Applied Intelligence (2023) 53:22249–22266
https://doi.org/10.1007/s10489-023-04647-9

Automated market maker inventory management with deep reinforcement learning

Óscar Fernández Vicente[1] · **Fernando Fernández**[1] · **Javier García**[2]

Accepted: 18 April 2023 / Published online: 24 June 2023

Abstract
Stock markets are the result of the interaction of multiple participants, and market makers are one of them. Their main goal is to provide liquidity and market depth to the stock market by streaming bids and offers at both sides of the order book, at different price levels. This activity allows the rest of the participants to have more available prices to buy or sell stocks. In the last years, reinforcement learning market maker agents have been able to be profitable. But profit is not the only measure to evaluate the quality of a market maker. Inventory management arises as a risk source that must be under control. In this paper, we focus on inventory risk management designing an adaptive reward function able to control inventory depending on designer preferences. To achieve this, we introduce two control coefficients, AIIF (*Alpha Inventory Impact Factor*) and DITF (*Dynamic Inventory Threshold Factor*), which modulate dynamically the behavior of the market maker agent according to its evolving liquidity with good results. In addition, we analyze the impact of these factors in the trading operative, detailing the underlying strategies performed by these intelligent agents in terms of operative, profitability and inventory management. Last, we present a comparison with other existing reward functions to illustrate the robustness of our approach.

Keywords Reinforcement learning · Market-making · Stock markets · Inventory risk management · Stochastic dynamic control · Artificial intelligence

1 Introduction

Every stock market is composed of different types of participants with diverse goals [1]. Retail traders, big corporations, arbitrage systems [2], and high-frequency operators [3] are some examples. Market makers (MM) are a specific type of participant whose main objective is to provide liquidity to the stock markets (Fig. 1). They accomplish this task by streaming buy and sell orders at different sides of the order book (OB), feeding this OB with more alternative prices to trade with. The OB is an electronic list of buy and sell orders that is constantly updated incorporating the new arriving limit orders, and removing those orders that have been already matched as well. Hence, it contains the available limit orders at every moment, in terms of price and amount of stock, of all the participants. The market-making task is especially relevant in those assets with low liquidity, as it allows traders to have extra price levels to operate with. Additionally, MMs usually earn trading profits by buying and selling these stocks. These trading profits come mainly from the bid-ask spread [4] of the stock market. In a nutshell, bid-ask spread represents the difference between the lowest price a seller is willing to buy the stock and the highest price a buyer is willing to buy. Every stock has a specific bid-ask spread depending on the liquidity of that asset (available orders), and it changes over time according to different circumstances. It is usually higher in markets with low liquidity. Regarding MM, the higher bid-ask spread the asset has, the more trading profits it will earn per operation.

In trading, holding inventory usually increases the risk of any agent, as its Mark-To-Market (MtM) is more exposed to price movements. MtM is a method of measuring the real value of an agent, and it is calculated by combining the avail-

✉ Óscar Fernández Vicente
oscar.f.vicente@alumnos.uc3m.es

Fernando Fernández
ffernand@inf.uc3m.es

Javier García
franciscojavier.garcia.polo@usc.es

[1] Universidad Carlos III, Av. Universidad, Leganés 28911, Comunidad de Madrid, Spain

[2] Universidad Santiago de Compostela, Praza do Obradoiro, 0, Santiago de Compostela 15705, A Coruña, Spain

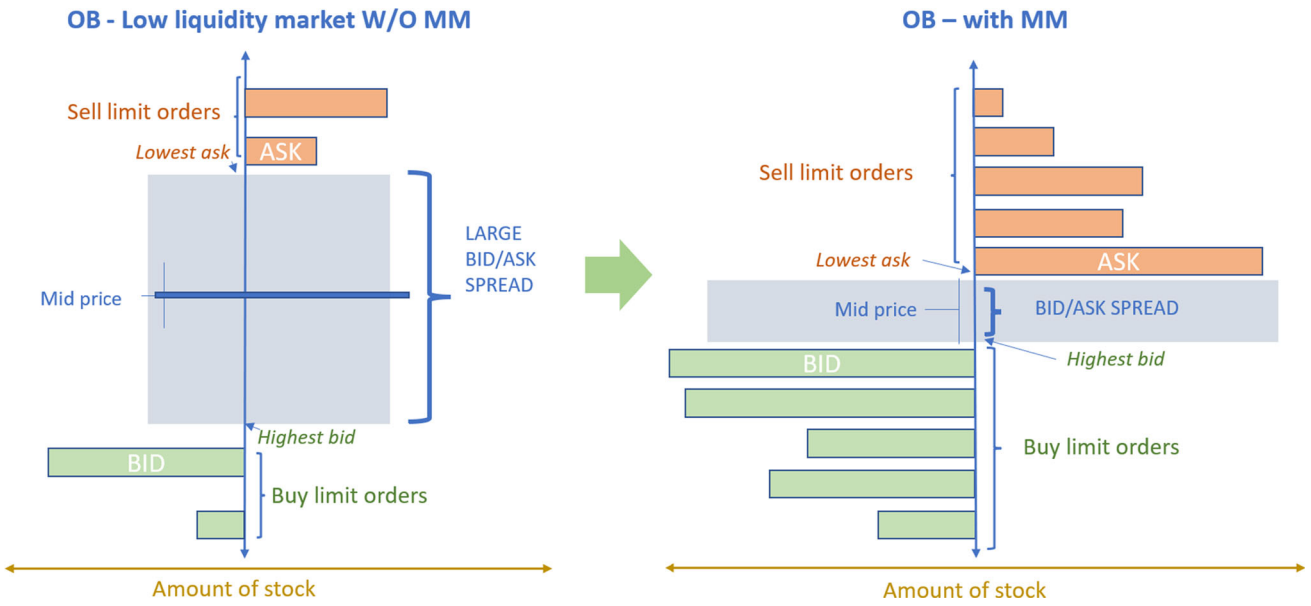

Fig. 1 Example of the impact of a MM in a low liquidity asset order book. A MM usually populates the Order Book allowing the rest of the Market's participants to have more prices to trade with

able cash plus the value of the inventory as if it was sold at that specific moment. It is critical to have good management of the accumulated inventory along the trading operative to mitigate possible risks, like a sudden devaluation of the asset.

Therefore, the definition of a good market-making strategy relies on two main goals: Increasing profits by buying and selling stocks, while reducing the risks of holding inventory. We can define the problem as a multi-objective challenge, where both objectives are inversely related. In classical multi-objective optimization problems, we have alternative ways to solve this, such as finding a Pareto front to take the non-dominated policies according to a prior utility function [5–7]. Focusing on RL, the usual approximation of solving multi-objective problems is to define a reward function that aggregates both objectives as a single scalar value. The definition of this reward function usually requires the domain's understanding and prior knowledge of the utility function [8].

Market-making inventory management through Reinforcement Learning (RL) is a topic that has been addressed by some authors, as it is broadly described in Section 2. However, most of those solutions rely on adding some sort of penalty term to the reward function, an action that reduces the reward obtained at every time step according to the full inventory value or its value variation. In some cases, this penalty is computed at the end of the trading sessions by removing the full value of this inventory. Nevertheless, both kind of approaches does not take into consideration the changing conditions of the MM along the trading session, penalizing linearly the accumulated inventory.

To address this issue, we define two coefficients that modulate the inventory exposure: *Alpha Inventory Impact Factor* (AIIF) and *Dynamic Inventory Threshold Factor* (DITF). These two coefficients, widely described in Section 4, determine the behavior of the MM through the definition of a dynamic inventory threshold, managing the inventory dynamically at every time step through its contribution to the reward function. Furthermore, we demonstrate this approach has good results in terms of MtM and inventory risk management, detailing variables such as profitability, inventory distributions, and the semantics of the policies adopted by the RL agent.

Therefore, in this manuscript, we show how to design an intelligent RL market maker agent that can be profitable while managing inventory risk dynamically and adaptively, through a novel reward function. This agent takes into account its changing situation, in terms of liquidity, adapting its policy through the trading sessions. Additionally, we analyze in-depth the underlying policies to understand the behavior of different risk-averse agents. We also run a comparison against other existing reward functions to evaluate the performance of our solution.

Regarding the organization of the remainder of the paper, Section 2 discusses the current state of art. Section 3 introduces Reinforcement Learning as a control technique. Section 4 details our MM agent, including the proposed reward function. Section 5 describes the environment and all the details of the experimental setup. Section 6 analyses the results of the proposed MM, including a comparison with other existing approaches. Finally, Section 7 presents all the conclusions derived from our study. The datasets generated during the current study are available in the GIT repository, https://github.com/ofvicente/automated_mm_paper.

2 Related work

Market-making strategies have been broadly studied not only from a classical approach but even using cutting-edge techniques, including RL [9]. In the following subsections, we will take a look at both of them, to have a better understanding of the main problems addressed and the proposed solutions.

2.1 Market-making as a classic financial problem

From a financial point of view, this problem has been addressed for many years and even decades. One of the most important contributions and a reference was made by Avellaneda and Stoikov [10], where they combine the Ho and Stoll [11] framework with the microstructure of the OB, based on the current inventory and a calculated reservation price. In a nutshell, they try to find a symmetrical optimal spread based on a reference price and a pre-defined inventory strategy. The reference price is conditioned by the imbalance between buy and sell orders placed in the OB. This reservation price is calculated with the following terms: (i) the current mid-price of the stock, (ii) the inventory they want to hold under the strategy, (iii) a risk aversion coefficient, and (iv) the volatility of the traded asset. With this reservation price, and including another additional term that represents the liquidity (density) of the OB, they calculate an optimal spread that fits the initial strategy.

