CEIS TOR VERGATA Research Papers

Università di Roma Tor Vergata

ISSN 2610-931X

CEIS Tor Vergata

RESEARCH PAPER SERIES

Vol. 22, Issue 6, No. 586 – December 2024

A Reinforcement Learning Algorithm For Option Hedging

Federico Giorgi, Stefano Herzel, Paolo Pigato

A Reinforcement Learning Algorithm For Option Hedging

Federico Giorgi Stefano Herzel
Paolo Pigato
Department of Economics and Finance
University of Rome "Tor Vergata"

Abstract. We propose an algorithm, based on Reinforcement Learning, to hedge the payoff on a European call option. The algorithm is first tested in a model where the problem has a well known analytic solution, so that we can compare the strategy obtained by the algorithm to the theoretical optimal one. In a more realistic case, considering transaction costs, the algorithm outperforms the standard delta hedging strategy.

Keywords: Reinforcement Learning; Dynamic Strategies; Risk management

1 Introduction

The hedging of European option payoffs is among the most extensively studied topics in quantitative finance. This problem arises when a trader takes a position—long or short—in an option and seeks to hedge it using a dynamic strategy involving the underlying asset. Under the assumptions of the Black-Scholes model, this challenge is addressed through the "Delta Hedging" strategy. This strategy involves holding an amount of the underlying asset equal to the option's "Delta," which measures the sensitivity of the option's price to changes in the underlying asset. Mathematically, Delta is computed as the first derivative of the option price with respect to the price of the underlying asset.

The Black-Scholes model assumes that the price of the underlying follows a geometric Brownian motion and that trading occurs continuously and without transaction costs. These assumptions allow for the derivation of closed-form option prices and provide hedging strategies. However, in real financial markets, these assumptions often do not hold. Market frictions such as transaction costs, discrete trading times, and deviations from log-normal price distributions complicate the implementation of Delta Hedging. Despite these limitations, Delta Hedging remains a cornerstone of practical financial risk management due to its simplicity and effectiveness in many scenarios.

In this paper, we explore the hedging problem in a more general and realistic setting than the one assumed by the Black-Scholes model. Specifically, we address the challenges of Delta Hedging in markets where trading occurs at discrete intervals rather than continuously and where transaction costs are significant.

To do so, we propose an approach based on Reinforcement Learning (RL). RL is a branch of machine learning where an agent interacts with an environment over time, observing its state and taking actions to maximize cumulative rewards. At each time step, the agent observes a new state and receives feedback in the form of a reward, which guides its future actions. RL algorithms train agents by simulating a large number of "episodes," enabling them to improve their understanding of the environment and refine their strategies through trial and error. Seminal works on RL, such as [Sutton and Barto, 2018], provide the foundation for applying these algorithms to a wide range of dynamic decision-making problems.

Recent applications of RL in finance have demonstrated its potential to address complex optimization problems. For instance, [Kolm and Ritter, 2019] and [Giorgi et al., 2024], apply RL to dynamic portfolio optimization under transaction costs, while [Buehler et al., 2019] and [Cao et al., 2019] leverage RL to develop optimal hedging strategies for derivatives. [Vittori et al., 2020] extend these approaches by combining RL with neural networks to handle high-dimensional state spaces.

We present an RL-based algorithm for Delta Hedging, in discrete-time, with transaction costs. The proposed algorithm combines the SARSA (State-Action-Reward-State-Action) algorithm with a neural network to estimate the value function. This combination allows the model to handle a continuous set of states (e.g., current holdings, market realizations) and actions (e.g., the quantity of shares to trade). The value function is iteratively refined through batches of simulated episodes. Initially, the agent follows its current policy to make decisions, but its performance improves over time as it incorporates new information from simulated rewards. RL algorithms, while powerful, are not guaranteed to perform optimally in all instances. To validate the effectiveness of our approach, we conduct comparative tests in scenarios where the optimal solution is known, to assess whether the algorithm produces accurate and robust results.

The remainder of this paper is organized as follows. Section 2 provides a detailed description of the hedging problem, including the key challenges associated with discrete-time trading and transaction costs. Section 3 introduces the fundamental concepts of Reinforcement Learning relevant to our approach. In Section 4, we present the algorithm and discuss its implementation. In Section 5 we evaluate its performance through numerical experiments. Finally, Section 6 concludes with a summary of findings and directions for future research.

2 The hedging problem

We analyze the problem faced by a trader who sells the contingent claim at time t_0 for a price P_0 and subsequently adopts a dynamic strategy to cover the exposure to the payoff of

the claim P_T at time T. We consider a market where it is possible to trade a single risky asset, whose price at time t is denoted by S_t. Let n_k represent the units of the underlying held in the portfolio at time t_k, and let c_k denote the cost incurred at time t_k to adjust the portfolio from n_{k-1} to n_k (this cost can be negative, reflecting a gain). Assuming that hedging occurs at discrete times $t_0, t_1, \ldots, t_{N-1}$, with $t_{N-1} < T$, the total cost of the strategy is given by $\sum_{k=0}^{N-1} c_k$, while the liquidation value of the final position in the asset is $n_{N-1}S_T$. Thus, the final profit and loss (P&L) of the trader's position at time T is:

$$H_T = -P_T - \sum_{k=0}^{N-1} c_k + n_{N-1}S_T + P_0. \tag{1}$$

The goal of the hedging strategy is to minimize the risk of H_T, specifically by minimizing its variance. A contingent claim is said to be "replicable" if a strategy exists that reduces the variance of H_T to zero. Markets in which any contingent claim can be perfectly replicated are referred to as "complete."

