A Comparative Study of Constraint-Driven Reinforcement Learning Towards Managing Market Making Risk

Zimmer, R. and Costa, O.

February 1, 2025

Abstract

Reinforcement Learning (RL) has emerged as a promising framework for developing adaptive and data-driven strategies, enabling market makers to optimize decision-making policies based on interactions with the limit order book environment. This paper explores the integration of risk management constraints into reinforcement learning-based market making to address challenges such as inventory risk, adverse market movements, and overnight exposure. We explore two types of restrictions: hard limits, which enforce strict conditions like zero inventory at market close, and incentive-based restrictions, which penalize risky behaviors via penalties in the reward structure. These mechanisms aim to enhance stability and incorporate domain-specific knowledge into the RL framework.

Our contributions include a practical implementation of a restricted RL agent using the PPO algorithm, alongside a comparative evaluation of restriction types against an unrestricted benchmark under a simplified market model. Experimental results, conducted under synthetic-realistic market conditions, demonstrate the effectiveness of incorporating restrictions in balancing profitability and risk management, as evidenced by an analysis of the financial return analysis and risk metrics. These findings allow for a path towards more robust and replicable approaches to dynamic risk management in market making.

1 Introduction

Market making in financial markets involves continuously quoting buy and sell prices, and Dealing with supply and demand imbalances, as well as the risks of holding inventory and speculation is an intrinsic aspect of market making and a focus of recent research [4, 7]. Market makers are essential, as they help narrowing bid-ask spreads, reduce price volatility, and maintain market stability, particularly in times of uncertainty by trading and providing liquidity in the market [9, 15]. Correctly reducing inventory risk benefits not only individual market makers but all market participants as a whole. With the increase of computational power and data-driven automated systems, placing optimal bid and ask prices as a market maker is becoming an almost completely automatized task, but such a transition comes with additional caveats, such as slippage, overnight inventory and adverse market movements risks on agent inventory [4, 2].

Recent research focusing on Reinforcement Learning (RL) shows promising results for optimizing placed bid-ask prices by market making strategies. The RL framework defines market makers as agents that learn by interacting with an environment - which in our case is the limit order book - and optimizing their decision-making policies through trial and error and the respective outcome (also called reward [1]) of the environment. The primary objective of RL-based market making is to find the so called optimal policy that maximizes cumulative rewards, which translates to choosing prices that provide positive spread differences while minimizing risk factors such as agent inventory or price volatility. The RL approach is based on the Bellman equation for state values or state-action pair values, which recursively define the value of a policy by considering the expected return of discounted future rewards. A common implementation of the RL approach is to continuously update the agent's policy based on the sampled environment information and rewards, allowing the agents to dynamically change their price choosing strategies and adapt to changing market conditions in real-time [21].

Unrestricted RL-based market making, however, can lead to potentially risky behaviors. In financial markets, and in long high-frequency intervals of trading, inventory risk and market impact are not to be

[1] Not to be confused with financial returns or rewards.

ignored, as agents accumulating large inventory positions become exposed to significant price fluctuations and adverse market direction changes. This makes holding inventory result in an unwanted market exposition to the market making agent, becoming thus subject to undesired risks. Recent literature discusses the means of implementing restrictions in the RL framework to ensure stability and applicability towards real-world implementations.

In the context of this paper we will discuss restrictions in two possible forms: as hard equality or interval limits, or as incentives integrated into the agent's reward structure. Hard limits enforce strict numerical conditions, such as having zero inventory at market close, that must be met. On the other hand, incentive-based restrictions simply increase the loss on rewards the closer the agent is to the restriction, such as a negative reward the larger the inventory on market close, thus making the agent adverse to so-called overnight risk, which will be discussed in further sections. Such restrictions are aimed to reduce target volatility and increase the possibilities of introducing domain knowledge and known stylized facts of the market into the learning process of the agent [13, 19].

Our main contributions aims to be a pragmatic implementation of restrictions for market making within an RL framework, contributing with a replicable way of minimizing undesired risk taken by the agent while maintaining the computational feasibility of the chosen architecture. Finally, the impact on the financial returns of the RL agents averse to overnight risk will be benchmarked against a closed-expression optimal solution under a simplified market model, aiming to validate the usage of hand-crafted restrictions in real-world implementations.

