the number of decision rounds increases, the policies remain the same and all agents optimize their decisions based on predictions by the expected value of the risky bet ($p_E = p_T$). This behaviour is expected as the underlying dynamic is ergodic, and the average time growth and expected value result in identical predictions. Our result shows that our method can also safely be applied to ergodic dynamics and non-ergodic cases.

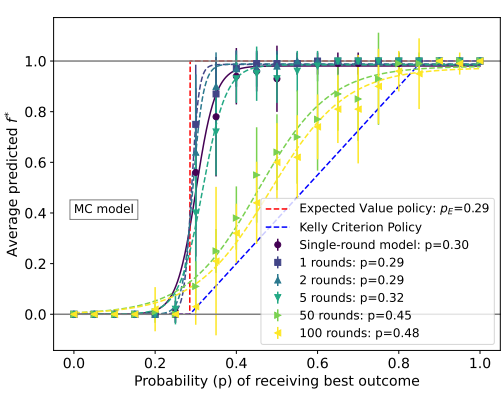

Figure 10. Policy results for agents trained on a portfolio management task. Agents must determine the optimal fraction of their wealth (f^*) to assign to a stock with a stochastic outcome determined by a binomial with probability p to have a multiplicative gain in wealth. The remainder of the wealth remains constant. The Kelly criterion determines the optimal fraction, shown in the figure by the blue dashed line. The alternative policy relies on expected values, as the red dashed line shows. Agents are trained using the Monte Carlo update rule and learn to accurately approximate the optimal policy (Kelly criterion) after subjecting them to repetitive cumulated training rounds. Due to the non-ergodic nature of the process, the agents learn to optimize towards the average time-growth prediction as the solution space becomes exponentially large for repeated decisions, in agreement with our predictions and the predictions by [11].

Consequently, Value functions are often determined by incorporating insights from financial market models or risk preferences [57]. For instance, to tackle the path-dependent nature of financial returns, it is common practice to use log returns rather than their actual returns as a value function. This approach echoes the techniques proposed by [40], who similarly use the log transform to combat non-ergodic increments. To expose an RL agent to a portfolio assignment task, we use a similar training setup as in the previous multiplicative experiment. However,

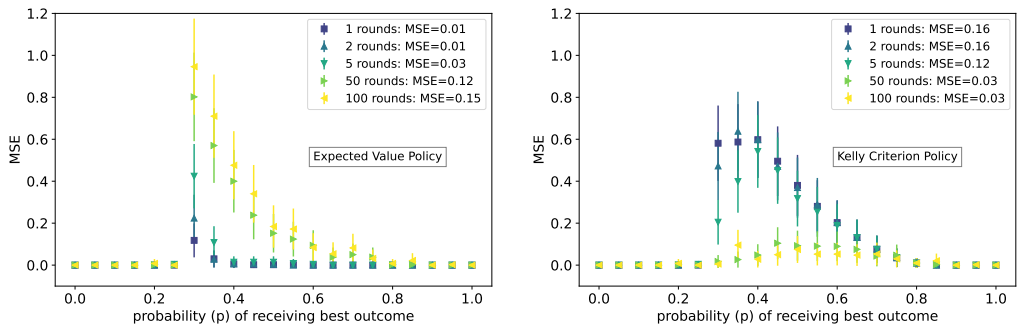

Figure 11. Mean squared error results for the different agents trained on the portfolio management compared to both an expected value optimization (left) or the optimal time growth from the Kelly criterion (right). Each entry represents the square error between the averaged agents policy and the expected value optimizer on the left-hand figure. The right-hand figure shows the same policy but for squared errors with respect to an optimal Kelly criterion. Mean Squared Errors (MSE) are shown in the legend for each training regime. The result confirms that the agents capture the optimal time growth for repeated rounds and produce a good approximation of the optimal policy, as determined by the Kelly criterion. Agents trained only on a single action selection cannot capture the time growth, resulting in a final policy as predicted by expected values.

agents are now trained to find the optimal fraction (f^*) of the total wealth (W) to invest in a bet with an underlying binomial distribution.

The Kelly criterion dictates the exact fraction of initial wealth to assign to the bet for an optimal gain, which aligns with our predicted growth maximization. The alternative policy is based on an expected value optimization, which states that an agent should not assign wealth to the bet until its expected value exceeds 1. From then on, the agent will wager all of its wealth immediately. We make these fractions a finite set to make the problem tractable for traditional RL update rules. As such, the agent has a fixed set of actions representing the fraction of wealth to assign to the bet ($[0, 0.2, 0.4, 0.6, 0.8, 1]$). This setup allows us to generalize our proposed methodology to a more complex state-action space. Nonetheless, we restrict the training to using the Monte Carlo update rule exclusively, as other traditional models need to be rigorously fine-tuned to handle the complex setup discussed in [55].

Figure 10 shows the resulting policy for an agent trained using the Monte Carlo update rule, with a risk bet where the invested wealth is increased by a factor of $r_1 = 3.0$, with a probability p. With probability $(1 - p)$, the agent will see its wealth multiplied by a factor $r_2 = 0.2$. The important difference here relies on the most likely value estimate in path-dependent multiplicative dynamics as discussed by [11]. When the space of all possible trajectories becomes sufficiently large, an agent will learn to optimize towards the most likely values. Due to the broken ergodicity, the most probable value no longer aligns with the expected value of the bet. In the context of EE, the prediction of the optimal fractions aligns with the maximization of the time growth as introduced by Kelly and as discussed by Breiman [58]. This is reflected in the results presented in section 3(b), which show how the agent's strategy changes from an almost perfect expected value-based strategy for single-round models to a near-optimal Kelly policy. The red (expected value) and blue (Kelly criterion) dashed curves show both pure strategies as references.