This work by Avellaneda and Stoikov has inspired other authors, such as Gueant et al. [12], who introduce an evolution with a new change of variables based on Hamilton-Jacobi-Bellman equations and a limited inventory to improve the model. More recently, the same author introduced another solution based also on Avellaneda's framework where they face the problem of multi-asset market-making, addressing the challenge of managing correlated assets [13]. All these works at the end of the day propose an evolution of this first work, by focusing on certain points.

Excluding Avellaneda's approaches, there are other works that deal with this problem. For example, regarding HFT[1] and its underlying challenges, some authors [14] propose specific solutions based on a previous liquidity analysis of these environments. The analysis of liquidity as a predictor of future movements is not only used in market-making tasks but also gives us insights into the underlying short-term price intentions.

As we have seen, many different paths have been followed to address the market-making challenge. Some of them are based on the order book, while others are more focused on liquidity, etc.

[1] High-Frequency Trading

2.2 Market-making as a machine learning problem

Apart from classical and pure mathematical approaches presented in Section 2.1, there are also some interesting works in the context of Machine Learning (ML). Recently, ML has been successfully used for solving a wide variety of tasks [15–19].

With respect to ML in trading, works can be divided into two different categories. The first one relates to Supervised Learning (SL), where the researchers try to model or predict some value of the market itself. And a second one, based on RL, where the technique and method are quite different, trying in this sense to model an agent that can participate in the market following a learned policy.

Regarding the first category, SL is used in some works [20] to design trading signals based on previous data. SL signals are usually the trigger for different trading actions. These signals are commonly generated using different sources: from the OB, news, and other alternative data sources. Additionally, SL is usually used to predict future values (such as future short-time stock prices [21]), valuable information that helps the MM to find better strategies.

While SL may be successful in achieving specific goals, RL is more suitable for tasks that involve decision-making or sequential control. And market-making (with inventory control) may be more accurately included in this second group. In this sense, there are also multiple lines of work. The main purpose of these works is to design some sort of intelligent agent able to act as a profitable MM, by defining (generally) a good reward function. However, many of the market-making RL strategies found do not take into account inventory risk, focusing only on maximizing the PnL[2]. For instance, Vicente et al. [22] design a profitable MM agent that competes in a multi-agent environment, optimizing only the MtM. They also study the transferring of certain learned policies among different environments. Other works are also based on the use of trading signals, such as mentioned in SL above. For instance Lokhacheva et al. [23] implement a Q-Learning agent based on these trading signals, where EMA[3] and RSI[4] are used basically to define the space state.

All these mentioned works do not consider inventory risk, which is an important weakness. Regarding those RL researchers that make a special focus on inventory management and the derived risks, we find two main approaches:

- Some works penalize the changes in the inventory held during the trading session. In this regard, there are at least two main alternatives: (a) apply a penalty only on inventory changes [24, 25], and (b) apply a penalty on the entire inventory held [26, 27] to discourage any kind of

[2] PnL: Profit and losses, earnings.

[3] Exponential Moving Average

[4] Relative Strength Index

asset accumulation. In both cases, a penalty term impacts directly the reward function in a fixed and predefined size. An aversion coefficient is usually included.
- Other approaches [28–30] penalize the inventory held by the RL agent only at the end of the episodes. In this sense, CARA[5] utility is used frequently to determine risk aversion.

All these works rely on static reward functions. They do not consider the changing situation of the trading (MM) agent. In this sense, it is worth remarking that having a very restrictive reward function usually impacts negatively on the total returns, as the agent behaves more cautiously. Moreover, a fixed penalty term, similar to the reward functions of the works presented above, may have similar undesirable results, as the MM agent is not able to adapt to the evolving circumstances. In most of the aforementioned works, this is an extended approach, and therefore, a weakness. The MM's Mark-to-Market changes along a trading session according to its operative, not only in terms of the total value but in its cash/inventory value proportion. For instance, as long as the MM increases its MtM following a profitable strategy, increasing its cash, it seems logical that it should have more room to inventory if this enables the agent to perform a more profitable policy. According to this, our agent could be more aggressive in searching for profits while containing risks. This adapting penalty term is the base, and one of the strengths, of our contribution.

3 Reinforcement Learning

Reinforcement learning [31] is a type of machine learning in which agents learn to perform tasks by interacting with their environment. These agents optimize their actions to learn the best way to complete a task. To achieve this, RL agents receive positive or negative rewards from those performed actions, adjusting their policies to obtain the best present and future rewards. These tasks are usually described as a Markov decision process (MDP), as they can be represented as a sequence of states. They are expressed by the following tuple: $\mathcal{M} = \langle S, A, T, R \rangle$, where S corresponds to the state space representation, A to the action space, T to the transition function between states, and finally R corresponds to the immediate rewards earned after every action taken. RL has been successfully applied to both deterministic and stochastic environments. Financial Markets are generally considered a stochastic domain because many players interact by buying and selling stocks in a common environment with limited/imperfect information about the underlying dynamics and intentions. This imperfect information translates into uncertainty for all the uninformed participants. In addition, if we put trading fees aside, trading can be considered a zero-sum game, where profits and losses are balanced among all the agents. Therefore, being consistently profitable is not an easy task. Some variables are generally used to estimate the stock environment at time t, such as price dynamics, trading volume, order book status, and/or many indicators derived from these basic inputs. Trading agents must find a profitable policy π using these variables.

There are many different approaches to design an RL control agent. In this paper, we use a model-free [32, 33] off-policy and value-based RL agent called Deep Q-Network DQN [34]. DQN is considered an evolution of Q-learning [35] algorithm, where a deep neural network (NN) is used to approximate the value function instead of a Q-table. This is necessary when we have multivariate continuous space states like in the solution presented here. Basically, value-based algorithms estimate the expected value of being in a specific state $V(s)$, or even the combination of state and action $Q(s, a)$, to find the best policy and the most profitable behavior in terms of present and future rewards.

In particular, DQN relies on a Q-function where every action taken at one specific state following the policy π has an expected value $Q^{\pi}(s, a)$. This value $Q^{\pi}(s, a)$ is defined as the following expected return (1):

$$Q^{\pi}(s, a) = \mathbf{E}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} \| s_t = s, a_t = a, \pi\right] \tag{1}$$

where γ represents the discount factor. This factor determines the weight given to future rewards. Therefore, when $\gamma = 0$ only immediate reward is considered. The return is considered expected due to the non-deterministic nature of this specific domain.

DQN is not an episodic algorithm, which means that rewards can be computed and policies can be updated without waiting until the end of an episode. However, it bootstraps the rewards along the different time steps according to the former function. Additionally, this algorithm includes that improves learning. First, it has two identical deep neural networks instead of one. One neural network makes updates constantly with new experiences, while the second one synchronizes every n time step to avoid instabilities taking actions. Second, there is a replay buffer that stores all the experiences and is used to train the neural network. This replay buffer dispatches experiences randomly, removing correlations in the observations sequence.

[5] Constant Absolute Risk Aversion

4 New RL model for inventory management

In order to control the MM inventory, an RL agent has two main levers to interact with. On one hand, it can increase buys or sells if it wants to increase or reduce inventory respectively. Both actions can be achieved by populating more competitive buy over sell prices, or vice versa, depending on the desired goal. With this "skewness" the MM inventory will be increased or decreased accordingly along the trading session due to that imbalance. On the other hand, every agent has the option of selling stocks (or buying) in the market at every moment. This action, usually known as hedging, allows the agent to reduce almost instantly the desired level of inventory held. The main problem with this alternative is that every hedging action is executed by matching a market-order. This means that it has a cost, as every market-order has to pay the current market spread.

Restrictive penalty terms or risk-averse policies usually impact negatively on final returns, as the agent has to be more cautious when operating. Generally, low restrictive penalty terms will translate into higher and even more volatile returns. Finding an equilibrium between both goals, to earn profits and to manage risks, should be the main objective of every operator. According to this, non-adapting penalty terms as stated in previous works cannot be optimal by definition. The idea behind our reward function is that the MMs should be able to hold, and thereby, manage the greater or lesser amount of inventory in response to changing conditions of the agent during the trading session. Therefore, if a MM is operating along a trading session its situation in terms of MtM and Cash/Inventory value evolves as well. This changing situation should be considered to adapt the inventory holding strategy in order to improve the profits. For instance, the trading strategy should not be the same when having a 50%-50% cash/inventory value ratio, 80%-20%, or 20%-80%. In the first case, an 80% cash and 20% inventory ratio, we could allow the inventory to be increased if that action could increase our profits as well, caused by following a less restrictive policy. The opposite should happen in a 20%-80% scenario, where inventory risk should be managed more carefully according to that imbalance. These proportions may also change during the trading session, forcing the MM to adapt its policy accordingly aiming to find the best policy. This is why the MM operating strategy should not only be a matter of keeping inventory always close to 0 units, or a matter of applying always a linear penalty term but finding a balance that improves profits while managing inventory risk.

In the following section, we describe how these concepts were incorporated into the reward function. After that, the state and action spaces used in this work are also described. Finally, we detail the training flow performed by the market maker.

4.1 Reward function

Learning good policies in RL is usually tied to the design of appropriate reward functions. Reward function engineering is, in fact, a matter of research by itself [36–39]. The proposed reward function presented here aims to find the highest trading profitability in terms of returns while controlling inventory dynamically using a specific penalty factor, as stated previously. It is important to remark on again the dynamic nature of our approach, as the penalty factor will adapt to available cash and inventory value at every moment. To learn this adapting policy, we have to design a proper reward function able to be profitable while controlling risks, by adjusting the operative to the operator's risk aversion and also adapting to the changing situation of the MM.

To achieve that, we introduce the following key concepts:

Definition 1 (Dynamic Inventory Threshold Factor (DITF)) factor that determines the reference ratio between the value of cash and inventory, which must be respected by the agent throughout the trading session (7).

According to this term, established at the beginning of the training stage, if we want that the MM agent follows the 1:1 ratio between cash value and inventory value as a reference, we should assign $DITF = 1$. This factor defines a dynamic threshold (thr) throughout the trading session, a threshold that determines the maximum inventory limit in which MM must fit at any given time.