2 Bibliography Review

A bibliographical research was performed to determine the state of the art in the field of the project, both in terms of the models and algorithms used as well as the chosen state and action spaces which perform best with regard to simulating realistic, real-world market dynamics. The search was made using the Web of Science repository, covering the last 5 years, with the following search key:

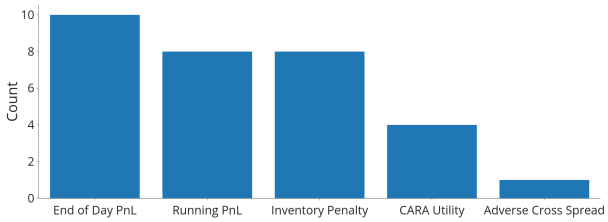
```
(
    "reinforcement learning" OR "dynamic programming" OR
    "optimal control" OR "control theory" OR "machine learning"
)
AND
(
    "market making" OR "market maker"
)
```

Initially, 64 references were selected and deemed relevant to the project, with 23 of them being selected for further analysis and effectively used in our analysis, and 4 additional references being added to the list after the initial selection. The references were selected based on their relevance to the project and tagged according to the following groups (for each non-binary category, combinations of tags were allowed):

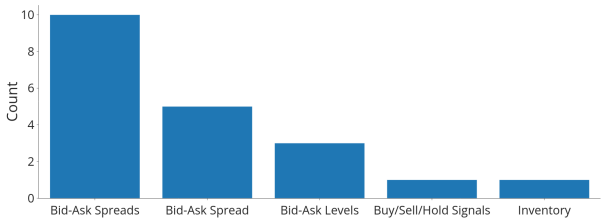
- Type of data (simulated, real-time connections, and others);
- Chosen state space (vectors of bid-ask spreads, order imbalance, N-depth order books, and others);
- Chosen action space (limit orders, market orders, and others);
- Chosen Reward space (spread, volume, profit, and others);
- Algorithms used (Q-Learning, Deep Q-Learning, Actor-Critic, or others);
- Whether a multi-agent approach was used (dueling or market agents);
- If a model-free environment was used (model-based or model-free);
- Metrics used for comparison (Sharpe Ratio, PnL, and others).
- Finally, results were tagged by their benchmarks, that is, the strategies used for comparison.

2.1 State, Action, and Reward Spaces

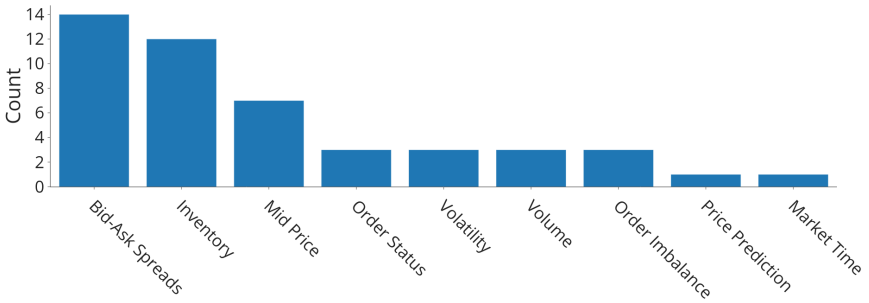
State-of-the-art references defined state spaces primarily based on market-level observations, specifically top-of-book quotes and n-depth book levels which align well with the real-time data available to market-making agents [12, 3]. Additionally, agent inventory was also a common feature, reflecting the importance of managing risk and liquidity in market-making strategies [16, 6]. Action spaces included mostly only one pair of quotes, with some references allowing the choice of multiple bid-ask levels. Some references also used book levels as discrete action space variables.