The completeness of a market model depends on the dynamics of the price process S_t and the cost functions c_k. In the Black-Scholes-Merton (BSM) model [Black and Scholes, 1973], [Merton, 1973], the dynamics of the underlying asset are described by:

$$dS_t = \mu S_t \, dt + \sigma S_t \, dW_t, \tag{2}$$

where μ is the drift, σ is the volatility, and W_t is a standard Brownian motion. Additionally, the model assumes the existence of a risk-free asset with continuously compounded interest rate r, continuous-time trading, and the absence of transaction costs. Under these conditions, any contingent claim is replicable.

We focus on a European Call option, a contingent claim with a payoff:

$$P_T = \max(S_T - K, 0), \tag{3}$$

where K is the strike price and T is the maturity. The objective is to determine a dynamic strategy that adjusts n_k at each time step to replicate the payoff in (3). Since P_T is an increasing function of S_T, the hedging strategy typically involves holding positive amounts of the underlying asset.

In the BSM model, the optimal hedging strategy is determined by the option's "Delta," the first derivative of the option's no-arbitrage price with respect to the underlying asset. The Delta is computed as:

$$n_t = \Phi\left(\frac{\ln(S_t/K) + \tau\left(r + \frac{\sigma^2}{2}\right)}{\sigma\sqrt{\tau}}\right), \tag{4}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution, and $\tau = T - t$ is the time to maturity. This Delta hedging strategy achieves perfect replication

in the BSM framework, ensuring $H_T = 0$ almost surely, provided that P_0 is the no-arbitrage price and the strategy is executed continuously in time.

However, in practice, perfect hedging is impossible when trading is restricted to discrete times $t_k \in \{t_0, \ldots, t_{N-1}\}$. Under these conditions, the Delta hedging strategy introduces replication errors, and the variance of the error depends on the time discretization and the "Gamma" of the option, the second derivative of the option price with respect to the underlying asset (see [Angelini et al., 2009] and [Angelini and Herzel, 2015] for details).

Additionally, the assumption of zero transaction costs in the BSM model is often unrealistic. When transaction costs are present, the Delta hedging strategy becomes suboptimal. In such cases, alternative strategies are needed. The remainder of this work explores how Reinforcement Learning can provide an effective alternative to the classical Delta hedging approach.

3 Reinforcement Learning

To make this paper as self-contained as possible, we summarize key concepts of Reinforcement Learning (RL) that are essential for the implementation of our algorithm.

In RL problems, an agent interacts with an environment over time. The "environment" refers to the system outside the agent's direct control, and the agent's goal is to maximize cumulative rewards through a sequence of actions. Figure 1 illustrates the typical RL framework. At each time step t, the agent observes the environment's "state":

$$s_t \in \mathcal{S}, \tag{5}$$

where $\mathcal{S}$ denotes the "state space". Based on this observation, the agent selects an "action":

$$a_t \in \mathcal{A}(s_t), \tag{6}$$

where $\mathcal{A}(s_t)$ represents the set of feasible actions, which may depend on the current state s_t. After taking the action, at time $t+1$, the agent receives a "reward":

$$R_{t+1} = R(s_t, a_t, s_{t+1}), \tag{7}$$

which depends on the current state, the action taken, and the resulting next state. Additionally, the agent transitions to a new state s_{t+1}, completing the interaction cycle.

The sequence of states, actions, and rewards forms a stochastic process. The mathematical framework commonly used to model this process is the "Markov Decision Process (MDP)". In an MDP, the state and reward at time $t+1$ depend only on the current state and action, not on the full history:

$$\mathcal{L}\left(s_{t+1}, R_{t+1} | s_0, a_0, R_1, \ldots, s_t, a_t\right) = \mathcal{L}\left(s_{t+1}, R_{t+1} | s_t, a_t\right), \tag{8}$$

where $\mathcal{L}(X|Y)$ denotes the conditional probability law of X given Y. In most RL problems, the transition dynamics $\mathcal{L}$ are not explicitly known and must be estimated through interactions with the environment.

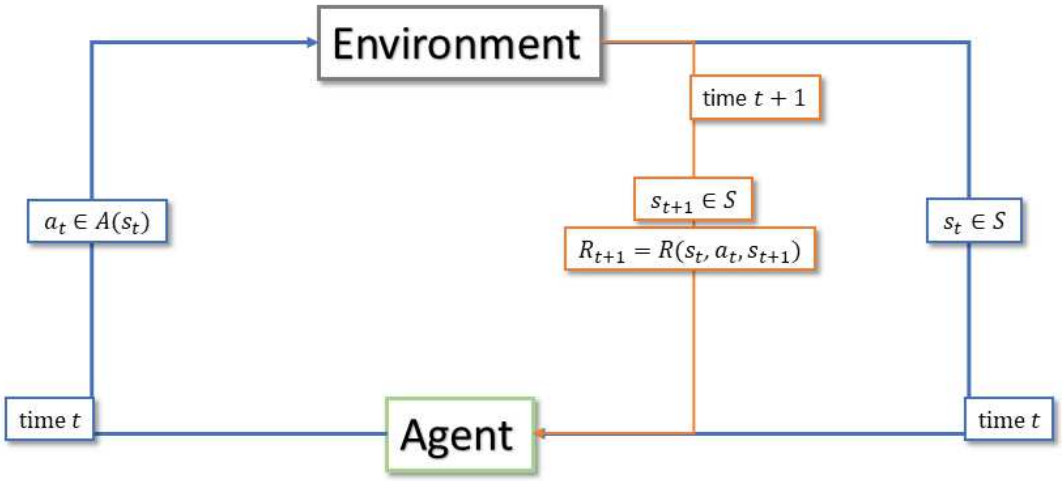


Figure 1: At time t, represented by the blue arrows, the agent receives the current state s_t from the environment (right arrow) and performs an action a_t, affecting the environment (left arrow). In the next period $t+1$, represented by the orange arrows, the environment returns a reward R_{t+1} and a new state s_{t+1}.