Definition 2 (Alfa Inventory Impact Factor (AIIF)) factor that modulates the agent's risk aversion by including a coefficient in the penalty term of the reward function (as in (6))

Regarding this factor, the higher this coefficient is, the stronger penalty the agent will receive from holding inventories out of the allowed threshold, previously defined.

These two concepts will help the agent not only to control risks while seeking profitability but to achieve it intelligently. We know that both objectives (earn profits and control inventory) are opposite, and we must find some strategy that optimizes both. To do that, we could have picked some static reward function that just "scalarise" both goals according to some fixed weight, as other works do. But it is important to remark that as long as the market maker operates in a trading session, the proportion between cash and inventory value changes according to the results of the performed strategy. Hence, it could admit more inventory in some cases than others, while respecting an overall risk aversion ratio. The two previous factors introduced above help us in this task. AIIF factor establishes a general risk aversion threshold that would be respected along the trading session; while DITF will modulate the overall risks depending on that evolving cash/inventory value ratio. These concepts are included in the penalty term of our reward function.

We define the reward function as follows:

Definition 3 (The Reward function for Inventory Management (RIM)) is a reward function $R : O_i \times Pnl_i \times Hc_i \times Pny_i \rightarrow \mathbb{R}$, computed as defined in (2),

$$R_i = O_i + Pnl_i - Hc_i - Pny_i \tag{2}$$

The different terms of the reward function are described as follows:

- (i) O_i: The trading profits obtained by the MM at time step t_i result from buys or sells. More precisely, as shown in (3):

$$O_i = \sum_{x=0}^{n} (B_x, S_x) * SpM_{(b,s)i} \tag{3}$$

  where (B_x, S_x) refers to the number of stocks traded (buys or sells) by investors with the MM, and $SpM_{(b,s)i}$ is the buy or sell spread quoted by the MM respectively at that time step t_i. Thereby, the higher the spread quoted by the MM, the higher the profits earned, as previously mentioned.
- (ii) Pnl_i: The inventory held by the market marker will gain or lose value according to mid-price variation in every time step t_i. Therefore, if the price rises up the inventory value will rise as well, and vice versa. This is what Pnl_i reward term refers to, computed as shown in (4):

$$Pnl_i = inv_i * \Delta Pr_i \tag{4}$$

  where inv_i is the quantity of stock held by the MM (positive or negative) at t_i, and ΔPr_i the difference between current and previous stock prices: $\Delta Pr_i = Pr_i - Pr_{i-1}$.
- (iii) Hc_i: The market maker has the option to reduce its inventory instantly by buying or selling it at every time step. When doing this, it pays a hedging cost as a penalty, as it is executed through a market-order. Hedging costs Hc_i are equal to the amount of inventory hedged multiplied by the market spread Sp_i. It is computed as shown in (5):

$$Hc_i = Hg_i * Sp_i \tag{5}$$

  where Hg_i is the amount of inventory hedged, and Sp_i is the current market spread, at t_i.
- (iv) Pny_i: The last term concerns the penalty of holding inventory. This term is especially relevant in our approach as it drives the policy according to the inventory risk. As introduced previously, market makers should carry the least amount of inventory possible, according to their liquidity situation, to avoid risks caused by price volatility. To achieve this, it is necessary to establish some restriction or penalty that discourages the agent to accumulate inventory during the trading session. Particularly, we have opted for a dynamic term that can adapt to the MM inventory and cash values at every moment. This term Pny_i impacts the reward as it penalizes the inventory that the MM holds out from a specific range. This range, or threshold, is not a fixed value, but instead, it is dynamically adjusted taking into account the desired proportion between the MM cash value (liquidity) and the average value of the inventory held in the last n steps. Hence, we define the penalty term as in (6):

$$Pny_i = AIIF * min\{|R_i|, |R_i * \frac{\overline{inv_i}}{\overline{thr_i}}|\} \tag{6}$$

where:

$$thr_i = DITF * |\frac{cash_i}{\frac{1}{n}\sum_{x=t-n}^{t} mid_i}| \tag{7}$$

$$\overline{thr_i} = \frac{1}{n}\sum_{t-n}^{t} thr_x \tag{8}$$

$$\overline{inv_i} = \frac{1}{n}\sum_{t-n}^{t} inv_x \tag{9}$$

The dynamic threshold thr_i (7) is calculated by dividing the MM cash value at t_i by the average of the last n stock mid-prices mid_i. The resultant value is multiplied by $DITF$, the *Dynamic Inventory Threshold Factor*. This factor defines the proportion between cash and inventory value mentioned above. This proportion is between the cash value at t_i and the average inventory held in the last n time steps. We use an average instead of the spot value to reduce the impact of price volatility and to stabilize convergence. The whole penalty is modulated using the first coefficient $AIIF$, our Alpha Inventory Impact Factor.

According to (6), it is important to remark that this penalty term behaves as a piecewise function, where this penalty increases linearly from $inv_i = 0$ until the dynamic threshold thr_i is reached. The idea behind this is to have a smooth penalty term that discourages proportionally the accumulation of inventory.

In the experimental Section 6 we describe the impact of $AIIF$ factor on profitability, inventory management, and policies.

4.2 State and action spaces

As with every RL problem definition, some key elements have to be defined apart from the reward function. On one hand, the agent must have a good understanding of the environment. This is achieved by defining a proper state space that contains enough valuable information to select the best possible actions at every time step. On the other hand, it is necessary to determine the action space, hence, all the actions that can be performed by the agent. State and action spaces have been designed based on the previous work of Ganesh et al. [40], with some adaptations such as the action space's discretization. All these elements are detailed as follows.

4.2.1 States

The observation state of our agent consists of 8 features: (i) the number of stocks bought in the previous time step, (ii) the number of stocks sold in the previous time step, (iii) the current inventory size (positive or negative), (iv) the inventory size in the previous step, (v) the stock mid-price variation between current and previous time step ($\Delta Pr_i = Pr_i - Pr_{i-1}$), (vi) the current reference bid-ask spread, (vii) the bid-ask spread in previous time step, and (viii) the total amount of stock traded (V_i), known as $volume$, by the MM in previous time step. Other space states have been also tested, such as more reduced space states, with worse results.

4.2.2 Actions

The MM proposed agent has three possible actions to execute. First, it can quote a buy price and/or a sell price. These prices, related to the current market spread, will determine how many operations are matched and how much trading profits the agent will earn. In addition, it can decide how many inventory units hedge through a market-order. To perform these actions there are three variables that MMs can interact with in every time step t_i: (i) buy spread B_i^s, (ii) sell spread S_i^s, and (iii) amount of inventory hedged H_i^s. With these three variables, a discrete action space has been defined.

Hence, buy and sell actions will consist of picking one η in the following list of evenly distributed values respectively: $\eta_{(b,s)i} \in \{-1, -0.8, -0.6, ..., 0.6, 0.8, 1\}$.

Therefore, streamed buy/sell spreads (prices) correspond to (10):

$$B_i^{\mathrm{Sp}}, S_i^{\mathrm{Sp}} = Sp_i * (1 + \eta_{(b,s)i}) \tag{10}$$

where Sp_i is the current price spread streamed by the market, and B_i^{Sp}, S_i^{Sp} are the spread that MM quotes at time t_i. Both buy and sell $\eta_{(b,s)i}$ can be equal or different, incurring in asymmetry in the latter case. This asymmetry, also known as skewness, allows MM to balance between buys and sells if needed.

In addition to buy or sell actions, hedging action will consist of picking another η from the list $\eta_{hi} \in \{0, 0.25, 0.5, 0.75, 1\}$. Depending on the specific η_{hi} selected by the agent, its inventory will be reduced proportionally by 0% to 100% in the following time step t_{i+1}. Thereby, as shown in (11):

$$inv_{i+1} = inv_i * \eta_{hi} \tag{11}$$

where inv_i is the inventory held by the market marker at t_i. Hedging action is not free, as it will incur specific costs as described in Section 4.1.

In conclusion, the action space consists of 11 buy $\times$ 11 sell $\times$ 5 hedge η, represented by the tuple $a_i = (\eta_{bi}, \eta_{si}, \eta_{hi})$. We have opted for this solution instead of using a continuous action space to increase the convergence rate while sacrificing a more concrete value, which we do not consider necessary for this exercise. Additionally, using relative values instead of absolute figures helps our agent to generalize better if the domain values change. As an example, having at t_i a market spread of $Spi_t = 7$, an inventory of $inv_i = 150$ units, and the following η returned by the agent $a_i = (0.8, -0.2, 0.5)$, the MM would stream the following values: $B_i^{\mathrm{Sp}} = 7 * 0.8 = 5, 6$, and $S_i^{\mathrm{Sp}} = 7 * -0.2 = 6.8$. Additionally, inventory would be reduced to $inv_{i+1} = inv_i * 150 = 75$ units, regardless of inventory increases or decreases due to the rest of trading operations.

4.3 Market Maker's training algorithm

To clarify the training process, Algorithm 1 presents the pseudo-code. This is an adaptation of the original DQN algorithm [34] to the market maker learning problem. According to the training flow, the market maker waits until new market information arrives (5, 6). With this information, the state space is updated and an action is selected. As DQN follows an epsilon-greedy strategy, the action is selected according to an epsilon value (7,9). Every action consists of selecting 3 different values as shown in previous Section 4.2.2: A buy price, a sell price, and a hedging percentage. Buy and sell prices are streamed back into the market (12), and at the same time, the inventory is hedged according to the percentage selected in the action(13). Investors, once having all the different prices, launch buy and sell orders against all the available MMs (14). At that moment the market maker computes its rewards, according to the reward function described in Section 4.1 (15). All the transitions are stored in a replay-buffer(17), transitions that will be used every 200 time steps to train the MM neural network(19).