(a) Tagged reward space function variables



(b) Tagged action space variables

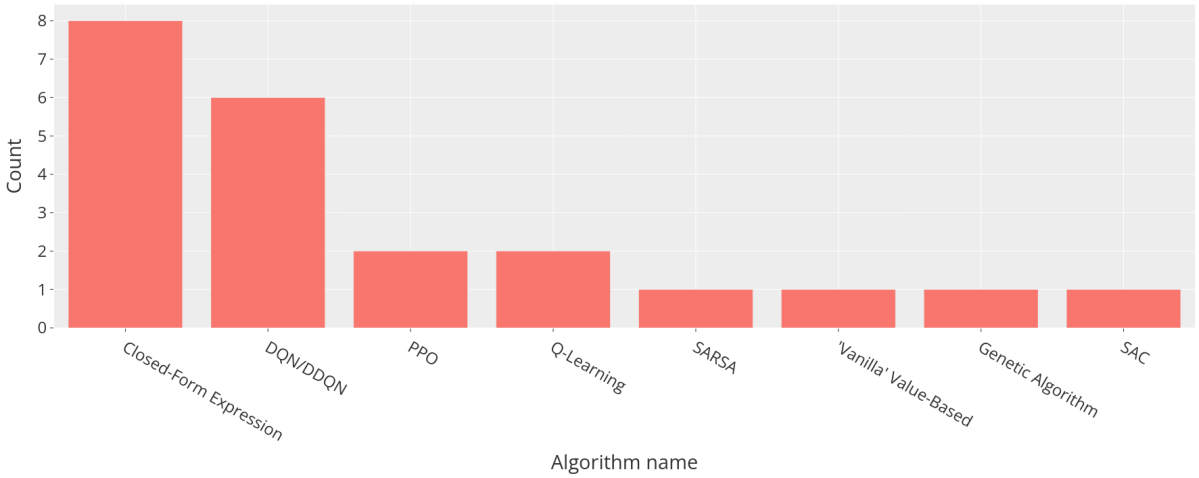


(c) Taged state space variables

Finally, almost all reward structures used some form of PnL and end-of-day or running liquidation and inventory penalties, mostly due to the need to balance profitability and inventory risk [20, 7]. However, the inclusion of additional explicit constraints or reward penalties, such as inventory and overnight risk penalties was rarely discussed or mentioned, even though being essential to ensure practical applicability and risk management in live trading scenarios [14, 18]. The tagged action space variables include a separate "Inventory" variable that refeers to portfolio strategies that output a target maximum inventory. For implementations that use specific quantities per order, no additional tags were used, as only one reference did not add quantities as part of the action space and instead used fixed quantities.

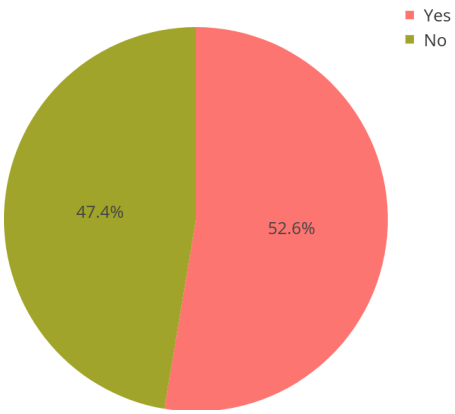
2.2 Algorithms and Performance Metrics

Recent trends in reinforcement learning research emphasize mostly model-free approaches, particularly Deep Q-Learning and Actor-Critic methods, which leverage neural networks to approximate value functions or policies to deal with the high-dimensional nature of limit order books [16, 6]. The literature showed that model-free approaches had strong adaptability capabilities towards changing market conditions while still maintaining acceptable training times, with PPO and DQN (and some DDQN) being the best performers in terms of convergence and stability [20, 8].

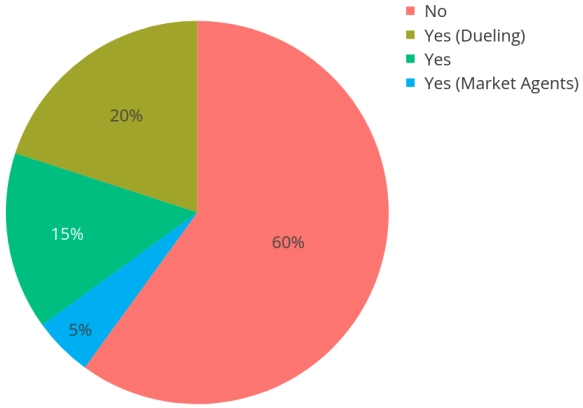


(a) Algorithms used as main research focus

While market makers that maintain overnight positions obtain a worse long-term sharpe ratio compared to strategies that at end-of-day have null inventories, the use of overnight position penalties was only mentioned in 3 of the references [14, 18, 20].



(a) Principal algorithm is model free



(b) Multiple agents used in training

Overall, the bibliographical review underscores the growing relevance and preference for reinforcement learning algorithms in the market-making domain, particularly when combined with realistic state, action, and reward spaces. The insights gained from the state and action spaces, as well as the reward function guided our choice for the space variables used in this paper, as well as the design for our chosen neural network architecture, while still showing a gap in the literature regarding usage of restrictions in the RL framework, both as hard limits and incentives.