To measure how much the agent is changing from an expected value optimization towards the Kelly criterion, we show the square distance between the predictions from the agent and the two different optimization strategies. Figure 11 shows the difference between the agent's policy and a perfect expected value strategy on the left. The figure on the right shows the square distance between the agent's policy and the optimal policy, predicted by the Kelly criterion. We also show the total mean squared error (MSE) for the policies in these figures. Our results show that if an agent is not exposed to the path-dependent nature of financial data, the Kelly criterion

is not learned and we revert to optimize expected returns. However, by training the agents using our proposed method, the path dependence of the dynamic is introduced, allowing the agent to closely approximate the optimal strategy as dictated by the Kelly criterion.

It is important to point out that our method does not fix the inherent challenges tied to the Kelly criterion in a reinforcement learning context [55]. Increasingly large state-action spaces and the continuous nature of the Kelly criterion make it a typical study for Deep RL methods. Our results demonstrate that it can suffice to expose the agents to the underlying dynamic of an optimization problem rather than relying on transformations of the value function or deep reinforcement learning techniques to find a proper approximation.

(f) Multiple rounds with separate states

As a final result, we present an important consideration when modelling the decision between safe and risky actions and dynamics with inherent uncertainty from the experiment as introduced in section 2. Introducing an agent into a non-ergodic context requires the agent to capture the time dynamic of the underlying process. An alternative to our earlier approach uses additional states in the architecture, effectively extending the state space of the environment to capture each decision at an individual state. An agent hops from one state to the next by choosing either the safe or risky action and carrying along its updated wealth. This alternative architecture is computationally more tractable but introduces a subtle difference within its training. As each state now acts independently of the previous one, the time dependency is harder to uncover from the consecutive updates.

In Figure 12, we show that the agents can still learn path dependency when subject to our proposed training method. However, the change from an expected value optimizer towards the optimal policy based on average time growth is much slower. It is comparable to the Monte Carlo update rule results in the experiment reported in section 3(b). The prediction for the optimizers aligns with the discussion from [11], where the most-probable value deviates from the average as extreme events become increasingly less likely to be encountered for increasing game repetitions. This illustrates the power of introducing path dependencies but also shows the complexity of the problem posed in our initial setup.

4. Discussion

We show in Figure 3 traditional RL models develop policies based on expected values, as predicted and intended by RL theory. Nonetheless, when ergodicity is broken, our models need to find the optimal value for individual agents whose effects are poorly predicted by the expected value of the process. The experiments reported in our study include the time dynamic as a pivotal process in determining an individual's optimal policy. Our results in section 3(b) and 3(c) show that Q-Learning, SARSA, and the extra PTD model can capture this underlying effect of the non-ergodic context, optimizing towards average time growth predictions (p_T) rather than the group average (P_E), shown in Figure 6. We do this by subjecting the agents to multiple rounds of the same decision-making process within a single training episode.

Our analysis shows distinct behaviours among the different RL models. Traditional Q-learning and the PTD model adapt quickly to the changing context of the extended game lengths, demonstrating a good alignment with average time growth optimization (p_T). However, in the alternative architecture, section 3(f), the PTD model fails to provide informative policies. The adaption of the traditional algorithms seems insufficient to capture the underlying dynamics due to a lack of information on the level of the individual states. The Monte Carlo method replaces the temporal difference with an arithmetic average of the rewards in the agent's trajectory in each episode. This reflects more closely the notion of an ensemble average and results in a slow adaption of the non-ergodic contexts. Nonetheless, as the results in section 3(b) suggest, the Monte Carlo model also adapts to the path dependence of the dynamic. This change in optimal policy for the MC update rule stems from the description provided by Redner [11]. Moreover, It can be

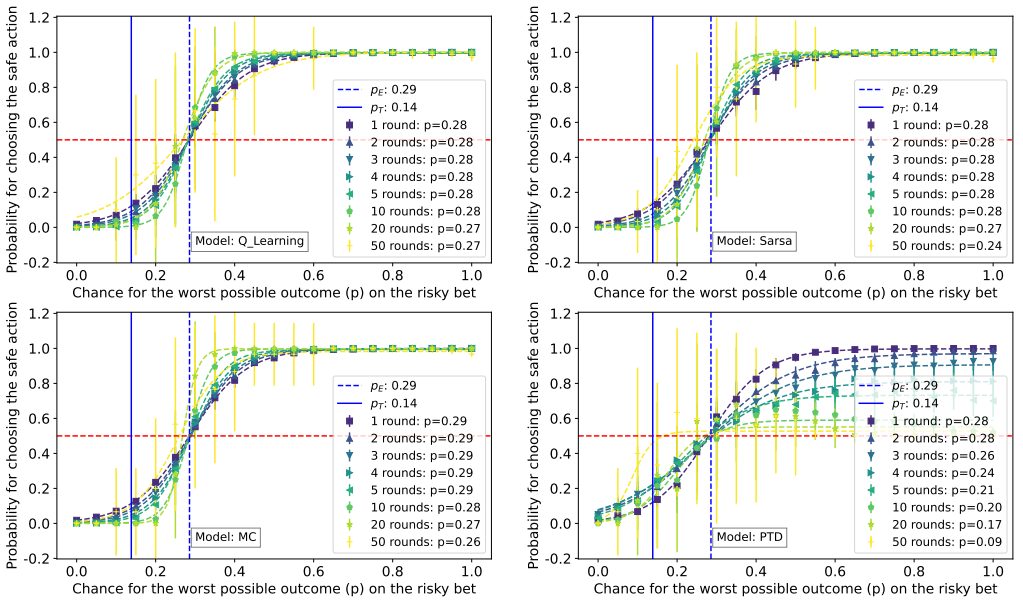

Figure 12. Policy results for models using the different update rules on the experiment reported in 3(b) but implemented with a different architecture of the state-action space. Each state represents one of the consecutive decisions in this experiment, with its own Q-values to be updated during the training. The average of all consecutive state action preferences determines an agent's final policy. A more detailed picture of the final policies for each state is shown in Appendix C(f) for the traditional update rules (Q-learning, Sarsa and Monte Carlo). These results show the complexity of the problem presented to the agent as the policy only slowly evolves from the ergodic prediction (p_E) towards the optimal policy (p_T). As shown in the bottom right figure, the PTD model, which only uses the final wealth, fails to provide informative policies for higher game repetitions in this training environment.

found in the experiment on the Kelly criterion, section 3(e) and the alternative state-dependent architecture experiment, section 3(f). For this last experiment, we see that the initial advantage of the TD models vanishes, and all models fail to capture the path dependence early on. At this point, the limited coverage of possible trajectories comes into play, and models behave similarly to the MC update rule to find the optimal value. The SARSA update rule is implemented with an online update and focuses on the actual trajectories of agents throughout one episode. The update is thus more limited to the agent's experience than to the optimal prospects. With increasing game repetitions, possible trajectories grow exponentially, resulting in less stable policies for this model. SARSA estimates, therefore, have a larger variance and are more sensitive to hyperparameter choices; see Appendix C(c).