Algorithm 1 DQN MM training flow

Initialize memory $\mathcal{D} \leftarrow \emptyset, t \leftarrow 0$
for *simulation* **do**
  Init simulation
  for every time step t_i in the simulation **do**
    get current market spread Sp_i from market data.
    initialize state, s_i
    if $rand < \epsilon$ **then**
      a_i=best_action(buy, $sell$ and $hedge\ \eta$)
    else
      a_i=random_action(buy, $sell$ and $hedge\ \eta$)
    end if
    execute action a_i (stream buy and sell spreads)
    execute hedging from the previous time step Hg_i
    Investors launch random buy/sell operations on the cheapest MM
    MM computes rewards r_i and observes s_{t+1}.
    MM applies ϵ decay
    store transition $\langle s_i, a_i, s_{i+1}, r_i \rangle$ in buffer $\mathcal{D}$
    $t \Leftarrow t + 1$
    if $t\%200 == 0$ **then**
      retrain Training NN with 1.024 random transitions from $\mathcal{D}$
      Target NN $\Leftarrow$ Training NN
    end if
    $s_i \Leftarrow s_{i+1}$
  end for
end for

5 Experimental setting and methods

Due to continuous state space, and the non-episodic nature of our approach, we have opted for using a Deep Q-Network for the MM agent. As introduced in previous sections, the Q-function Q(s,a) is approximated by a DQN, by using a fully connected network. In fact two, instead of one, identical neural networks are used: a training NN and a target NN. This improves the stabilization in the learning stage [34]. These neural networks have been defined with a total of 8 state space variables in the input layer, and 605 possible actions in the output layer. Furthermore, the network has 3 hidden layers with 32 neurons each one. The input and the three hidden layers have ReLu as the activation function. The output layer, however, uses a linear activation function. For every given state the neural network approximates the reward of every action, selecting the action with the best-expected reward. Epsilon-greedy is the exploration strategy. Other network architectures have also been tested, such as 3×32, 3×64 and 5×64 (layers $\times$ neurons).

Regarding the training stage, the neural network is fitted every 200 time steps (1 epoch). Experience replay is used to improve the training. According to this, all the experiences tuples $e_i = (s_i, a_i, r_i, s_{i+1})$ are stored in a replay-buffer of 1M size. Every epoch, the training neural network takes mini-batches of 1.024 random experiences from this replay-buffer fitting the weights using back-propagation. Adam has been chosen as the learning optimizer and MAE as the loss metric. The learning rate is set to $lr = 0.01$. The discount factor has been set to $\gamma = 0.6$. These hyperparameters have been selected after testing other alternative values.

Regarding the trading environment, there are three types of experimental MMs competing with each other in every simulation, as follows:

- **1 DQN market maker** (DQN MM): The Deep Q-Learning (DQN) intelligent market maker proposed in this paper.
- **1 Random market maker** (Random MM), a market maker with uniform random spreads and hedges on every time step. Same action space as DQN.
- **1 Persistent market maker** (Persistent MM), a market maker with fixed uniform random spread and hedges over every simulation. The spreads and hedge are selected at the beginning of each simulation and kept until the simulation is finished. Same action space as DQN.

All the experiments have been conducted in a simulated event-based trading environment called ABIDES [41]. This kind of simulated environment allows us to run multiple setups, increasing the variance of the data and reducing overfitting, among other benefits. ABIDES, in its more recent versions, is being integrated as an OpenAI gym environment [42]. In a nutshell, ABIDES is an agent-based simulation tool where the defined agents interact with each other in a trading session, as a real market does. Every agent included in the market has its own strategy (non-informed traders, market makers, momentum agents, etc), and the natural interaction among them is what defines the price movement of the negotiated assets. This tool is being used in many trading studies [43–45] as it is able to simulate very realistic trading Markets. It allows the generation of diverse market configurations according to the needs of the specific study. In our work, we have tried to generate a "standard" market in terms of price volatility. With our setup, we were essentially looking for credible price movements, while keeping the stochastic essence of every trading environment. According to this, the market simulation configuration in which the experiments were conducted had the following participants/setup:

- 100 noise agents: These agents place orders in the OB in a random direction of fixed size.
- 10 value agents. They have access to fundamental time series and operate according to variations of mid-price related to their mid-price forecast.
- 10 Momentum agents. This kind of agent operates when 50 and 20 steps moving averages are crossed.
- 1 adaptive POV agent. This agent provides certain liquidity to the synthetic market by placing orders at fixed intervals.

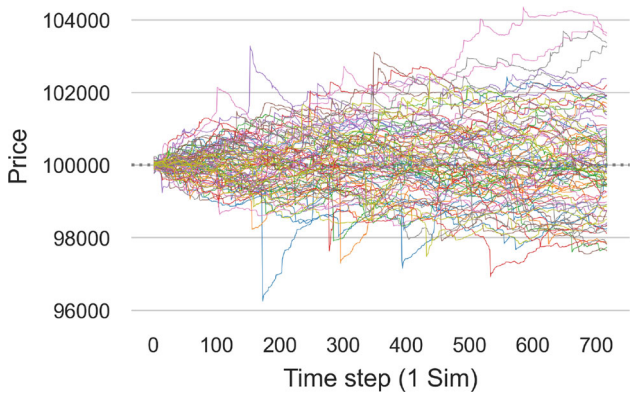

Fig. 2 Mid-prices evolution sample. This figure illustrates how the price changes along a random sample of different simulations., showing the stochastic nature of ABIDES simulator

- 1 exchange agent. Finally, this agent orchestrates the interaction and integration of all the agents and the order book.

The market generated from the interaction of these agents has enough liquidity to allow for different spreads along the simulations. Additionally, the returns are reasonable and the price movements are suitable for a standard financial market. To illustrate these points, Figs. 2 and 3 show different price trajectories and returns respectively across full experiments.

All these agents interact for two hours, determining a trading session. This interaction defines the movement of the stock price, its volatility, and also the reference market spread Sp. In addition to these agents, 50 investor agents interact solely with the market makers. These investor agents place random buy or sell orders with a fixed size. As greedy investors, they always choose the MM with the narrowest spread available in the OB, hence the cheapest one. If there

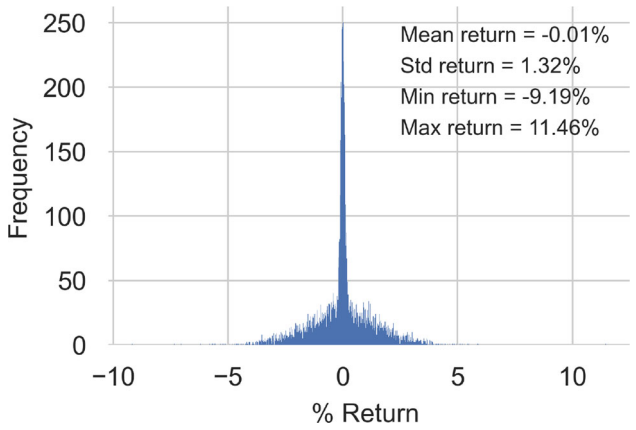

Fig. 3 Histogram of returns. The plot shows the returns (price variations) distribution at the end of all simulations. $Ret_t = (Price_t/Price_0) - 1$

were many MM with the same cheapest B_i^{Sp} or S_i^{Sp}, the investor would pick one of them randomly.

As mentioned above, every training/testing simulation is based on a 2-hour market session (9:30h to 11:30h). One DQN MM agent, one Random MM, and one Persistent MM are included. Full experiments have a total of 150 independent simulations (trading sessions), with 5 total experiments per setup.

No trading costs have been considered. Asset prices evolve stochastically according to agents' natural interaction. All simulations begin with an opening price of $1.000,00.

The experiment results are evaluated in terms of the total MtM and cash/inventory value ratios at the end of the trading sessions.

6 Experimental evaluation

In this section, we describe all the experiments, from the agent training to the comparison with other existing reward functions. First, to evaluate the impact of different penalty factors, we train our DQN MM with 9 different setups (Section 6.1). This phase gives us a first idea of the learning and performance of the different MM setups. Once we have these DQN MM trained, we test them in new fresh environments (Section 6.2). Then, we analyze the test results in terms of profitability (MtM), inventory management, and cash/inventory value ratio as main metrics. Furthermore, we inspect the policies performed by the DQN in all the testing environments (Section 6.3). Finally, to compare our solution against other existing approaches, we set up additional experiments in which we evaluate these reward functions (Section 6.4), demonstrating how our proposed solution is able to generate better market maker policies.