3 Methodology

3.1 Problem Definition

The market-making problem addressed in this work involves designing an optimal trading policy for an agent using reinforcement learning (RL). The agent aims to maximize profit while managing risks, particularly inventory risk under the restriction of having zero inventory at market close, and interacts with the environment by quoting bid and ask prices and adjusting offered quantities. The environment dynamics

are modeled by an underlying limit order book (LOB) and its behaviour will be described in the following sections. As discussed previously, the main challenge for choosing an adequate agent and its policy lies in balancing profitability with risk management, especially regarding inventory at the close of the market, where overnight positions can expose the agent to significant risks.

3.2 A Formal Description of the Chosen RL Environment

In modeling the RL environment, we initially utilize a continuous-time, continuous-state Markov Chain framework, and later transition to a discrete representation to address computational space constraints. The specific case in which a Markov Chain also has an associated reward distribution R for each state transition is called a Markov Reward Process and given that the MM problem also has a decision process that affects the transition probabilities it is therefore called a Markov Decision Process (MDP) in control literature. A Markov Decision Process can generically be defined as a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, R)$, where:

- $\mathcal{S}$ is a set of states called the state space.
- $\mathcal{A}$ is a set of actions called the action space.
- $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function for the MDP.
- $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S}' \to \mathbb{R}$ is the reward function associated with each state transition.

Then, the agent interacts with the environment by choosing actions from the action space $\mathcal{A}$ in response to observed states $s \in \mathcal{S}$ according to a policy $\pi(s, a)$. The agent's goal is to maximize cumulative rewards over time, which are obtained from the reward function R. An adjusted version of the Bellman equation is used to derive the optimal policy for the agent, and the episode's return is used to evaluate the agent's performance.

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[G_t | s_t, \pi\right]$$

The return G_t is defined as the sum of rewards obtained from time t to the end of the episode (usually, as well as in our case, discounted by a factor γ to favor immediate rewards):

$$G_t = \int_t^T \gamma^{k-t} R_k dk$$

where R_t is the observed reward at time t and T is the time of the episode's end.

3.2.1 Chosen State Space

We choose a state space that tries to best incorporate the historical events of the limit order book into a single observable state using commonly used indicators and LOB levels, as well as intrinsic features to the agent deemed relevant and chosen through our initial bibliography research, as well as [7]. Given our performed bibliographical research, we chose the agent's current inventory for the intrinsic feature and a set of indicators for the extrinsic features: the Relative Strength Index (RSI); order imbalance (O); and micro price (MP). Additionally, for a fixed number D of LOB price levels the pair (δ^d, Q^d), where δ^d is the half-spread distance for the level $d \leq D$, and Q^d the amount of orders posted at that level is added to the state as a set of tuples, for both the ask and bid sides of the book. The state space can therefore be represented by the following expression:

$$s_t \in \mathcal{S} = \left\{ \mathrm{RSI}_t, \mathrm{OI}_t, MP_t, \mathrm{Inventory}_t, \left(\delta_t^{d,ask}, Q_t^{d,ask}\right)_{d=1}^{D}, \left(\delta_t^{d,bid}, Q_t^{d,bid}\right)_{d=1}^{D} \right\}$$

where $0 < t < T$.

The indicators for our chosen market simulation framework are defined individually by values directly obtained from the observed LOB, and serve as market state summaries for the agent to use:

- **Order Imbalance (OI):** Order imbalance measures the relative difference between buy and sell orders at a given time. It is defined as:

$$\mathrm{OI}_t = \frac{Q_t^{\mathrm{bid}} - Q_t^{\mathrm{ask}}}{Q_t^{\mathrm{bid}} + Q_t^{\mathrm{ask}}},$$

where Q_t^{bid} and Q_t^{ask} represent the total bid and ask quantities at time t, respectively. $\mathrm{OI}_t \in [-1, 1]$, with $\mathrm{OI}_t = 1$ indicating complete dominance of bid orders, and $\mathrm{OI}_t = -1$ indicating ask order dominance.

- **Relative Strength Index (RSI):** The RSI is a momentum indicator that compares the magnitude of recent gains to recent losses to evaluate overbought or oversold conditions. It is given by:

$$\mathrm{RSI}_t = 100 - \frac{100}{1 + \frac{\text{Average Gain}}{\text{Average Loss}}},$$

where the *Average Gain* and *Average Loss* are computed over a rolling window (in our case, fixed 5 minute observation intervals). Gains are the price increases during that window, while losses are the price decreases.