3.1 Policies and Value Functions

A "policy" defines the agent's behavior, specifying the action a to take for a given state s. A stationary Markovian policy depends only on the current state:

$$\begin{aligned} \pi : \mathcal{S} &\to \mathcal{A}(s), \\ s &\mapsto a = \pi(s). \end{aligned} \tag{9}$$

The combination of an initial state distribution, the MDP dynamics (8), and a policy π determines the evolution of the system as a "homogeneous Markov chain" (see [Csáji and Monostori, 20 for extensions to time-varying environments). To evaluate policies, we define the "cumulative discounted reward":

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{10}$$

where $\gamma \in (0, 1]$ is the discount factor, which weights immediate rewards more heavily than distant ones. The "state-value function" for a policy π is:

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t | s_t = s \right], \tag{11}$$

where $\mathbb{E}_\pi$ denotes the expectation assuming the agent follows policy π. Similarly, the "action-value function" is:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t | s_t = s, a_t = a \right], \tag{12}$$

which quantifies the expected cumulative reward starting from state s, taking action a, and subsequently following π.

An "optimal policy" π^* maximizes the expected cumulative reward for all states:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, v_\pi(s) \quad \forall s \in \mathcal{S}. \tag{13}$$

All optimal policies share the same "optimal value functions":

$$v_*(s) = \max_\pi v_\pi(s), \tag{14}$$

$$q_*(s,a) = \max_\pi q_\pi(s,a). \tag{15}$$

These satisfy the "Bellman optimality equations" (see [Sutton and Barto, 2018]):

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \mathbb{E}\left[R_{t+1} + \gamma v_*(s_{t+1}) | s_t = s, a_t = a\right], \tag{16}$$

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a_{t+1} \in \mathcal{A}(s_{t+1})} q_*(s_{t+1}, a_{t+1}) | s_t = s, a_t = a\right]. \tag{17}$$

3.2 Exploration, Exploitation, and Learning

In practice, RL algorithms alternate between "exploration" and "exploitation". During exploitation, the agent selects actions based on the current estimate of the optimal policy, while exploration involves trying alternative actions to improve the policy. A common approach is the ϵ-greedy policy:

$$\pi_{\epsilon\text{-greedy}}(s) = \begin{cases} \tilde{a}, & \text{if } u < \epsilon, \\ \operatorname{argmax}_a q(s,a), & \text{if } u \geq \epsilon, \end{cases} \tag{18}$$

where $\epsilon \in (0,1)$, $u \sim U(0,1)$, and $\tilde{a}$ is a random action. Over time, ϵ decreases to favor exploitation as the policy improves.

Under appropriate conditions, RL algorithms converge to the optimal value functions (15) through repeated training and interaction with the environment (see [Sutton and Barto, 2018]). This process is a hallmark of RL, distinguishing it from classical dynamic programming by its reliance on simulation rather than explicit knowledge of the MDP dynamics.

3.3 Model-Based vs. Model-Free RL

RL algorithms can be classified as "model-based" or "model-free". Model-based approaches rely on either known or estimated MDP dynamics (8), enabling planning and simulation. In contrast, model-free methods use only observed data, avoiding model risk but requiring sufficient data for training. Each approach has trade-offs in terms of efficiency, robustness, and applicability.

4 The algorithm

The hedging problem must be formulated in a manner suitable for applying Reinforcement Learning (RL). The objective is to optimize the mean-variance utility of the hedging target H_T defined in (21):

$$\mathbb{MV}[H] = \mathbb{E}[H] - \frac{\kappa}{2}\mathbb{V}[H], \tag{19}$$

where κ is a constant representing the trader's risk aversion. Define the change in the units of the asset S at time t_k as:

$$\Delta n_k := n_k - n_{k-1}.$$

The cost of updating the portfolio at time t_k is given by:

$$c_k := S_{t_k}\Delta n_k + c(\Delta n_k),$$

where the first term reflects the cost of trading in a perfectly liquid market, and the second term, $c(\Delta n_k)$, represents transaction costs, which are always positive and become zero only in a perfectly liquid market.

The hedging target H_T can then be expressed as:

$$H_T = -P_T - \sum_{k=0}^{N-1}\left(S_{t_k}\Delta n_k + c(\Delta n_k)\right) + n_{N-1}S_T + P_0. \tag{20}$$

Let us define the "profit and loss" (P&L) of the portfolio position at time t_k as:

$$\varrho_k := (-P_{t_k} + P_{t_{k-1}}) - n_{k-1}(S_{t_k} - S_{t_{k-1}}).$$

Substituting this into H_T, we obtain:

$$H_T = \sum_{k=1}^{N}\varrho_k - c(\Delta n_{k-1}), \tag{21}$$

where the hedging target is expressed as the sum of "local costs", defined as the differences between P&L and transaction costs.