In short, RL is constructed to handle consecutive decision-making processes. Still, it is fundamental to understand the different dynamics that can impact the optimal value. Our results highlight that the temporal difference method enables fast adaptation to the time dynamic within expected outcomes in non-ergodic contexts.

An alternative approach to interpreting the optimization of reinforcement learning models uses stochastic precedence as discussed in [59]. With our simulation indicating the impact of path dependence, we show that verifying the ergodic assumption is essential to train agents for optimal policies. The resulting predictions are optimal for individual agents and for the ensemble when the agents optimize for average time growth in their policies (p_T).

5. Conclusion

We have shown that the standard implementation of RL algorithms cannot capture the behaviour of non-ergodic contexts. With the explicit introduction of the time-sensitive dynamics, we enable a number of traditional RL models to learn better optimizers for the underlying dynamics, the average time growth (g). We show that TD-based models can capture this optimization early on, depending on the hyperparameter selection. However, for a more extended horizon, we notice that even the MC model, using the mean reward explicitly in its update rule, can capture the time-sensitive dynamics. This aligns with the idea that agents' mean or ensemble average will also converge to the time-optimal value over long time horizons. We further exploit this feature in an experiment on portfolio management where agents find a close approximation to the optimal strategy determined by the Kelly criterion. Moreover, our results show that the proposed method for training agents can also be applied in ergodic contexts without loss of generality.

As such, we present a novel idea for training RL agents in non-ergodic contexts without transformations of the value function and rewards. Ultimately, we highlight the importance of considering the ergodic assumption before designing experiments and applications using traditional RL models.

Appendix

A. Softmax function

The transformation of Q-values to reflect preference among the two actions is carried out by a normalized softmax function

$$s(Q_a) = \frac{\exp\left[Q_a - \max\limits_{i\in\mathbb{A}}(Q_i)\right]}{\sum\limits_{k\in\mathbb{A}} \exp\left[Q_k - \max\limits_{i\in\mathbb{A}}(Q_i)\right]}. \tag{A 1}$$

We use a shift in the Q-values by subtracting the maximum value of the Q-values for both actions ($Q_a - \max(Q_i)$) before applying the softmax function itself. The reason is twofold: firstly, the Q-values can grow exponentially after training, even with small rewards. These values are computationally hard and, therefore, need rescaling. Second, the shift scales the difference between two values to a unified starting value as the value for the maximum Q-value always becomes zero after rescaling.

B. Model parameters

Throughout the experiment, we used a single selection of model parameters and appropriately tuned hyperparameters. The set of this selection is shown in Table 2. For reproducibility, one can use the same choice of parameters as reported in this table. However, the results can be reproduced to different reward vectors described throughout the paper and hyperparameter discussion, as reported in Appendix C. Figure 13 shows an example of the results for one of such tests where the result is reflected by a combination of hyperparameters as selected for the model training in the experiment. We show each model converges to policies that reflect the expected value optimizer (p_E) for various sets of randomly selected reward vectors. This shows the generalizability of our model and claims for RL as an expected value optimizer.

C. Hyperparameter tuning

As in many ML applications, reinforcement learning is subject to a set of hyperparameters. To select proper regimes for training the agents, we sweep a set of hyperparameters to find any regimes which result in poor training results. These training results are reflected by policies that

Parameter	Value	Parameter	Value
r_1	0.5	#Episodes	2500
r_2	2.0	#Runs	50
r_3	1.2	Discount γ	0.1
α-strategy	Decaying	ϵ-greedy strategy	Decaying
α initial value	0.9	ϵ-greedy initial value	0.3
α threshold value	0.01	ϵ-greedy threshold value	0.02
α decay-rate	0.995	ϵ-greedy decay-rate	0.998

Table 2. Parameter values for the training of different models throughout the experiment