- **Micro Price ($P_{\mathbf{micro}}$):** The micro price is a weighted average of the best bid and ask prices, weighted by their respective quantities:

$$P_{\mathrm{micro},t} = \frac{P_t^{\mathrm{ask}} Q_t^{\mathrm{bid}} + P_t^{\mathrm{bid}} Q_t^{\mathrm{ask}}}{Q_t^{\mathrm{bid}} + Q_t^{\mathrm{ask}}},$$

where P_t^{ask} and P_t^{bid} represent the best ask and bid prices at time t.

3.2.2 Chosen Action Space

The control, or agent, interacts with the environment choosing actions from the set of possible actions, such that $a \in \mathcal{A}$ in response to observed states $s \in \mathcal{S}$ according to a policy $\pi(s, a)$ which we will define shortly, and the end goal is to maximize cumulative rewards over time. The agent's chosen action impacts the evolution of the system's dynamics by inserting orders into the LOB that might move the observed midprice, to introduce features of market impact into our model.

The action space $\mathcal{A}$ includes the decisions made by the agent at time t, specifically the desired bid and ask spreads pair $\delta^{\mathrm{ask}}, \delta^{\mathrm{bid}}$ and the corresponding posted order quantities $Q^{\mathrm{ask}}, Q^{\mathrm{bid}}$:

$$\mathcal{A} = \left\{(\delta^{\mathrm{ask}}, \delta^{\mathrm{bid}}, Q^{\mathrm{ask}}, Q^{\mathrm{bid}}), \forall \delta \in \mathbb{R}^+, \forall Q \in \mathbb{Z}\right\}$$

3.2.3 Episodic Reward Function and Returns

The episode reward function $R_t \in \mathbb{R}$ reflects the agent's profit and inventory risk obtained during a specific time in the episode. It depends on the spread and executed quantities, as well as the inventory cost and was choosen according to commonly used reward structures taken from the literature review.

The overall objective is to maximize cumulative utility while minimizing risk associated with inventory positions, and later insert restrictions so the risk for inventory is either limited at zero at market close, or incurring in larger penalties on the received rewards. For our model the utility chosen is based on a running Profit and Loss (PnL) score while still managing inventory risk. The choosen reward function is based on a risk-aversion enforced utility function, specifically the *constant absolute risk aversion (CARA)* [1, 17] and depends on the realized realized spread δ and the realized quantity q (not to confuse with the agent's posted order quantity Q).

The running PnL at time t is computed as follows, and a penalty for holding large inventory positions is discounted from the *PnL* score, as follows:

$$\begin{aligned}
\text{Running PnL}_t &= \delta_t^{\mathrm{ask}} q_t^{\mathrm{ask}} - \delta_t^{\mathrm{bid}} q_t^{\mathrm{bid}} + \mathrm{I}_t \cdot \Delta M_t, \\
\text{Penalty}_t &= \eta \left(\text{Inventory}_t \cdot \Delta M_t\right)^+, \\
\mathrm{PnL}_t &\coloneqq \text{Running PnL}_t - \text{Penalty}_t
\end{aligned}$$

where η is the penalty factor applied to positive inventory changes.

Finally, the reward function is defined as the negative of the exponential of the running PnL, which is a common choice for risk-averse utility functions: The constant absolute risk aversion (CARA) utility function is defined as follows, where γ is the risk aversion parameter:

$$R_t = U(\mathrm{PnL}_t) = -e^{-\gamma \cdot \mathrm{PnL}_t},$$

3.3 State Transition Distribution

The previously mentioned transition probability density P is given by a Stochastic Differential Equation expressed by the Kolmogorov forward equation for Markov Decission Processes:

$$\frac{\partial P(s', t|s, a)}{\partial t} = \int_{\mathcal{S}} \mathcal{L}(x|s, a, t) P(s'|x, a, t) dx \tag{1}$$

for all $s, s' \in \mathcal{S}$ and all times t before market close T, that is, $t \leq T$, where a is choosen by our control agent according to a policy $\pi(s)$. $\mathcal{L}$ is the generator operator and governs the dynamics of the state transitions given the current time.

In continuous-time and state MDPs, the state dynamics is reflected by $\mathcal{L}$ and modern approaches to optimal control solve analytically by obtaining a closed-form expression for the model's evolution equations, as in **(author?)** [2, 11]. or numerically by approximating its transition probabilities, as in **(author?)** [10, 18, 5]. Closed-form expression for $\mathcal{L}$ are obtainable for simple models, that usually require that market order impact be not disconsidered, which is not the case for our proposed model, and solving for the generator operator is therefore outside the scope of this paper. A numerical approach will be used furthermore when we define a neural network approximator for the actor's policy using the Proximal Policy Optimization (PPO) algorithm for model weight optimization in Section 4. as well as both restriction formulations.