Based on the reformulated hedging target, we propose the following "reward function":

$$R_t = (\varrho_t - c(\Delta n_{t-1})) - \frac{\kappa}{2}\left(\varrho_t - c(\Delta n_{t-1})\right)^2, \tag{22}$$

which encourages the agent to optimize the mean-variance utility of the local costs, and consequently, the overall hedging target. This is validated under the assumptions:

$$\mathbb{V}[H_T] = \sum_{k=1}^{N}\mathbb{V}[\varrho_k - c(\Delta n_{k-1})], \quad \mathbb{E}[\varrho_k] = 0,\ k = 1, \ldots, N.$$

To define the state space, we include all relevant and non-redundant information that the agent needs to make decisions at time t. The state variable is:

$$s_t := (S_t, \tau, n_{t-1}) \in \mathbb{R}^3,$$

where S_t is the current value of the underlying, $\tau = T - t$ is the time to maturity, and n_{t-1} is the current position in the underlying asset. The action variable is defined as:

$$a_t := \Delta n_t \in \mathbb{R},$$

where the action space is constrained by:

$$a_t \in (-n_{t-1}, L - n_{t-1}) = \mathcal{A}(s_t), \tag{23}$$

ensuring that the trader does not short-sells and does not exceed a lot size L. This state-dependent action space is also bounded, which facilitates optimization during the greedy policy computation.

We compute the trading costs as

$$c(a) = m \times tick \times (|a| + 0.01a^2), \tag{24}$$

where $tick > 0$ is the tick size, and $m > 0$ reflects market friction. The term $tick \times |a|$ reflects the cost of crossing a bid-offer spread, while the quadratic term models the market impact of the trade. This function is often used in the literature to quantify trading costs, but the flexibility of the algorithm allows to change it to reflect specific market conditions.

Although the reward function depends on the option price P_t, it is not explicitly included in the state space. Instead, P_t is simulated using the Black-Scholes-Merton (BSM) formula $P_t = p^{BSM}(S_t)$, ensuring compatibility with the RL framework without relying on a specific pricing model. Importantly, the intermediate option prices cancel out, as shown:

$$\sum_{t=1}^{T} \left(p^{BSM}(S_t) - p^{BSM}(S_{t-1})\right) = p^{BSM}(S_T) - p^{BSM}(S_0) = (S_T - K)^+ - P_0.$$

Our algorithm is based on the SARSA approach introduced in [Rummery and Niranjan, 1994] and refined in [Sutton and Barto, 2018]. SARSA estimates the optimal action-value function q_* through simulations of State-Action-Reward-State-Action (SARSA) sequences:

$$s_0^{(j)}, a_0^{(j)}, r_1^{(j)}, s_1^{(j)}, a_1^{(j)}, \ldots. \tag{25}$$

The value function is updated iteratively using the formula:

$$q_{k+1}(s_{t-1}, a_{t-1}) = q_k(s_{t-1}, a_{t-1}) + \alpha \left(R_t + \gamma q_k(s_t, a_t) - q_k(s_{t-1}, a_{t-1})\right), \tag{26}$$

where $\alpha \in [0, 1]$ is the learning rate. Setting $\alpha = 1$ simplifies this to:

$$q_{k+1}(s_{t-1}, a_{t-1}) = R_t + \gamma q_k(s_t, a_t). \tag{27}$$

The algorithm combines the SARSA method with supervised regression for value function approximation. Training proceeds over batches, starting with an initial value function $\hat{q}^{(0)}(s, a)$. Exploration is managed using the ϵ-greedy policy (18), where ϵ is gradually reduced during training.

To optimize the greedy policy we tested several global optimization methods, including basin-hopping [Wales and Doye, 1997], brute force, differential evolution [Storn and Price, 1997], dual annealing [Xiang et al., 1997], and we finally opted for SHGO [Endres et al., 2018] that, in our experience, ensured more stable and robust performances even in complex state-action spaces.

In the next section, we present numerical results demonstrating the algorithm effectiveness.

5 Applications

5.1 Case 1: Without Transaction Costs

In our first experiment, we compare the Reinforcement Learning (RL) policy to the standard delta hedging strategy under the assumption of no transaction costs.

5.1.1 Simulation Setup

We model the underlying dynamics using a geometric Brownian motion (2), with daily drift and volatility parameters set to $\mu = 0\%$ and $\sigma = 1\%$, respectively. The initial price is $S_0 = \$100$. The scenario involves a lot of $L = 100$ plain vanilla European call options on the underlying S, traded at-the-money with a strike price $K = \$100$. The options have a maturity of 10 days, during which the trader rebalances their position 5 times daily, resulting in $T = 50$ total rebalancing times. The tick size (minimum price change) is $tick = \$0.1$, and the risk-free rate is assumed to be $r = 0\%$. Transaction costs are set to zero by applying $m = 0$ in the general cost formula (24).

The RL agent is trained over $N_B = 10$ consecutive batches, each consisting of $J = 75,000$ episodes using the MDP (2). The epsilon-greedy policy begins with $\epsilon = 50\%$ and decreases at each batch iteration as $\epsilon \leftarrow \epsilon/3$ (see Table 1). During training, the SHGO global optimization algorithm [Endres et al., 2018] is used to compute the policy's maximization.

5.1.2 Training Results

Table 1 and Figure 2 illustrate the agent's training performance. As the batches progress, the rewards improve, reflecting the agent's learning and adaptation to the hedging problem.