6.1 Training the market maker

Our DQN MM competes against two additional MMs in the trading sessions: one random agent, and one persistent agent. To evaluate the DQN MM performance in terms of profitability and inventory risk management, multiple training setups have been launched applying different *Alpha Inventory Impact Factors* (AIIF). A total of 10 different AIIFs have been tested. Starting from $AIIF = 0$, where no inventory penalty is applied to the reward function, up to $AIIF = 100$ where inventory impact is severe. Therefore, $AIIF = \{0, 0.2, 0.5, 0.8, 1, 1.5, 2, 5, 10, 100\}$. In addition, the *Dynamic Inventory Threshold Factor* DITF, has been established in this experimentation with a fixed value of $DITF = 0.5$. We have chosen this value because we find it feasible to keep inventory value below 25% of total MtM at any time. All the MM starts with cash of 100.000,00 $ and

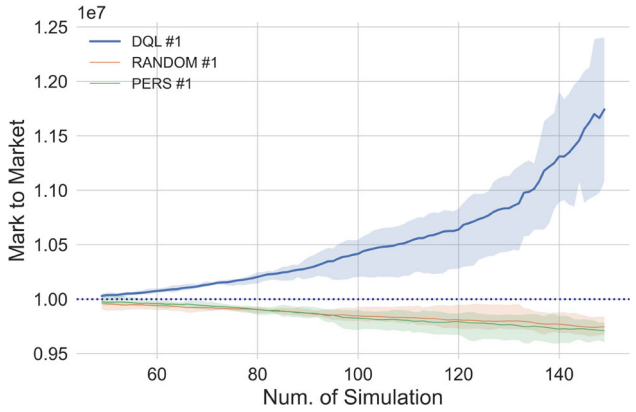

Fig. 4 Single training experiment, applying an AIIF = 0 in the reward function. The three competing agents are shown. It is noted how the DQN MM agent is able to learn along simulations how to be profitable (MtM), compared to the other two random agents

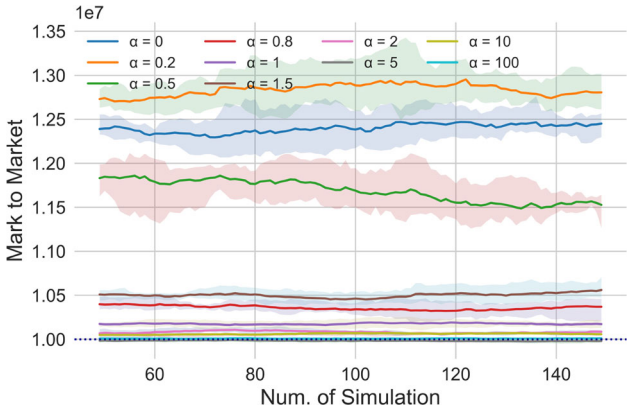

Fig. 6 DQN MM testing rounds, with different AIIF factors. Higher profits (MtM) are linked to lower $AIIF$ factors, as inventory control is more relaxed

0 units of asset inventory. A total of $\approx 612K$ gradient steps have been performed per single experiment.

First training results (with $AIIF = 0$) show very good performance in terms of Mark-To-Market, finding a more profitable policy than random and persistent MMs (Fig. 4). These two non-intelligent MM, in fact, end with lower MtM even than at starting point. Regarding alpha coefficients (AIIF), if we take a closer look at Fig. 5 where the different values of $AIIF$ are represented, it is remarkable to see how the slopes of the mark-to-market curves moderate as long as the AIIF factor increases.

6.2 Testing the policies

Once the agents have been trained using different AIIF factors, they are all tested on new experiments. At first sight, we notice they perform well in terms of profitability with almost all small AIIFs (Fig. 6). As long as alpha is increased, profitability is impacted. However, the inventory is controlled as well, as expected (Fig. 7).

If we analyze inventory distributions (Fig. 8), it is clear that as soon as the AIIF penalty factor increases the inventory also shrinks. Therefore, the distributions get narrower according to this coefficient as expected.

But not only AIIF factor plays a key role in managing the inventory risk. This management is also related to *Dynamic Inventory Threshold Factor* (DITF). This factor, which is established in this research to a value of 0.5, defines the proportion between cash and inventory value to be along the experiments. In Figs. 9, 10, and 11 we can notice how this threshold (in red) adapts dynamically along them according to the different AIIF factors, based only on the cash held by the MM at every moment, and the inventory value. Here we can see three examples, $AIIF = 0$ (no penalty), $AIIF = 1$ and $AIIF = 5$. As long as AIIF factor is increased, inventories are kept more strictly inside the thresholds. Note that

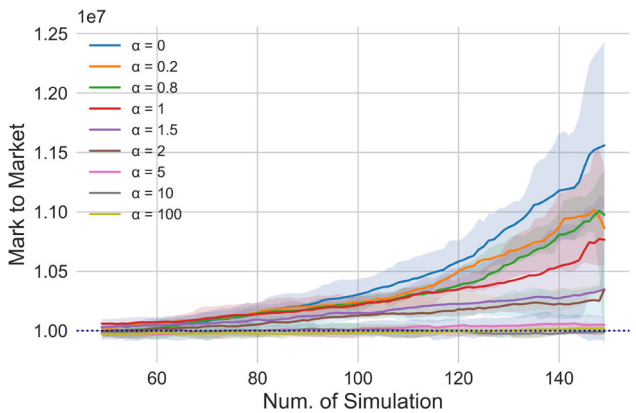

Fig. 5 DQN MM training results applying different $AIIF$. It is noticeable how MtM return reduces as long as $AIIF$ factor is increased. This effect has to do with the control of inventory. (Random and Persistent agents are not shown)

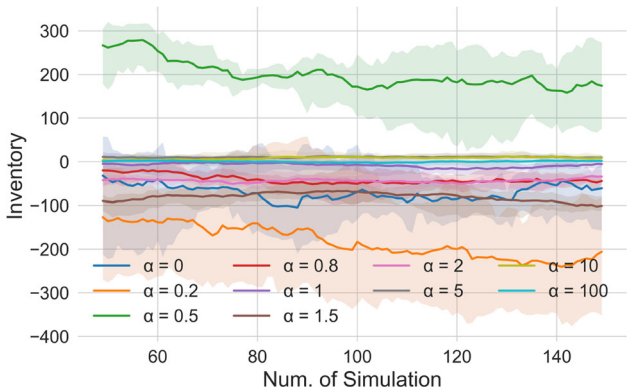

Fig. 7 DQN MM testing's rounds inventories, applying different $AIIF$ factors. In general terms, as long as the AIIF factors are increased, inventories are reduced as well, due to the controlling effect that this factor has on them

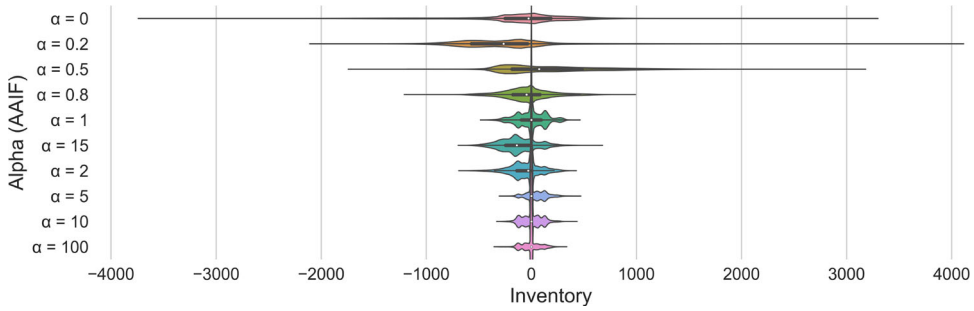

Fig. 8 DQN MM inventory distributions along different experiments. This Figure illustrates the different inventories held by every agent along the testing stage, according to their different AIIF factors. It is remarkable how, as long as the AIIF factors are increased, the inventory distribution and dispersion are reduced due to the inventory control nature of the factor

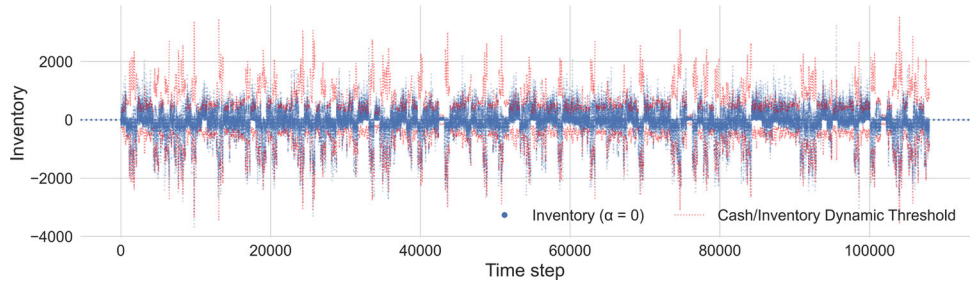

Fig. 9 DQN MM instant inventories held and thresholds along the testing experiment at every time step. Due to the lowest restrictive $alpha$ factor applied ($AIIF = 0$), noisy inventory thresholds and big variance in inventories are noted. The inventory control is improved with higher $alphas$, as shown in the following Figs. 10 and 11

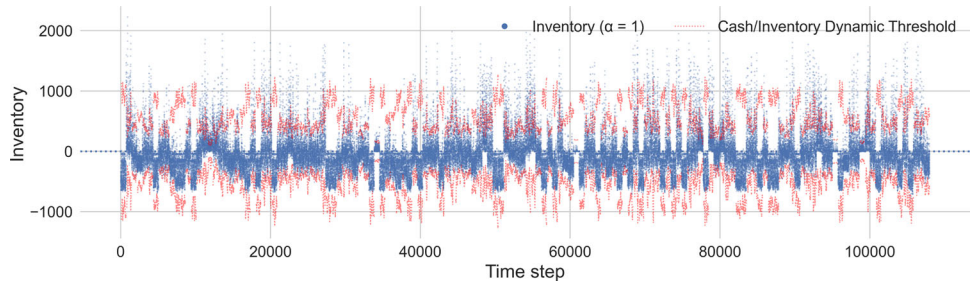

Fig. 10 DQN MM inventories and thresholds along the experiment with AIIF = 1. More stable thresholds are shown compared to the lower AIIFs of the previous Figure. In addition, the inventories are also more constrained

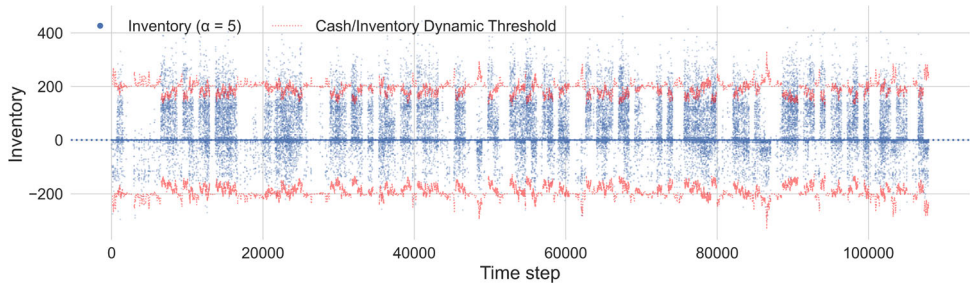

Fig. 11 DQN MM inventories and thresholds along the experiment with AIIF = 5. Here, extremely stable and low thresholds with very constrained inventories inside these thresholds (note different range values than Figs. 9 and 10). For the sake of legibility, y-axis ranges in Figs. 9, 10 and 11 have been adjusted