3.4 Market Model Description and Environment Dynamics

For our model of the limit order book the timing of events follows a *Hawkes process* to represent a continuous-time MDP that captures the observed stylized fact of clustered order arrival times.

The Hawkes process is a *self-exciting process*, where the intensity $\lambda(t)$ depends on past events. Formally, the intensity $\lambda(t)$ evolves according to the following equation:

$$\lambda(t) = \mu + \sum_{t_i < t} \phi(t - t_i),$$

where $\mu > 0$ is the baseline intensity, and $\phi(t - t_i)$ is the *kernel function* that governs the decay of influence from past events t_i. A common choice for ϕ is an exponential decay:

$$\phi(t - t_i) = \alpha e^{-\beta(t - t_i)},$$

where α controls the magnitude of the self-excitation and β controls the rate of decay.

The bid and ask prices for each new order are modeled by two separate *Ornstein-Uhlenbeck (OU) processes* to capture the mean-reversion behavior of spreads over the midprice:

$$ds_t = \theta(\mu - s_t)dt + \sigma dW_t,$$

where s_t is the market spread at time t, θ is the rate of mean reversion, μ is the long-term spread mean and σ its volatility. The Wiener process W_t is used to represent random market fluctuations. Both μ and σ will be estimated using historical datasets of market LOBs in Section 5 ¡experimentos¿ to replicate real observed market conditions.

The bid and ask spreads δ_t^{bid} and δ_t^{ask} for orders conditioned on their arrival follow normal distributions:

$$\delta_{t+1}^{\text{ask}} \sim \mathcal{N}(\mu + M_t + s_t, \sigma^2), \quad a_{t+1}^{\text{bid}} \sim \mathcal{N}(\mu + M_t - s_t, \sigma^2),$$

Whenever a new limit order that narrows the bid-ask spread or a market order arrive the midprice is updated to reflect the top-of-book orders. The midprice M_{t+1} is therefore obtained by averaging the current top-of-book bid and ask prices:

$$M_{t+1} \coloneqq \frac{2M_t + \delta_t^{ask} - \delta_t^{bid}}{2} \sim \mathcal{N}\left(\mu + M_t, \frac{\sigma^2}{2}\right)$$

and at $t = 0$, the midprice is defined according to some starting point of the simulation, usually set to observed historical prices at market open.

We can obtain the distribution for the midprice by analyzing the dynamics of the top of book bid and ask prices, resulting in the following *Brownian Motion* process:

$$dM_t = \mu dt + \frac{\sigma^2}{2} dW_t,$$

where W_t is a Wiener process, and therefore means our choosen midprice process reflects a stylized fact of LOBs commonly observed in markets ¡inserir referencia book HFT¿, that is, having normally distributed returns.

Finally, the order quantities q_t^{ask} and q_t^{bid} are modeled as Poisson random variables:

$$q_t^{\text{ask}}, q_t^{\text{bid}} \sim \text{Poisson}(\lambda_q),$$

where λ_q is the average order size.

3.5 Decision Process and Steps to Maximize the Agent's Objective

In reinforcement learning, the Bellman equation is a fundamental recursive relationship that expresses the value of a state in terms of the immediate reward and the expected value of subsequent states. For a given policy π, the Bellman equation for the value function $V(s)$ with respect to the chosen reward function is:

$$V(s) = \mathbb{E}_\pi \left[R(s', s, a) + \gamma V(s') \right],$$

where $V(s)$ is the value function, representing the expected return (cumulative discounted rewards) starting from state s, going to state s' by taking action a and continuing the episode, $R(s', s, a)$ is the reward obtained from taking action a in state s and ending in state s', and γ is the discount factor, which weighs the value favorably towards immediate in comparison to future rewards.

The Bellman equation underpins the process of optimal policy derivation, where the goal is to find the policy π^*, that is, the control for our action space $\mathcal{A}$ that maximizes the expected return, $V^*(s)$. The optimal value function satisfies the equation:

$$V^*(s) = \max_\pi \mathbb{E}_\pi \left[R(s, a) + \gamma V^*(s') \right].$$

Thus, the RL approach is a way of obtaining the optimal control π^* given an agent's **objective** that maximizes the expected return for all states. For our approach, a numerical approximation of the policy is taken due to computational space limitations, the use of neural networks for the underlying approximator is a viable approach and the current State-of-the-art, as proposed by ¡ref¿. The design of a **loss function**, a neural network architecture and choice of optimizer will be discussed in the following section shortly after a small overview of the considered algorithms.