Batch	ϵ	$\mathbb{E}_0\left[\sum_{t=1}^{T}\gamma^{t-1}R_t\right]$
1	50.000%	-848.449
2	16.667%	-252.860
3	5.556%	-143.128
4	1.852%	-108.470
5	0.617%	-96.350
6	0.206%	-94.690
7	0.069%	-92.977
8	0.022%	-93.011
9	0.007%	-92.451
10	0.002%	-92.717

Table 1: Reward evolution during training (no transaction costs).

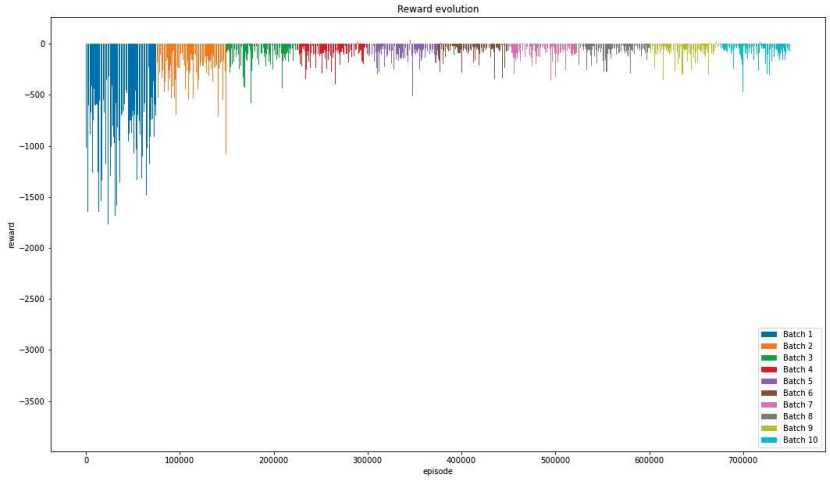


Figure 2: Reward evolution across batches (no transaction costs).

5.1.3 Out-of-Sample Testing

For out-of-sample validation, $J = 10,000$ MDP paths are simulated. Figure 3 compares the histograms of the hedging target H_T (20) obtained using the optimal delta strategy (4) and the RL-based strategy. Both methods achieve similar outcomes, with the delta strategy showing less variance due to its deterministic nature.

Figures 5 and 4 provide further insights:

- Figure 5a is composed of two plots. In the first (5a) we compare, for each simulation, the P&L of the RL hedging strategy (on the x- axis), to the P&L of the static strategy

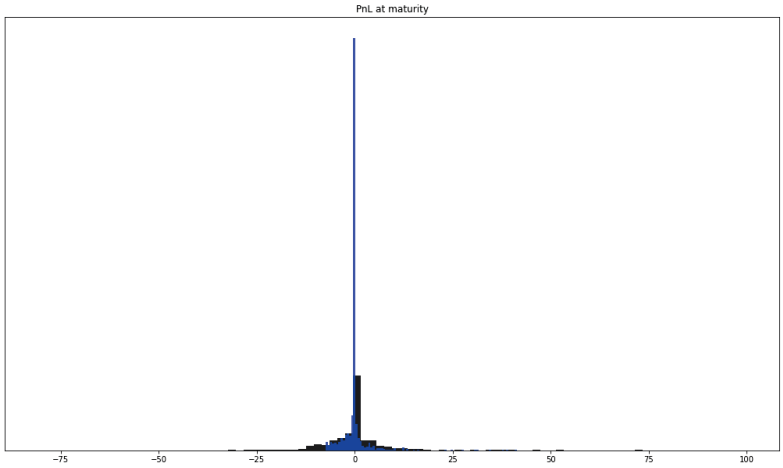


Figure 3: Histogram of hedging targets (no transaction costs) for the optimal delta strategy (blue) and RL agent (black).

in the option. A successful hedging strategy would return the same values for the two P&L's, therefore we can judge the effectiveness of the strategy by comparing it to the line $y = x$. The second (5b) shows the scatter plot of the trades performed by the RL agent versus those provided by the delta hedging (4). This plot shows a significant difference between the two policies, even if the final result is comparable.

- Figure 4 shows the evolution of one out of sample path of the hedging strategy performed by the RL agent. In blue, we can see the path of the agent's position in the option, whereas in orange we can see the path of the agent's position in the underlying. In green, we see the evolution of the hedged portfolio, which is approximately zero at all times. We see that the RL agent tries to keep the total value of the hedged portfolio (the green line) as close to zero as possible.

5.2 Case 2: With Transaction Costs

5.2.1 Simulation Setup

We introduce transaction costs by setting the multiplier $m = 1$ in the cost formula (24). The RL agent is trained in this new environment using the same setup as in the no-cost case.

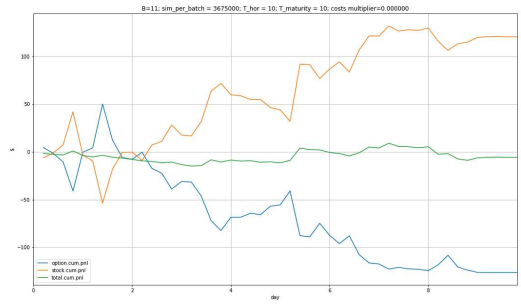
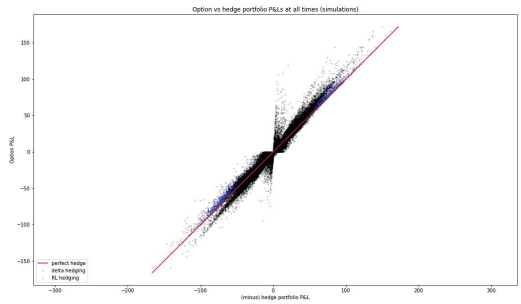
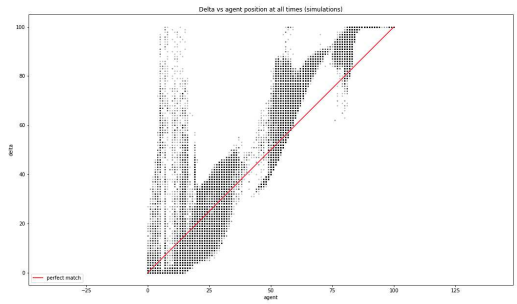


Figure 4: No-transaction costs case. The evolution of one path of the hedging strategy performed by the RL agent.