Table 1 DQN Cash/Inventory Average Ratio and Benchmarks

Experiment	DQN MM		Random MM		Persistent MM	
	Avg[a]	σ[b]	Avg[a]	σ[b]	Avg[a]	σ[b]
$AIIF = 0$	1, 08	0, 12	1, 31	0, 31	1, 44	1, 18
$AIIF = 0, 2$	1, 18	0, 06	1	0, 3	1, 24	0, 96
$AIIF = 0, 5$	1, 01	0, 15	1, 34	0, 23	1, 59	0, 95
$AIIF = 0, 8$	1, 32	0, 38	1, 33	0, 29	2, 22	2, 36
$AIIF = 1$	1, 48	0, 65	1, 22	0, 31	1, 29	0, 71
$AIIF = 1, 5$	1, 39	0, 21	1, 05	0, 34	1, 31	1, 32
$AIIF = 2$	3, 06	16, 55	1, 2	0, 28	1, 3	0, 69
$AIIF = 5$	11, 62	90, 3	1, 23	0, 29	1, 28	0, 43
$AIIF = 10$	7, 96	24, 42	1, 27	0, 29	1, 18	0, 45
$AIIF = 100$	59, 34	413, 12	1, 21	0, 28	1, 22	0, 44

Note: Table with the average ratio between the cash of the MM and the value of the inventory held at every experiment. It is noticeable that as long as the AIIF factor is increased, so does the cash over the inventory value. Every experiment corresponds to different Alpha factors (AIIF)
[a]Average Cash/Inventory ratio per experiment
[b]Cash/Inventory ratio standard deviation per experiment

y-axis ranges of Figs. 9, 10 and 11 have been adjusted independently for the sake of legibility.

Finally, we have explained that the dynamic threshold is calculated with the value of the inventory at every time. In this sense, we can talk about Cash / Inventory value ratios, as it is the main driver of the penalty term. Concerning this point, we can take a look at Table 1, where average ratios are presented per experiment. Again, we can observe how increasing AIIF factor impacts directly this ratio, by improving this value. This is very reasonable, as the agent is constantly managing the trade-off between profitability and inventory value. Thereby, the higher the AIIF factor, the higher Cash / Inventory value ratio returned.

6.3 Qualitative analysis of the generated policies

As formerly detailed, DQN agent behavior relies on a neural network that returns the best action for each state. On many occasions, neural networks are considered "black boxes", as they suffer from a certain lack of interpretability. In order to have a better understanding of the experimental results, and the policies followed by the different MMs, we analyze the different buy, sell, and hedging spreads performed in the different experiments.

Regarding buy and sell strategies, as shown in Fig. 12, it is noticeable how both $\eta_{(b,s)}$ move from negative to positive values as long as the AIIF rises. This suggests that with smaller or negative $\eta_{(b,s)}$ the MM is more aggressive in terms of prices (narrower spread), being able to capture a higher number of trades. Furthermore, it is also remarkable how strategies evolve not only in terms of negative or positive values, but increasing the distance between buy and sell $\eta_{(b,s)}$

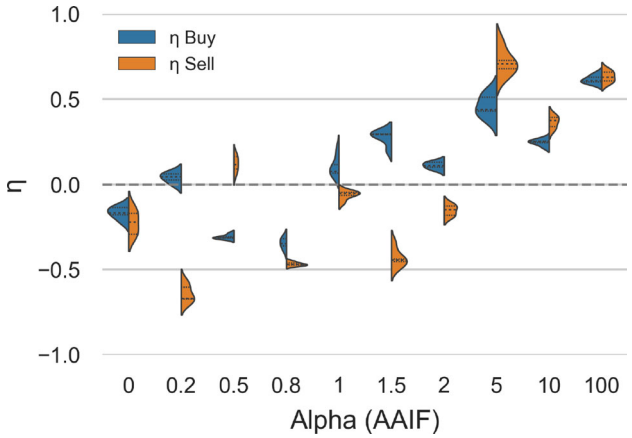

Fig. 12 DQN MM buy and sell strategy distributions, in terms of prices, according to different AIIF. The higher ηs, the more expensive price streamed by the MM. η_h=0 represents the same price as the last market spread. Different strategies are performed depending on the AIIF factors. With high AIIF factors, higher spreads are streamed by the MM

(as it happens around $AIIF = 0.5$ and $AIIF = 1.5$). Thus, different approaches and behaviors are found by the MM.

Regarding the hedging strategies (Fig. 13), we notice that as long as the inventory penalty increases so does hedging η_h. This increase makes MM keep low inventory despite losing profitability due to hedging costs. Thereby, buy/sell $\eta_{(b,s)}$ adaptive strategies combined with different hedging actions η_h composes the unique learned policies.

If we look at the average status of the MMs (Figs. 14 and 15), it is remarkable how higher AIIFs can even finish with less value in terms of MtM than starting value. Focusing on $AIIF = 5$, although the inventory position ends in a very constrained situation, we can see how the experiment finishes below the starting Mark-to-Market reference line.

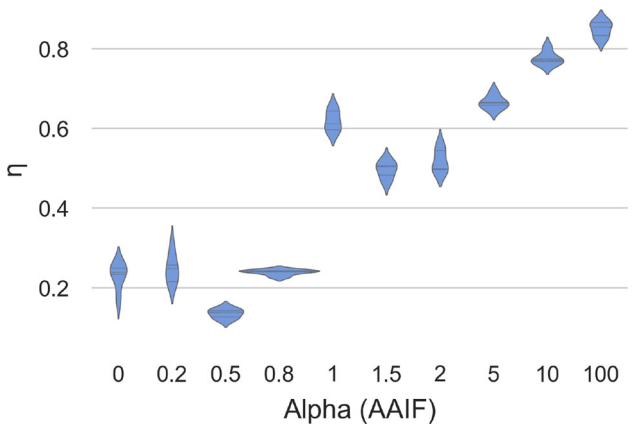

Fig. 13 DQN MM hedge strategy distributions according to different AIIF. η_h represents the amount of inventory the MM is reducing instantly by buying or selling at market price in a specific time step. This "urgency" has an extra cost. It is remarkable how higher hedges η_h are performed as long as the alfa factor is increased

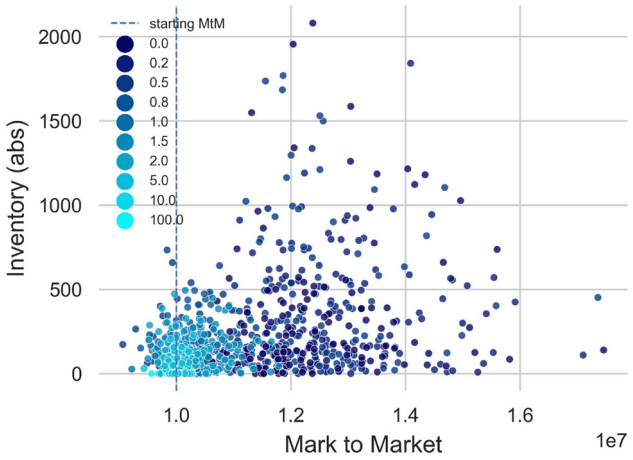

Fig. 14 DQN single inventory and MtM per AIIF results are obtained from every test experiment. Each point represents the average MtM and Inventory of the MM of every testing simulation (150 simulations x 10 *alphas*). As noticed, the dispersion is higher with lower AIIF factors

So, according to this, it is very important to be aware of the impact of increasing AIIF factor on the overall performance. All the details regarding the MtM can be found in Table 2.

Finally, aiming to fully understand the policies behind actions, we find it relevant to analyze how AIIF factor impacts the number of stocks traded by the DQN MM. Taking into account that higher AIIFs usually rely on higher buy and sell $\eta_{(b,s)}$ as stated before, we may expect that the number of operations performed by the MM could decrease as long as AIIF factor increases. In Fig. 16 we can appreciate that this intuition is accurate and especially relevant with higher AIIFs. In summary, and with all these insights raised we can say that, as long as the AIIFs increases, the MM becomes more selective with actions performed, preferring higher spreads and increasing the use of hedging.

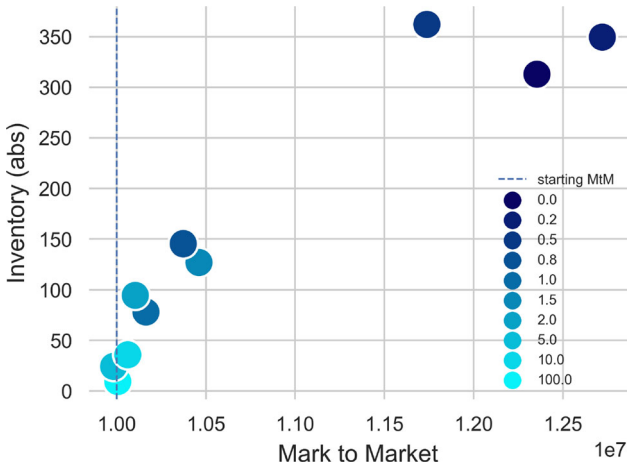

Fig. 15 DQN inventory and MtM in average per AIIF are obtained from every test experiment (grouped by *alpha*). Here we can notice the inverse correlation between MtM+Inventory and AIIF factors. In addition, it is clearly shown the competing nature of both goals: to reduce the inventory while increasing the MtM

6.4 Comparison with existing reward functions

To test the robustness of our approach it is important to perform a benchmark with other reward functions. Accordingly, we have selected different and recent works [22, 25, 27], already introduced in Section 2. These three different works represent common MM RL approaches, as they focus on different metrics to define the reward of their agents. In order to do a "fair" comparison among methods, we have used only their respective reward functions, while preserving the rest of the parameters in the most similar possible. Therefore, every agent shares the same state and action space, the same DQN architecture, and the same hedging policy, as presented before in Section 5. The rest of the experimental settings (environment agents, episode lengths, random competing agents, etc) have been preserved as well.