3.5.1 Deep Reinforcement Learning and Chosen Loss Function

In the context of deep reinforcement learning the loss function is a derivation from the usual return function $G(\tau)$. In the context of **Proximal Policy Optimization (PPO)** and Actor Critic algorithms overall, this loss function can be interpreted as the negative of the expected return, which we aim to minimize during training. The model is trained to improve its policy such that the chosen actions maximize the cumulative rewards over time, directly aligning with the Bellman equation's goal of maximizing the value function.

For **PPO**, the loss function is:

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

where $r_t(\theta)$ is the probability ratio between the new and old policies, $\hat{A}_t$ is the advantage function, and the clipped term ensures that updates to the policy do not deviate excessively from the previous policy, promoting stability during training. The critic loss is defined as:

$$L_{\text{critic}} = \mathbb{E}_t \left[\left(R_t - V(s_t) \right)^2 \right],$$

which minimizes the error between the predicted value $V(s_t)$ and the actual return R_t. Both losses are minimized simultaneously during training to improve the policy and the value function, through multiple sequential backward passes over batches of gathered episodes. Each pass updates the model's weights by applying the usual backward propagation pass for neural networks to minimize the loss function according to the Adam optimizer.

3.5.2 Benchmark Closed-Form Expression for Simplified Model

To guide the agent's learning, we incorporate a closed-form expression for the best bid-ask pair prices derived from a simplified version of the environment. This analytical solution offers an initial policy for the agent to start from, specifically by assuming normally distributed non-mean reverting spreads and constant order size and exponentially distributed order flow dynamics. This policy serves both as a starting for training and a benchmark for evaluating the agent's learned performance. As it depends on a simpler model for the market to be implemented, our expectations are for it to beat the deep RL models for the simpler market model, and by definition be unusable for the more complex one.

¡citar artigo que usa modelo simplificado de mercado vs complexo sem versao analitica¿

4 Implementation and Model Description

4.1 Model Architecture

4.2 Main Training Loop

Initialize the environment, Actor-Critic model, and optimizer
for each episode in range *num_episodes* **do**
  Collect trajectories:
    Initialize the state: *state*
    Initialize an empty trajectory buffer
  for each timestep in the episode **do**
      Select action a using the Actor network
      Execute action a and observe reward r and next state $state'$
      Store transition $(state, a, r, state')$ in the trajectory buffer
      Set $state = state'$
    if episode ends or maximum timestep reached **then**
        Break out of timestep loop
    end if
  end for
  Compute GAE and Returns:
    Compute advantages and returns using GAE
  Update model:
    Update the Actor and Critic networks using the collected trajectories
    Compute episode reward and store it in reward history
end for

4.3 Generalized Advantage Estimation and Weight Update Mechanism

Input: rewards r_t, values V_t, done flags d_t
Initialize advantages as an empty list
advantage $\leftarrow 0$
value $\leftarrow$ append terminal value 0 to V_t (last value)
for each timestep t in reversed range of trajectory length **do**
  Compute TD-error:
    $\delta_t = r_t + \gamma \cdot V_{t+1} \cdot (1 - d_t) - V_t$
  Compute advantage:
    advantage $= \delta_t + \gamma \cdot \lambda \cdot (1 - d_t) \cdot advantage$
    Insert advantage at the beginning of advantages list
end for
Return: advantages, returns $R_t = V_t +$ advantages
Input: trajectories, number of epochs *epochs*, batch size *batch_size*
for each epoch in range *epochs* **do**
  Shuffle and create data batches from trajectories
  for each batch of states, actions, old log probs, advantages, returns in data loader **do**
    Compute new log probs:
      Use the policy network to compute action probabilities and log probs for the batch
    Calculate the ratios:
      ratios = exp(log_probs − old_log_probs)
    Compute surrogate loss:
      surr1 = ratios · advantages
      surr2 = clip$(ratios, 1 - \epsilon, 1 + \epsilon)$· advantages
      policy loss = $-\min(surr1, surr2)$.mean()
    Compute value loss:
      value_loss = MSELoss(state_values, returns)
    Compute entropy:
      entropy = entropy of the action distribution
    Compute total loss:
      $loss = policyloss + valueloss -$ entropy_coef $\cdot$ *entropy*
    Backpropagation:
      Zero gradients, perform backward pass, and update network parameters
  end for
end for

5 Realized Experiments and Result Discussion

5.1 Experiment Setup

The experiments were conducted using the OpenAI Gym environment for the limit order book, which was implemented in Python. The environment was set up with a limit order book with dynamics according to the equations described in Section 3.4.