(a) RL vs benchmark hedge



(b) RL vs benchmark trades

Figure 5: No-transaction costs case. Figure (a) is the scatter plot of the (negative) P&L hedging obtained by the RL agent, versus the option P&L. Figure(b) is the scatter plot of the trades performed by the RL agent versus those of the standard delta hedging.

5.2.2 Training Results

Table 2 and Figure 6 show the evolution of rewards during training. The agent adapts to transaction costs, learning to optimize hedging performance while minimizing costs.

5.2.3 Out-of-Sample Testing

Figure 7 compares the histograms of hedging targets under transaction costs for both strategies. The RL agent shows a slightly larger variance, but effectively accounts for transaction costs, resulting in more conservative trading.

Figure 8 demonstrates the RL agent's ability to reduce trading costs compared to the delta hedging strategy, highlighting its cost-efficiency.

Figures 9 and 10 provide further details:

Batch	ϵ	$\mathbb{E}_0[\sum_{t=1}^{T} \gamma^{t-1} R_t]$
1	50.000%	-1220.311
2	16.667%	-358.981
3	5.556%	-187.995
4	1.852%	-133.114
5	0.617%	-116.638
6	0.206%	-111.007
7	0.069%	-109.778
8	0.022%	-102.280
9	0.007%	-102.054
10	0.002%	-102.651

Table 2: Reward evolution during the training phase across batches under transaction costs.

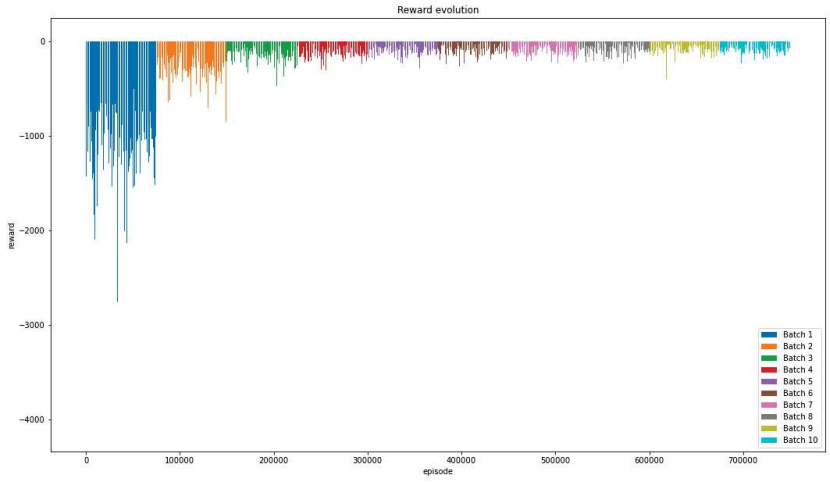


Figure 6: Reward evolution across batches (transaction costs).

- Figure 9a shows the scatter plot of the hedging portfolio P&L agent, versus the option P&L. As we have seen in Figure 7, the two outcomes are close to each other, and hence their scatter plot is distributed along the 45° line. Figure 9b shows the scatter plot of the trades performed by the RL agent versus those provided by the optimal Black-Merton-Scholes solution (4).

- Figure 10a shows the evolution of one out of sample path of the hedging strategy performed by the RL agent. In blue, we can see the path of the agent's position in the option, whereas in orange we can see the path of the agent's position in the underlying.

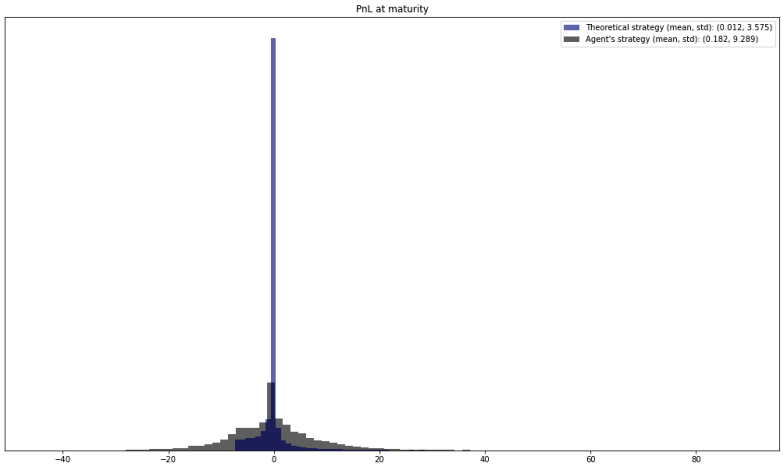


Figure 7: Histogram of hedging targets (transaction costs) for the optimal delta strategy (blue) and RL agent (black).