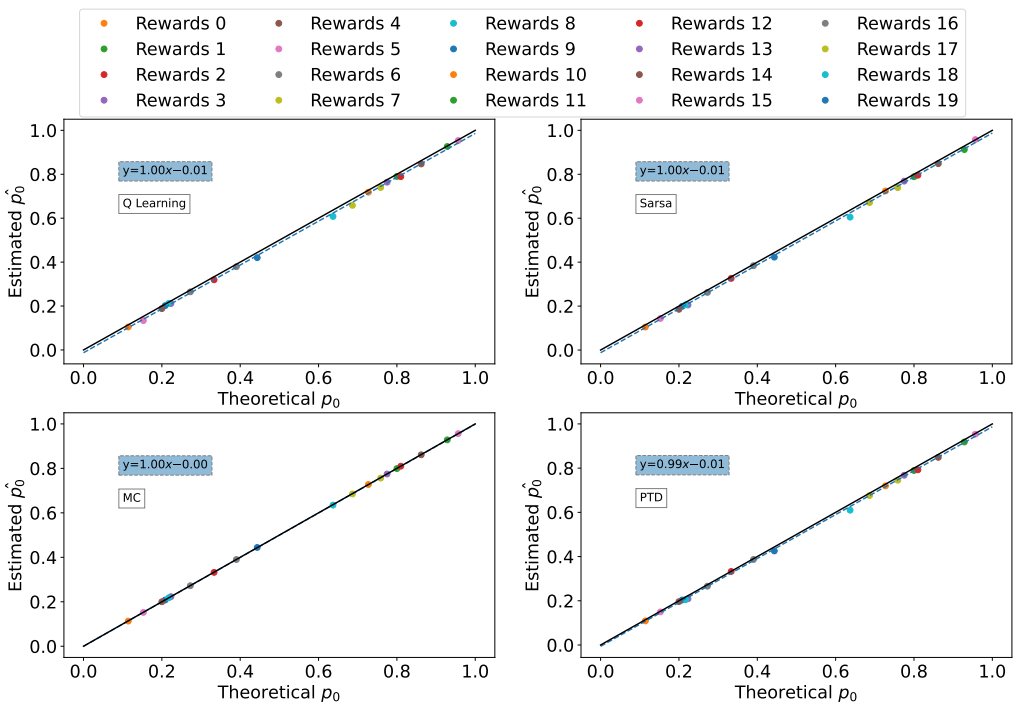

Figure 13. An evaluation displaying the correlation between the predicted (p_E) and the estimated inflection points ($\hat{p}_0$) for different reward vector configurations and hyperparameter combinations in a single-round experiment setting. Each subplot represents a different model—Q Learning, Sarsa, Monte Carlo (MC), and Pure Temporal Difference (PTD)—and the regression line fitted to the valid training results, with the theoretical first bisector ($y = x$) as a reference. Optimal hyperparameters yield estimates closely aligned with the theoretical values, as indicated by regression lines that nearly coincide with the bisector.

are not able to find the inflection point (p_0) as dictated by the expected wealth (p_E) within a reasonable number of training episodes.

(a) Policy stability

The stability of the final policies of the trained agents is crucial. In Figure 14 we compare the normalized difference between the model estimates ($\hat{p}_0$) and the expected value prediction (p_E). This value can be calculated throughout the training process to evaluate the saturation of the different models for a stable policy. Our results show a clear saturation for all models with $\hat{p}_0$ approaching the expected value prediction p_E.

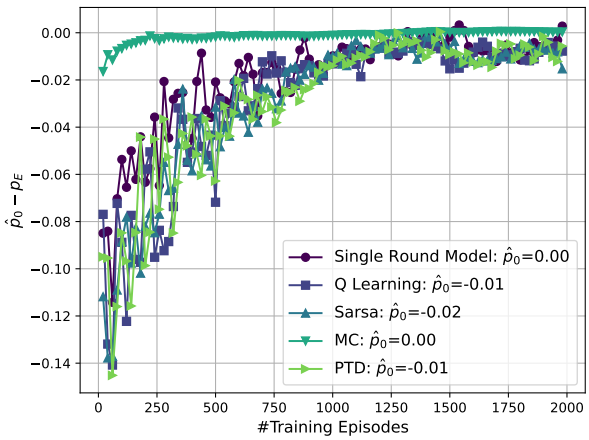

Figure 14. Convergence of reinforcement learning models towards stable policy estimates during training. These results illustrate the normalized difference between the expected value prediction (p_E) and the empirically estimated inflection point ($\hat{p}_0$) over 2000 training episodes. Convergence to zero means the models' policies accurately optimize for expected values. These results confirm that each model effectively functions as an expected value estimator. Each model is evaluated over 20 agents and reflects the average policy of this set of agents.

(b) Decay strategies

Central to the stability of policies are the hyperparameters that determine how the learning rate (α) and exploration-exploitation trade-off (ϵ) behave over the extent of our training. At the start of the training, we require agents to discover as much of the parameter space as possible but decrease the magnitude of updates (α) and deviations from policies (ϵ) near the end of the training episodes. Our hyperparameter sweeps on different strategies for both the learning rate and exploration-exploitation parameter, including a constant parameter regime and exponential decay strategy. For both parameters, a constant parameter choice showed detrimental consequences on the stability of the policies, independent of the choice for a number of training episodes, as shown in Figure 15. An exponential decay with decay rate (r_t) and initial value (α_0) resulted in stable policies, with an adaptive strategy depending on the number of episodes and a minimum threshold (α_{min}),

$$\alpha = \min\left(\alpha_{min}, \alpha_0 * r^t\right) \tag{A 1}$$

The decay strategy on ϵ is determined similarly to the learning rate decay, with free choice for the decay rate (r_t). Stable policies for a finite number of training episodes required an exponential decay strategy as indicated by the convergence between the estimated inflection point (p_0) and the theoretical value, represented by the first bisector in the figure.

(c) Discount

Discounting (γ) is one of the fundamental hyperparameters for temporal difference RL models. The discount determines how much focus is put on future rewards in the agent's update rule. Higher values emphasise future rewards, and low values focus on a greedier strategy for immediate rewards. We trained our models with identical hyperparameters and only swept the discount factor with similar results. For the specific reward vector used throughout the study ($(r_1, r_2, r_3) = (2, 0.5, 1.2)$), we show the results for the remaining discount choices ($\gamma \in \{0.1, 0.2, 0.8, 0.9\}$). The results for each choice of discounting are shown in Figure 16. All models agree with our conclusion for the example presented throughout the study, where policies for all models converge to optimize towards average time growth optimization rather than expected value estimates. One model stands out for its variability in its resulting policy for many game repetitions. The SARSA update rule only uses its past knowledge of the solution space and is thus more restricted to its ideal prospects. This results in high policy variance and a large difference among the different choices of discount parameters (γ) during training.