Both restricted agents were trained using the Proximal Policy Optimization (PPO) algorithm, which is a model-free, on-policy algorithm that optimizes the policy directly. A total of 10,000 episodes were simulated, with each episode consisting of a total time of at least 390 minutes (or 6.5 hours), with about 1 time step per minute. Per gathered trajectory, 500 epochs were used for updating both the policy and value networks. We used the Adam optimizer with a learning rate of 10^{-4} and a discount factor of 0.99. For comparison metrics, we used the Sharpe ratio, the average daily return, and the average daily volatility.

5.2 Experiment Results

The agent's policy was trained using the PPO algorithm with no restrictions on the agent's actions.

References

[1] Kenneth J. Arrow. *Aspects of the Theory of Risk-Bearing*. Yrjö Jahnsson Foundation, Helsinki, 1965. Reprinted in *Essays in the Theory of Risk-Bearing*, Markham Publishing Co., 1971.

[2] Marco Avellaneda and Sasha Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, apr 2008.

[3] Alexey Bakshaev. Market-making with reinforcement-learning (sac). *Quantitative Finance*, 2020.

[4] Á. Cartea, S. Jaimungal, and J. Penalva. *Algorithmic and High-Frequency Trading*. Cambridge University Press, 2015.

[5] Javier Falces Marin, David Diaz Pardo de Vera, and Eduardo Lopez Gonzalo. A reinforcement learning approach to improve the performance of the avellaneda-stoikov market-making algorithm. *PLOS ONE*, 17(12), DEC 20 2022.

[6] Sumitra Ganesh, Nelson Vadori, Mengda Xu, Hua Zheng, Prashant Reddy, and Manuela Veloso. Reinforcement learning for market making in a multi-agent dealer market, November 2019.

[7] Bruno Gasperov, Stjepan Begusic, Petra Posedel Simovic, and Zvonko Kostanjcar. Reinforcement learning approaches to optimal market making. *MATHEMATICS*, 9(21), NOV 2021.

[8] Bruno Gasperov and Zvonko Kostanjcar. Market making with signals through deep reinforcement learning. *IEEE ACCESS*, 9:61611–61622, 2021.

[9] Lawrence R. Glosten and Paul R. Milgrom. Bid, ask, and transaction prices in a specialist market with heterogeneously informed traders. *Journal of Financial Economics*, 14(1):71–100, 1985.

[10] Olivier Gueant. Computational methods for market making algorithms. In M Ehrhardt and M Gunther, editors, *PROGRESS IN INDUSTRIAL MATHEMATICS AT ECMI*, volume 39 of *Mathematics in Industry-Cham*, pages 509–515, 2022. 21st European Conference on Mathematics for Industry, ECMI, Univ of Wuppertal, Wuppertal, GERMANY, APR 13-15, 2021.

[11] Olivier Guéant. Optimal market making, May 2017.

[12] Jiafa He, Cong Zheng, and Can Yang. Integrating tick-level data and periodical signal for high-frequency market making, 2023.

[13] Alice Jerome and Michael Brown. Incorporating domain knowledge in financial reinforcement learning. *Journal of Financial Engineering*, 9(3):215–240, 2022.

[14] Joseph Jerome, Leandro Sanchez-Betancourt, Rahul Savani, and Martin Herdegen. Model-based gym environments for limit order book trading, 2022.

[15] Maureen O'Hara. *Market Microstructure Theory*. Blackwell Publishers, 1995.

[16] Yagna Patel. Optimizing market making using multi-agent reinforcement learning, 2018.

[17] John W. Pratt. Risk aversion in the small and in the large. *Econometrica*, 32(1-2):122–136, 1964.

[18] Matias Selser, Javier Kreiner, and Manuel Maurette. Optimal market making by reinforcement learning. *Quantitative Finance*, 2021.

[19] Peter Selser and Emily Clark. Stylized facts and their role in financial market simulations. *Journal of Computational Finance*, 14(2):125–145, 2021.

[20] Tianyuan Sun, Dechun Huang, and Jie Yu. Market making strategy optimization via deep reinforcement learning. *IEEE ACCESS*, 10:9085–9093, 2022.

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2018.