Hence, a total of 4 different reward functions have been compared:

- **Full inventory penalty**: Inspired in [27][6], this reward function includes a penalty term related to the absolute amount of inventory held by the MM at every time step, multiplied by a risk aversion coefficient, as shown in (12):

$$R_i = O_i - \lambda * |inv_i| - Hc_i \tag{12}$$

  λ has been set to $\lambda = 0.15$, following the experimental setup of the mentioned work.
- **Asymmetrically dampened PnL**: Based on [25][7], this reward function includes a penalty term that penalizes the incomes coming from Inventory value increases. It is modulated by a scale factor η, as shown in (13):

$$R_i = O_i + Pnl_i - max(0, \eta * Pnl_i) - Hc_i \tag{13}$$

  η has been initially set to $\eta = 0.1$ as shown in the respective work.
- **PnL**: No inventory penalty term is included. This is a common approach in many works (i.e. [22]) as introduced in Section 2. Here, only the profits derived from trades are considered, apart from hedging costs, as shown in (14):

$$R_i = O_i - Hc_i \tag{14}$$

[6] $R_{t+1} = (Q_t^{ask} - M_{t+1})1Q_t^{ask}exe + (M_{t+1} - Q_t^{bid})1\{Q_t^{bid}exe\} - \lambda|I_{t+1}|$

[7] $r_i = \psi(t_i) - \max(0, \eta * Inv(t_i)\triangle m(t_i))$

Table 2 DQN Mark-to-Market experiment results (x10^3) and baseline agents

Experiment	DQN MM			Random MM			Persistent MM		
	Avg[a]	σ[b]	Var[c]	Avg[a]	σ[b]	Var[c]	Avg[a]	σ[b]	Var[c]
$AIIF = 0$	12.399	48	24,50%	9.711	10	-2,98%	9.648	13	-3,35%
$AIIF = 0.2$	12.833	66	28,05%	9.559	7	-4,30%	9.575	14	-4,45%
$AIIF = 0.5$	11.688	122	15,27%	9.957	39	-0,33%	9.889	17	-0,71%
$AIIF = 0.8$	10.358	23	3,69%	9.823	9	-1,96%	9.884	14	-1,26%
$AIIF = 1$	10176	7	1,72%	9.889	9	-0,94%	9.900	9	-0,88%
$AIIF = 1.5$	10.501	24	5,60%	9.775	12	-2,71%	9.811	13	-1,76%
$AIIF = 2$	10.081	12	0,85%	9.944	13	-0,47%	9.922	13	-0,81%
$AIIF = 5$	9.984	3	-0,18%	10.064	16	0,69%	10.038	13	0,37%
$AIIF = 10$	10.061	6	0,57%	9.969	7	-0,36%	9.998	9	0,15%
$AIIF = 100$	10.008	1	0,11%	10.019	15	0,04%	10.048	11	0,41%

Note: Table with all the MtM values along different testing experiments. Every experiment corresponds to different Alpha factors (AIIF). Starting MtM = 10.000 (x10^3)
[a]Average MtM along the experiment
[b]MtM standard deviation along the experiment
[c]MtM increase/decrease at end of the experiment (return)

- **RIM** (Our reward function, (2)): For the sake of comparison, different AIIFs have been included: $AIIF \in \{0.2, 0.5, 0.8, 1\}$.

Seven different DQN agents were trained independently, based on the described reward functions. Examining the results in Fig. 17 and Table 3, we see that our reward function with AIIF = 0.2 was the most profitable during the 150 tested trading sessions, followed by the *Asymmetrically dampened* reward function and our reward function with AIIF = 0.5. These two last cases obtain almost the same profitability. With higher AIIF values ($\alpha = 0.8$ and $\alpha = 1$) MtM decreases, as stated earlier, due to inventory control. All these mentioned agents had stable returns during the test experiments, as shown in Fig. 17. However, the agent with the *PnL* reward function performed much worse in both MtM and inventory management, indicating that inventory metrics should be included in the reward function to design better market-making agents.

Regarding inventory management, and taking a look at Fig 18, we notice how our reward function performed well in all cases. *Full inventory penalty* reward function also performed well, despite lower returns. However, *Asymmetrically dampened PnL* and *PnL* reward functions perform poorly in managing inventory, as shown in the distribution dispersion.

As an overall conclusion of this benchmark, we can conclude that our reward function allows the market maker to perform better in terms of MtM, and also, in terms of inventory management than the rest of the tested functions. The average ratio figures between the cash and the inventory value presented in Table 3 also illustrate how our market makers can manage properly these proportions. Furthermore, from the obtained results we can affirm that including some kind of penalty term in the reward function helps to improve the profitability of the agent while constraining the inventory risks. In fact, those reward functions that are based only on PnL, ignoring inventory, can lead to the design of ineffective and also unstable agents.

6.5 Discussion

Achieving different goals in a multi-objective environment is not always easy. Market-making can be considered a multi-objective task. The presented results (Table 3) demonstrate that our proposed agent and reward function can deal with

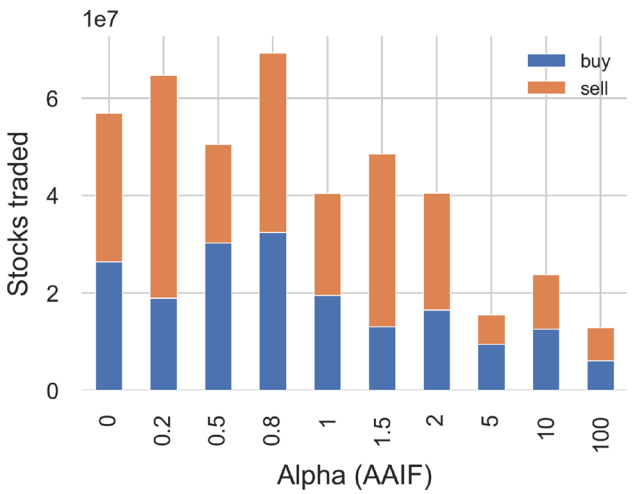

Fig. 16 Buy and sell amount of stock traded according to different AIIF. As long as the AIIF factor is increased, the number of executed operations is reduced. This has to do with the increase of buy and sell $\eta_{(b,s)}$ shown in Fig. 12, a point that makes the MM to be less competitive in terms of price, and therefore, captures less orders from the investors

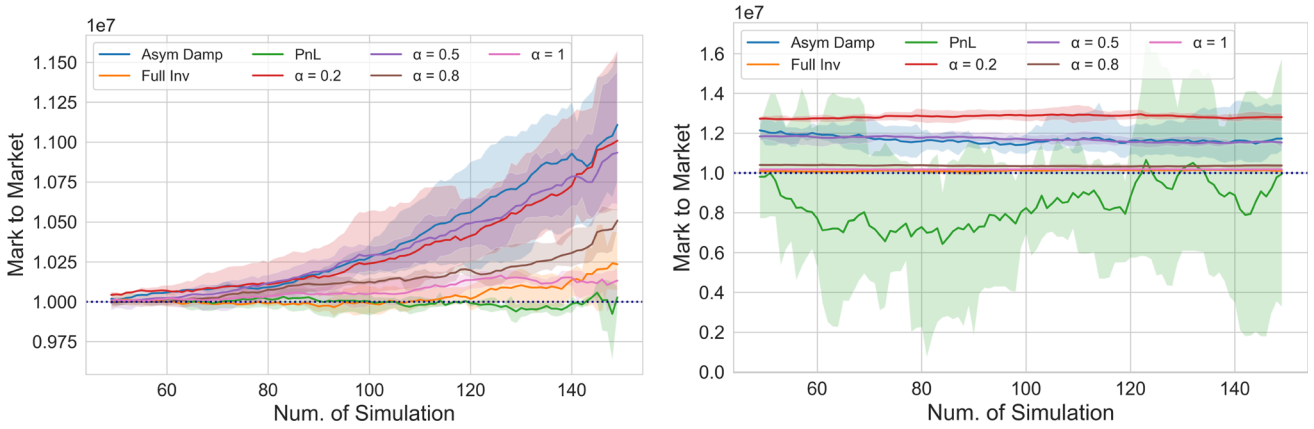

Fig. 17 Benchmark results in terms of MtM (cash + inventory value). Training stage (left) and testing stage (right). Four different reward functions are compared: Full Inv 12, Asym Damp 13, PnL 14, and our reward function RIM 2 (*alpha*). Different AIIF factors (α) have been included for our reward function. It is noticeable that, as long as the *alpha* factors are increased, the MtM is reduced as well due to the more strict control on the inventory. In addition, PnL reward function, which focuses on improving only the returns, shows a very unstable behavior with high variance

Table 3 Benchmark results. MtM, inventory, and cash/inventory ratio are presented

	Mark-To-Market			Inventory		Cash/Inv. Value ratio	
Experiment	Avg[a]	σ^b	Var^c	Avg[a]	σ^b	Avg[a]	σ^b
$AsymDamp$	11.672	162	17, 27%	2.975	288	1, 08	0, 25
$FullInv$	10.107	27	1, 02%	−32	14	1, 15	0, 19
PnL	8.338	1.048	−0, 53%	16.569	1.882	1, 03	0, 14
$AIIF = 0.2$	12.833	66	28, 05%	−183	35	1, 18	0, 06
$AIIF = 0.5$	11.688	122	15, 27%	199	31	1, 01	0, 15
$AIIF = 0.8$	10.358	23	3, 69%	−40	9	1, 32	0, 38
$AIIF = 1$	10.176	7	1, 72%	−8	4	1, 47	0, 64