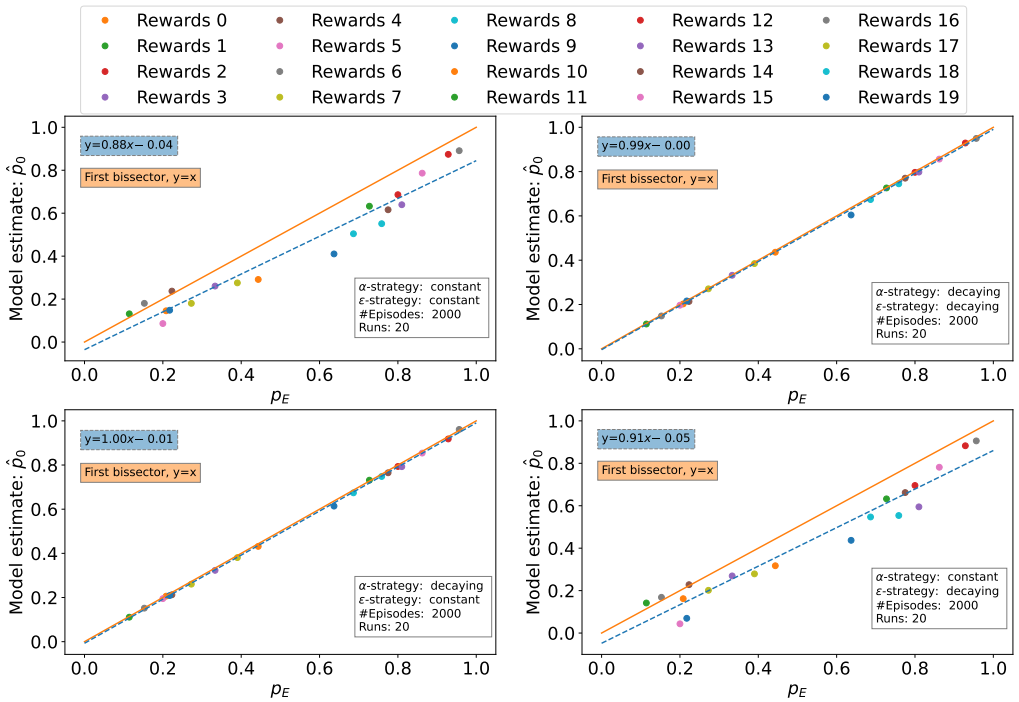

Figure 15. Example of hyperparameter sweep for different decay strategies on learning rates (α) and exploration-exploitation trade-off (ϵ). Constant strategies imply no decay on the parameters with detrimental results for stable policies on a random selection of rewards (r_1, r_2, r_3). An exponential decay can include the number of training episodes as an input to determine the decay rate ($r_t = \frac{a}{N}$). The variable a determines the steepness of the decay with respect to the number of training episodes (N).

(d) Policy evaluation in normalized g-space

We show a different visualisation of the results obtained by the experiment in section 3(b) to accentuate the agent's switch from ensemble averages to time averages. In the left column of Figure 17, we transform the horizontal axis (the probability (p)) to the normalised expected value of the risk bet so that the expected value of both actions is equal at 1. In the right column of Figure 17, the horizontal axis reflects the normalised time-average growth rate so that the time averages of both prospects are equal at 1. The evolution of the policies now shows how the preferences for the agent shift from optimising ensemble averages to time averages as the amount of game repetition increases.

(e) Simulation with normal distribution on parameter p

To highlight the difference in strategy, a selection of parameter p can have different underlying distributions. Next to the result presented in Figure 7 we can select from a normal distribution ($p \in N(\mu, \sigma^2)$). Here, we determine the expectation value from the optimal values for time-growth (p_T) and expected value (p_E) optimizers as

$$\mu = \frac{p_E + p_T}{2}. \tag{A 2}$$

The standard deviation is then

$$\sigma^2 = \left(\frac{p_E - p_T}{2}\right)^2 \tag{A 3}$$

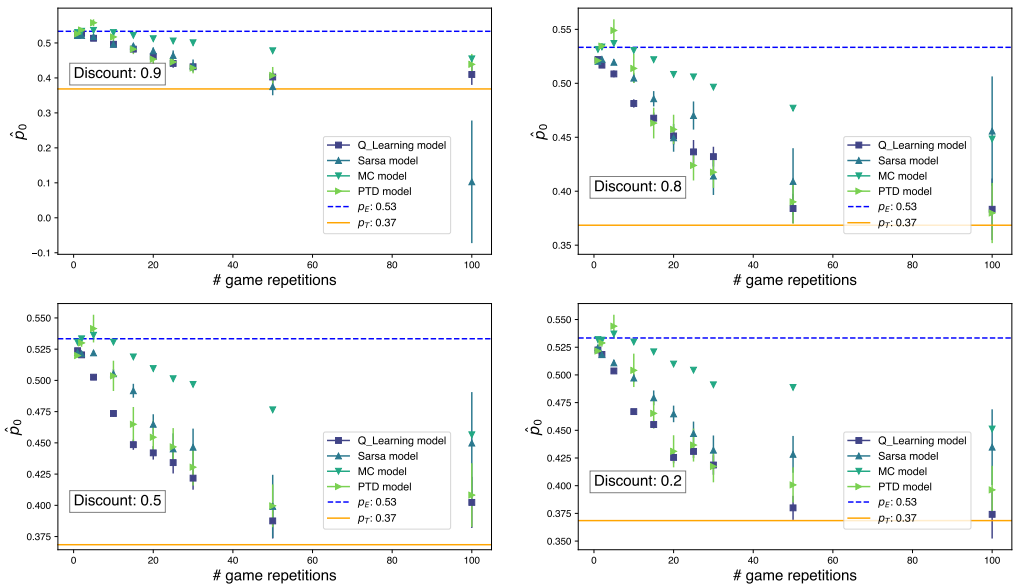

Figure 16. Overview of results for training agents identical to the studies parameter selection but different discounting values. The discount varies from low values in the top left (a: $\gamma = 0.1$) and the top right (b: $\gamma = 0.2$) to high values at the bottom, (c: $\gamma = 0.8$) and (d: $\gamma = 0, 9$). The results agree with our main result regarding this variety of discounting factors, even for extreme values. Our agents will learn to optimize their policy towards the average time growth optimizer, independent of the discounting factor.

This allows us to focus the simulation on the regime with different policies between the two optimizers, as shown in Figure 18.