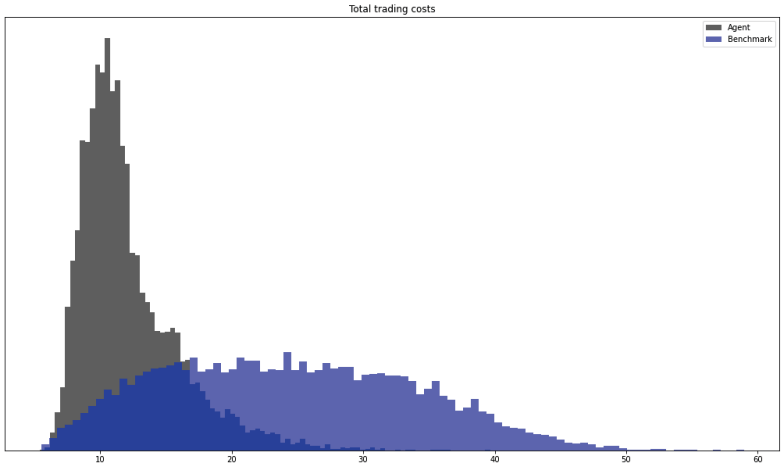
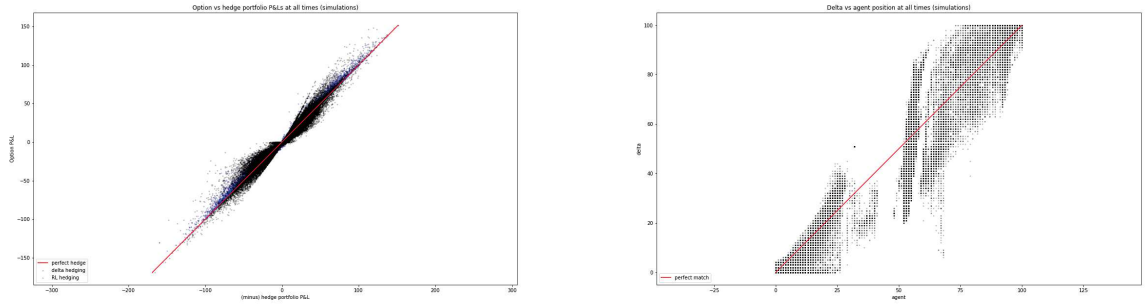


Figure 8: Total trading costs for the delta hedging strategy (blue) and RL agent (black).

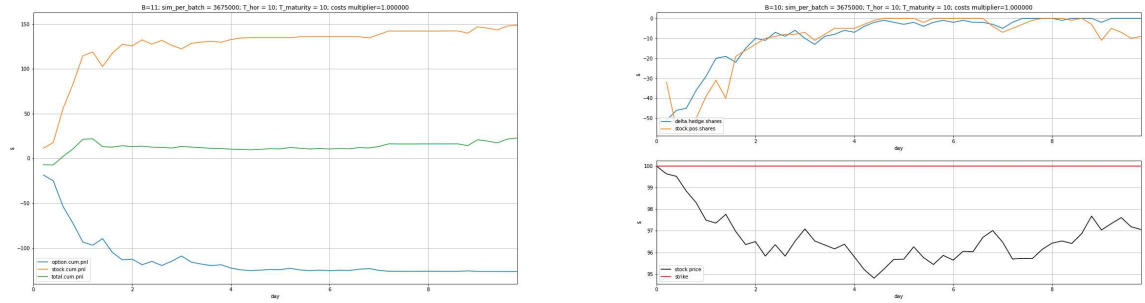
In green, we see the evolution of the hedged portfolio, which is approximately zero at all times. Figure 10b shows the evolution of the optimal and RL strategy in terms of shares of the underlying (top plot), whose value path is displayed in black (bottom plot).



(a) RL vs benchmark hedge

(b) RL vs benchmark trades

Figure 9: For the transaction costs case, in (a), the scatter plot of the (negative) hedge portfolio P&L obtained by the RL agent, versus the option P&L In (b), the scatter plot of the trades performed by the RL agent versus those provided by the optimal Black-Merton-Scholes solution (4).



(a) RL hedging strategy

(b) Hedge portfolio and underlying evolution

Figure 10: For the transaction costs case, in (a), the evolution of one out of sample path of the hedging strategy performed by the RL agent. In (b), the evolution of the optimal and RL strategy in terms of shares on the underlying (top plot), whose value path is displayed in black (bottom plot).

6 Conclusion

In this paper, we have proposed a Reinforcement Learning (RL) algorithm for hedging European call options in a market with discrete-time trading and transaction costs. Our approach combines the SARSA algorithm with the SHGO global optimization algorithm [Endres et al., 2018] to compute the policy maximization. The use of RL allows the algorithm to learn and adapt to complex market dynamics, and is applicable to more general situations than the traditional Delta Hedging strategies.

We began by validating the algorithm in an environment where the Black-Scholes model assumptions hold, allowing for a direct comparison between the RL-based strategy and the theoretical optimal Delta Hedging strategy. The results demonstrated that the RL agent successfully approximates the optimal hedging policy, achieving comparable performance in terms of minimizing the variance of the hedging error.

Next, we extended the evaluation to a market setting that incorporates transaction costs. The RL algorithm outperformed the standard Delta Hedging strategy, effectively balancing hedging accuracy with cost efficiency. By explicitly accounting for transaction costs in its reward function, the RL agent adapted its trading behavior to minimize unnecessary trades, reducing overall costs while maintaining robust hedging performance.

This work demonstrates that RL can provide an effective and adaptable solution for option hedging in modern financial markets. By addressing practical constraints such as transaction costs and discrete trading, the proposed algorithm offers a step forward in the application of machine learning techniques to quantitative finance.