(f) Alternative game architecture: final policies for each state

As discussed in section 3(f) we implemented an alternative architecture for our agents to optimize the decision between safe and risk bets. As an agent now has as many states as there are game repetitions, the final policy needs to be determined from each of these individual states. Figure 19 shows how the distribution of policy for each of the individual states changes when we subject the agent to more game repetitions. For a single round in the game, there is no policy distribution as there is only one state. As we start increasing the number of game repetitions our results indicate that the agent does not initially capture the underlying temporal dynamic. It is only for large repetitions and exponentially large state-space that the agents start to optimize their policy towards the average time growth estimate (p_T), as predicted by Redner [11].

Acknowledgements. This research received funding from the Flemish Government (AI Research Program

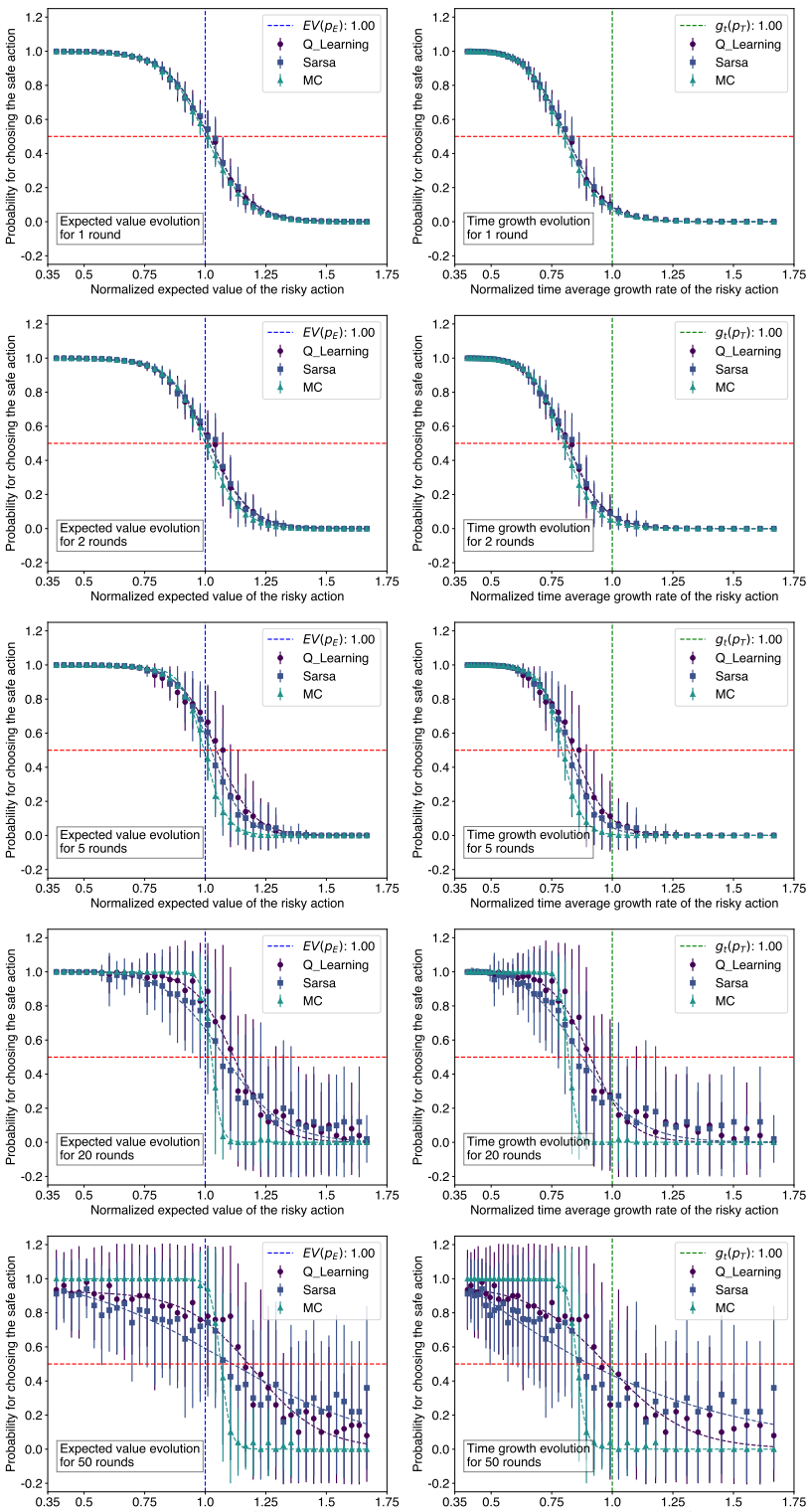

Figure 17. Transformed policy evolution of the agents trained on repeated multiplicative bets. To describe the optimisers themselves, we show a transformation of the probability (p), the chance to get the worst outcome of the risky action. In the figures on the left, we show the transformation to an expected value evolution on the horizontal axis. The figures on the right show the policy as a function of the average time growth rate of the risky action. Our results show that this evolution can explicitly illustrate the maximization of the growth rate.

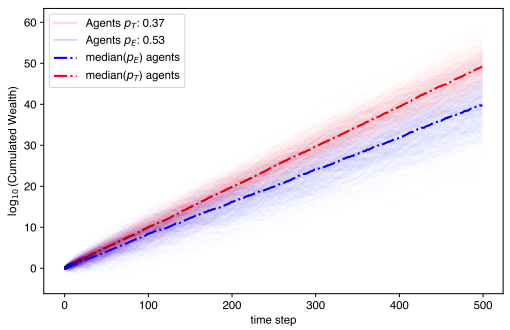

Figure 18. Simulation of 500 agents choosing between the risky or safe bet for 500 time steps. Agents are modelled using both the policy from our new method with optimization for average time growth (p_T, red curves) as for traditional RL agents optimized for expected values (p_E, blue curves). We evaluate the agents cumulated wealth at each time step on a log_{10}-scale on the vertical axis. For both groups of agents, the median of the agent's cumulated wealth is shown as an intermittent line on the graph. The advantage of optimizing time growth is several orders higher than the traditional results. The selection of values for the parameter p is pulled from a normal distribution ($N(\mu, \sigma^2)$), with the mean and variance derived from the two optimizers (p_E, p_T).