As future research on this topic, we believe that it would be valuable to explore the performance of the RL algorithm in alternative market settings, such as those with stochastic volatility, jump diffusion dynamics, or illiquid trading conditions. The impact of incorporating more sophisticated cost models and risk metrics into the reward function should also be investigated. Finally, extending the approach to multi-asset portfolios and higher-dimensional problems would also represents a natural next step.

Acknowledgements

We acknowledge financial support by the project PRICE, financed by the Italian Ministry of University and Research under the program PRIN 2022, Prot. 2022C799SX. We thank Riccardo Cogo, Giovanni Nappi and Andrea Cosentini for discussion and valuable comments.

References

[Angelini and Herzel, 2015] Angelini, F. and Herzel, S. (2015). Evaluating discrete dynamic strategies in affine models. *Quantitative Finance*, 15(2):313–326.

[Angelini et al., 2009] Angelini, F., Herzel, S., et al. (2009). Measuring the error of dynamic hedging: a laplace transform approach. *Journal of Computational Finance*, 13(2):47.

[Black and Scholes, 1973] Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654.

[Buehler et al., 2019] Buehler, H., Gonon, L., Teichmann, J., and Wood, B. (2019). Deep hedging. *Quantitative Finance*, 19(8):1271–1291.

[Cao et al., 2019] Cao, J., Chen, J., Hull, J. C., and Poulos, Z. (2019). Deep hedging of derivatives using reinforcement learning. *Available at SSRN 3514586*.

[Csáji and Monostori, 2008] Csáji, B. C. and Monostori, L. (2008). Value function based reinforcement learning in changing markovian environments. *Journal of Machine Learning Research*, 9(8).

[Endres et al., 2018] Endres, S. C., Sandrock, C., and Focke, W. W. (2018). A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization*, 72(2):181–217.

[Giorgi et al., 2024] Giorgi, F., Herzel, S., and Pigato, P. (2024). A reinforcement learning algorithm for trading commodities. *Applied Stochastic Models in Business and Industry*, 40(2):373–388.

[Kolm and Ritter, 2019] Kolm, P. N. and Ritter, G. (2019). Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, 1(1):159–171.

[Merton, 1973] Merton, R. C. (1973). Theory of rational option pricing. *The Bell Journal of economics and management science*, pages 141–183.

[Rummery and Niranjan, 1994] Rummery, G. A. and Niranjan, M. (1994). *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK.

[Storn and Price, 1997] Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

[Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press.

[Vittori et al., 2020] Vittori, E., Trapletti, M., and Restelli, M. (2020). Option hedging with risk averse reinforcement learning. *arXiv preprint arXiv:2010.12245*.

[Wales and Doye, 1997] Wales, D. J. and Doye, J. P. (1997). Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116.

[Xiang et al., 1997] Xiang, Y., Sun, D., Fan, W., and Gong, X. (1997). Generalized simulated annealing algorithm and its application to the thomson model. *Physics Letters A*, 233(3):216–220.

RECENT PUBLICATIONS BY *CEIS Tor Vergata*

The Long-Run Effects of R&D Subsidies on High-Tech Start-Ups: Insights From Italy
Christoph Koenig, Letizia Borgomeo and Martina Miotto
CEIS *Research Paper*, 585 October 2024

With a Little Help From the Crowd: Estimating Election Fraud with Forensic Methods
Christoph Koenig
CEIS *Research Paper*, 584 October 2024

Biases and Nudges in the Circular Economy: A Review
Luca Congiu, Enrico Botta and Mariangela Zoli
CEIS *Research Paper*, 583 October 2024

Energy Shocks, Pandemics and the Macroeconomy
Luisa Corrado, Stefano Grassi, Aldo Paolillo and Francesco Ravazzolo
CEIS *Research Paper*, 582 August 2024

The Multivariate Fractional Ornstein-Uhlenbeck Process
Ranieri Dugo, Giacomo Giorgio and Paolo Pigato
CEIS *Research Paper*, 581 August 2024

The Macro Neutrality of Exchange-Rate Regimes in the presence of Exporter-Importer Firms
Cosimo Petracchi
CEIS *Research Paper*, 580 July 2024

Monetary Regimes and Real Exchange Rates: Long-Run Evidence at the Product Level
Jason Kim, Marco Mello and Cosimo Petracchi
CEIS *Research Paper*, 579 June 2024

On the Output Effect of Fiscal Consolidation Plans: A Causal Analysis
Lorenzo Carbonari, Alessio Farcomeni, Filippo Maurici and Giovanni Trovato
CEIS *Research Paper*, 578 May 2024

Ordered Correlation Forest
Riccardo Di Francesco
CEIS *Research Paper*, 577 May 2024

Ups and (Draw)Downs
Tommaso Proietti
CEIS *Research Paper*, 576 May 2024

DISTRIBUTION

Our publications are available online at www.ceistorvergata.it

DISCLAIMER

The opinions expressed in these publications are the authors' alone and therefore do not necessarily reflect the opinions of the supporters, staff, or boards of CEIS Tor Vergata.

COPYRIGHT

MEDIA INQUIRIES AND INFORMATION

For media inquiries, please contact Barbara Piazzi at +39 06 72595652/01 or by e-mail at piazzi@ceis.uniroma2.it. Our web site, www.ceistorvergata.it, contains more information about Center's events, publications, and staff.

DEVELOPMENT AND SUPPORT

For information about contributing to CEIS Tor Vergata, please contact at +39 06 72595601 or by e-mail at segr.ceis@economia.uniroma2.it