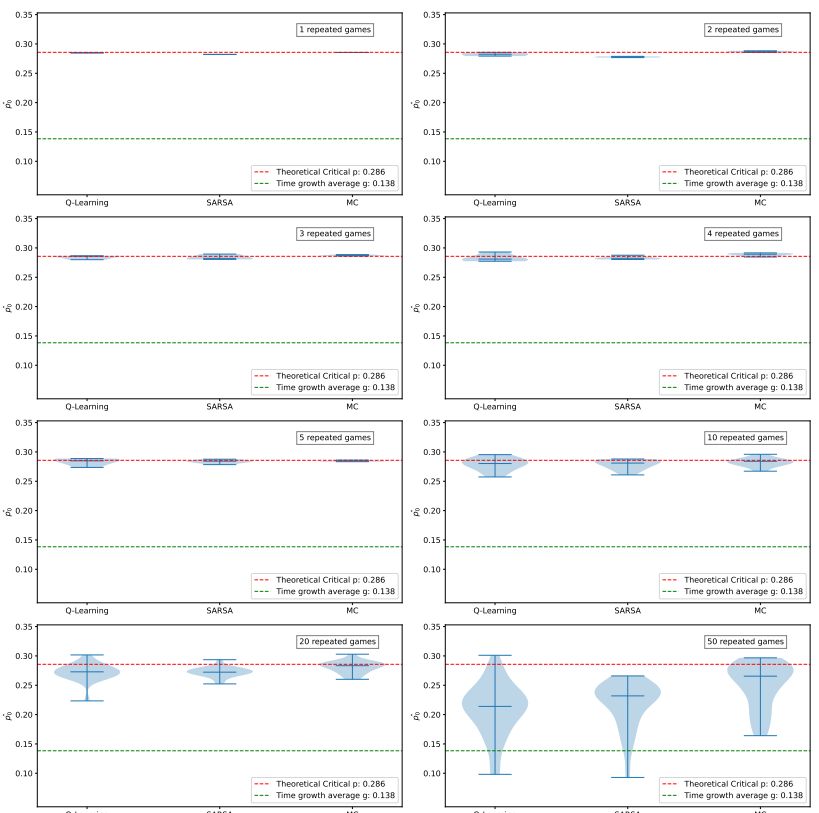

Figure 19. Distribution of the inflection points for each of the individual states within a trained agent's policy for the extended game architecture for the Q-learning, SARSA and Monte-Carlo models. Each inflection point shows the policy for an agent at a consecutive state for the decision between the risk or safe action. Although the evolution is less pronounced than for the experiment discussed in section 3(b), the extended architecture allows the agent to learn the temporal dynamic on larger repeated game rounds. The results highlight the complex nature of the problem as the agents struggle to accurately determine the optimal value (p_T) for all or even a majority of states.

References

1. Yearsley JM, Pothos EM. 2016 Zeno's paradox in decision-making. *Proceedings of the Royal Society B: Biological Sciences* **283**, 20160291.
2. Calder M, Craig C, Culley D, De Cani R, Donnelly CA, Douglas R, Edmonds B, Gascoigne J, Gilbert N, Hargrove C et al.. 2018 Computational modelling for decision-making: where, why, what, who and how. *Royal Society open science* **5**, 172096.
3. Nielsen MA. 2015 *Neural networks and deep learning* vol. 25. Determination press San Francisco, CA, USA.
4. Prince SJ. 2023 *Understanding Deep Learning*. MIT press.
5. Sutton RS, Barto AG. 2018 *Reinforcement learning: An introduction*. MIT press.
6. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M et al.. 2016 Mastering the game of Go with deep neural networks and tree search. *nature* **529**, 484–489.
7. Punjani A, Abbeel P. 2015 Deep learning helicopter dynamics models. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* pp. 3223–3230. IEEE.
8. Kiran BR, Sobh I, Talpaert V, Mannion P, Sallab AAA, Yogamani S, Pérez P. 2022 Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems* **23**, 4909–4926. (10.1109/TITS.2021.3054625)
9. Chapman M, Xu L, Lapeyrolerie M, Boettiger C. 2023 Bridging adaptive management and reinforcement learning for more robust decisions. *Philosophical Transactions of the Royal Society B* **378**, 20220195.
10. Peters O. 2019 The ergodicity problem in economics. *Nature Physics* **15**, 1216–1221.
11. Redner S. 1990 Random multiplicative processes: An elementary tutorial. *Am. J. Phys* **58**, 267–273.
12. Bellman R. 1956 Dynamic programming and Lagrange multipliers. *Proceedings of the National Academy of Sciences* **42**, 767–769.
13. Bellman R. 1958 Dynamic programming and stochastic control processes. *Information and control* **1**, 228–239.
14. Watkins CJCH. 1989 *Learning from delayed rewards*. PhD thesis King's College, Cambridge United Kingdom.
15. Watkins CJ, Dayan P. 1992 Q-learning. *Machine learning* **8**, 279–292.
16. Bellemare MG, Dabney W, Munos R. 2017 A distributional perspective on reinforcement learning. In *International conference on machine learning* pp. 449–458. PMLR.
17. Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T et al.. 2018 A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **362**, 1140–1144.
18. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G et al.. 2015 Human-level control through deep reinforcement learning. *nature* **518**, 529–533.
19. Kaiser L, Babaeizadeh M, Milos P, Osinski B, Campbell RH, Czechowski K, Erhan D, Finn C, Kozakowski P, Levine S et al.. 2019 Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*.
20. Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Žídek A, Potapenko A et al.. 2021 Highly accurate protein structure prediction with AlphaFold. *Nature* **596**, 583–589.
21. Bai Y, Jones A, Ndousse K, Askell A, Chen A, DasSarma N, Drain D, Fort S, Ganguli D, Henighan T et al.. 2022 Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
22. Ibarz B, Leike J, Pohlen T, Irving G, Legg S, Amodei D. 2018 Reward learning from human preferences and demonstrations in Atari. In *Advances in Neural Information Processing Systems*.
23. Christiano PF, Leike J, Brown T, Martic M, Legg S, Amodei D. 2017 Deep Reinforcement Learning from Human Preferences. In *Advances in Neural Information Processing Systems*.
24. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M. 2013 Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
25. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. 2017 Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine* **34**, 26–38.
26. Dong H, Dong H, Ding Z, Zhang S, Chang T. 2020 *Deep Reinforcement Learning*. Springer.

27. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. 2017 Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
28. Schulman J, Moritz P, Levine S, Jordan M, Abbeel P. 2015 High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.
29. Jin C, Yang Z, Wang Z, Jordan MI. 2020 Provably efficient reinforcement learning with linear function approximation. In *Conference on learning theory* pp. 2137–2143. PMLR.
30. Chen S, Li Y. 2020 An overview of robust reinforcement learning. In *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)* pp. 1–6. IEEE.
31. Lockwood O, Si M. 2022 A review of uncertainty for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* vol. 18 pp. 155–162.
32. Gu S, Yang L, Du Y, Chen G, Walter F, Wang J, Knoll A. 2022 A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*.
33. Amani S, Thrampoulidis C, Yang L. 2021 Safe reinforcement learning with linear function approximation. In *International Conference on Machine Learning* pp. 243–253. PMLR.
34. Wang K, Mu C, Ni Z, Liu D. 2023 Safe reinforcement learning and adaptive optimal control with applications to obstacle avoidance problem. *IEEE Transactions on Automation Science and Engineering*.
35. Zhao W, He T, Chen R, Wei T, Liu C. 2023 State-wise safe reinforcement learning: A survey. *arXiv preprint arXiv:2302.03122*.
36. Vanhoyweghen A, Ginis V. 2023 Human decision-making in a non-ergodic additive environment. *Proceedings of the Royal Society A* **479**, 20230544.
37. Heger M. 1994 Consideration of risk in reinforcement learning. In *Machine Learning Proceedings 1994* , . Elsevier.
38. Greenberg I, Chow Y, Ghavamzadeh M, Mannor S. 2022 Efficient risk-averse reinforcement learning. *Advances in Neural Information Processing Systems* **35**, 32639–32652.
39. Mihatsch O, Neuneier R. 2002 Risk-sensitive reinforcement learning. *Machine learning* **49**, 267–290.
40. Baumann D, Noorani E, Price J, Peters O, Connaughton C, Schön TB. 2023 Non-ergodicity in reinforcement learning: robustness via ergodicity transformations. *arXiv preprint arXiv:2310.11335*.
41. Kelly JL. 1956 A new interpretation of information rate. *the bell system technical journal* **35**, 917–926.
42. Sarkissian J. 2020 Quantum coupled-wave theory of price formation in financial markets: Price measurement, dynamics and ergodicity. *Physica A: Statistical Mechanics and its Applications* **554**, 124300.
43. Peters O, Adamou A. 2022 The ergodicity solution of the cooperation puzzle. *Philosophical Transactions of the Royal Society A* **380**, 20200425.
44. Vilk O, Orchan Y, Charter M, Ganot N, Toledo S, Nathan R, Assaf M. 2022 Ergodicity breaking in area-restricted search of avian predators. *Physical Review X* **12**, 031005.
45. Nosonovsky M, Roy P. 2020 Allometric scaling law and ergodicity breaking in the vascular system. *Microfluidics and Nanofluidics* **24**, 53.
46. Kelty-Stephen DG, Mangalam M. 2022 Fractal and multifractal descriptors restore ergodicity broken by non-Gaussianity in time series. *Chaos, Solitons & Fractals* **163**, 112568.
47. Shityakov S, Kravtsov V, Skorb EV, Nosonovsky M. 2023 Ergodicity Breaking and Self-Destruction of Cancer Cells by Induced Genome Chaos. *Entropy* **26**, 37.
48. Mangalam M, Kelty-Stephen DG. 2022 Ergodic descriptors of non-ergodic stochastic processes. *Journal of the Royal Society Interface* **19**, 20220095.
49. Chung F, Handjani S, Jungreis D. 2003 Generalizations of Polya's urn problem. *Annals of combinatorics* **7**, 141–153.
50. Hayhoe M, Alajaji F, Gharesifard B. 2017 A Polya urn-based model for epidemics on networks. In *2017 American Control Conference (ACC)* pp. 358–363. IEEE.
51. Vanhoyweghen A, Verbeken B, Macharis C, Ginis V. 2022 The influence of ergodicity on risk affinity of timed and non-timed respondents. *Scientific Reports* **12**, 3744.
52. Meder D, Rabe F, Morville T, Madsen KH, Koudahl MT, Dolan RJ, Siebner HR, Hulme OJ. 2021 Ergodicity-breaking reveals time optimal decision making in humans. *PLOS Computational Biology* **17**, e1009217.
53. Vanhoyweghen A. 2024 *Ergodicity & Human Decision Making*. PhD thesis Vrije Universiteit Brussel.

54. Alscher A, Brauer M. 2015 Path dependence in the evolution of the business portfolio configuration of large multi-business firms. *International Studies of Management & Organization* **45**, 287–318.
55. Aboussalah AM, Xu Z, Lee CG. 2022 What is the value of the cross-sectional approach to deep reinforcement learning?. *Quantitative Finance* **22**, 1091–1111.
56. Jiang R, Saunders D, Weng C. 2022 The reinforcement learning Kelly strategy. *Quantitative Finance* **22**, 1445–1464.
57. Moody J, Saffell M. 2001 Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* **12**, 875–889.
58. Breiman L et al.. 1961 Optimal gambling systems for favorable games. *The Kelly Capital Growth Investment Criterion* pp. 47–60.
59. Price JR. 2023 *The Dynamics of Decision Making with Uncertain Outcomes*. PhD thesis University of Warwick.