Note: Table with the results of the different reward functions tested, in terms of Mark-To-Market, inventory, and cash/inventory value ratio
[a]Average value along the experiment
[b]Standard deviation along the experiment
[c]MtM increase/decrease at end of the experiment (return)

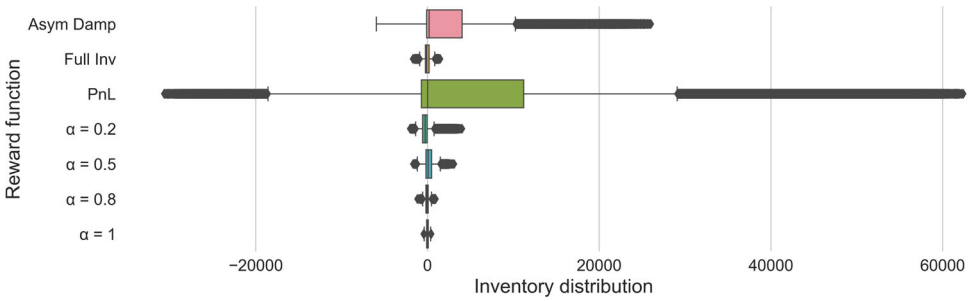

Fig. 18 Benchmark results in terms of Inventory distribution. This Figure illustrates the different inventories held by every agent during the testing stage, according to their different reward functions. Four different reward functions are compared: Full Inv 12, Asym Damp 13, PnL 14 and our reward function RIM 2 (*alphas*). Regarding our reward function, 4 different AIIF factors (α) have been included: $\alpha \in \{0.2, 0.5, 0.8, 1\}$. It is clear how PnL reward function tends to accumulate more inventory (positive or negative) compared to the rest of the alternatives. In addition, our proposed reward function behaves very well with all the different *alpha* factors

this, adapting properly to the changing environment while seeking the two main objectives:

- To be profitable, as shown in the results. In this sense, the MtM average results show that our agent obtains good returns and performs better than most of the previous ones, except for AsymDamp.
- To keep inventory controlled, as evidenced by comparing the average inventory returns with those of PnL and AsymDamp alternatives.

As a limitation of this method, we can remark that we must have one pre-trained agent per configuration needed before going live. Defining a scalarized reward function, as done in this work, requires prior knowledge of the future utility function. If the utility function changes "a posteriori" we need to have pre-trained agents that can adapt to it.

Conversely, RL enables us to define agents simply by specifying the reward function. This means it always seeks the best policy, regardless of system complexity. This is a noteworthy advantage in stochastic and dynamic environments such as stock markets, where rule-based agents would require continuous calibrations and adaptations.

7 Conclusion and future work

In this paper, we have presented a robust approach for learning a market-making agent based on RL, where the agent can modulate automatically its inventory according to a desired liquidity ratio. We have introduced a reward function that, when combined with the DQN algorithm, can modulate the inventory while seeking profitability and adapting to the evolving liquidity of the agent. This reward function has two key factors that allow the agent to control the inventory risk dynamically while looking for profitability. First, the AIIF factor modulates the risk aversion related to inventory risk by serving as a penalty term. Additionally, DITF factor can dynamically adjust the behavior of the agent as it adapts to the cash/inventory value held at every time. These two terms combined enable the MM to control its risks dynamically while adhering to the initial risk aversion strategy. Therefore, the MM will not follow a static policy driven by a standard scalarised multi-objective reward function. Rather, it will adapt to the market based on its holdings of cash and inventory.

The MM agent has been tested with different penalty factors to evaluate the performance of these parameters in a simulated but realistic environment, and has also been compared to random baseline agents. Experimental results show that the presented approach allows for the design of an RL MM that, with only two liquidity parameters, efficiently modulates its operations.

Additionally, we have not only designed this intelligent agent, but we have also been able to understand and evaluate the underlying strategies followed by our MM in terms of price and hedge spreads (actions). We have analyzed this taking into account different penalty coefficients, getting valuable insights about the trading operative in terms of pricing, inventory hedging, and inventory dynamic thresholds. This analysis allows us to delve into the reasoning behind the learned strategies, without leaving the agent to operate as a "black box".

Last but not least, we have compared our proposed solution with other existing reward functions, concluding that our function can achieve good returns while managing inventory risk in a more robust way than the alternative tested approaches.

Concerning future work, there are some aspects that can also be explored and even improved. The main thing we find relevant is related to the selection of a proper AIIF factor. With our current proposal, this coefficient is predetermined and does not consider the changing market conditions over time. It is well known that market volatility evolves over time, even for the same asset, behaving differently depending on the hour of the day, the arrival of new economic data, or some other events. Including a volatility parameter that modulates this AIIF coefficient according to the market "texture" at every time could be a good upgrade of the reward function.

Author Contributions **Óscar Fernández**: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - original draft, Writing - review & editing. **Fernando Fernández and Javier García**: Conceptualization, Methodology, Formal analysis, Writing - review editing & Validation.

Funding This work has been partially funded for APC: Universidad Carlos III de Madrid (Read & Publish Agreement CRUE-CSIC 2023). This research was also funded in part by JPMorgan Chase & Co. Any views or opinions expressed herein are solely those of the authors listed, and may differ from the views and opinions expressed by JPMorgan Chase & Co. or its affiliates. This material is not a product of the Research Department of J.P. Morgan Securities LLC. This material should not be construed as an individual recommendation for any particular client and is not intended as a recommendation of particular securities, financial instruments or strategies for a particular client. This material does not constitute a solicitation or offer in any jurisdiction

Data availability The datasets generated during the current study are available in the GIT repository, https://github.com/ofvicente/automated_mm_paper.

Declarations

Ethical approval This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Hendershott T (2003) Electronic trading in financial markets. IT Prof Mag 5(4):10
2. Huberman G (2005) A simple approach to arbitrage pricing theory. In: Theory of Valuation. World Scientific, pp 289–308. https://doi.org/10.1142/9789812701022_0009
3. Chordia T, Goyal A, Lehmann BN et al (2013) High-frequency trading. SSRN Electron J. https://doi.org/10.2139/ssrn.2278347
4. Copeland TE, Galai D (1983) Information effects on the bid-ask spread. J Financ 38(5):1457–1469. https://doi.org/10.1111/j.1540-6261.1983.tb03834.x
5. Veasna K, Feng Z, Zhang Q et al (2023) Machine learning-based multi-objective optimization for efficient identification of crystal plasticity model parameters. Comput Methods Appl Mech Eng 403:115740. https://doi.org/10.1016/j.cma.2022.115740
6. García J, Majadas R, Fernández F (2020) Learning adversarial attack policies through multi-objective reinforcement learning. Eng Appl Artif Intell 96(104):021. https://doi.org/10.1016/j.engappai.2020.104021
7. Hua Y, Liu Q, Hao K et al (2021) A survey of evolutionary algorithms for multi-objective optimization problems with irregular pareto fronts. IEEE/CAA J Autom Sin 8(2):303–318. https://doi.org/10.1109/JAS.2021.1003817
8. Hayes CF, Rădulescu R, Bargiacchi E et al (2022) A practical guide to multi-objective reinforcement learning and planning. Auton Agent Multi-Agent Syst 36(1):26
9. Gašperov B, Begušić S, Posedel Šimović P, et al (2021) Reinforcement learning approaches to optimal market making. Mathematics 9(21). https://doi.org/10.3390/math9212689
10. Avellaneda M, Stoikov S (2008) High-frequency trading in a limit order book. Quant Finance 8:217–224. https://doi.org/10.1080/14697680701381228
11. Ho T, Stoll HR (1981) Optimal dealer pricing under transactions and return uncertainty. J Financ Econ 9(1):47–73. https://doi.org/10.1016/0304-405X(81)90020-9
12. Guéant O, Lehalle CA, Tapia JF (2011) Dealing with the inventory risk. Mathematics and Financial Economics, A Solution to the Market Making Problem. https://doi.org/10.1007/s11579-012-0087-0
13. Guéant O (2017) Optimal market making. Appl Math Finance 24(2):112–154. https://doi.org/10.1080/1350486X.2017.1342552. https://hal.archives-ouvertes.fr/hal-02862554
14. Ait-Sahalia Y, Saalam M (2017) High frequency market making: implications for liquidity. SSRN Electron J. https://doi.org/10.2139/ssrn.2908438
15. Masini RP, Medeiros MC, Mendes EF (2023) Machine learning advances for time series forecasting. J Econ Surv 37(1):76–111. https://doi.org/10.1111/joes.12429
16. Li Q, Peng H, Li J, et al (2022a) A survey on text classification: From traditional to deep learning. ACM Trans Intell Syst Technol 13(2). https://doi.org/10.1145/3495162
17. Liu H, Zheng C, Li D et al (2022) Multi-perspective social recommendation method with graph representation learning. Neurocomputing 468:469–481. https://doi.org/10.1016/j.neucom.2021.10.050
18. Liu H, Zheng C, Li D et al (2022) Edmf: Efficient deep matrix factorization with review feature learning for industrial recommender system. IEEE Trans Ind Inf 18(7):4361–4371. https://doi.org/10.1109/TII.2021.3128240
19. Li Z, Liu H, Zhang Z et al (2022) Learning knowledge graph embedding with heterogeneous relation attention networks. IEEE Trans Neural Netw Learn Syst 33(8):3961–3973. https://doi.org/10.1109/TNNLS.2021.3055147
20. Li X, Deng X, Zhu S et al (2014) An intelligent market making strategy in algorithmic trading. Front Comput Sci 8(4):596–608. https://doi.org/10.1007/s11704-014-3312-6
21. Dixon MF (2017) High frequency market making with machine learning. SSRN Electron J. https://doi.org/10.2139/ssrn.2868473
22. Vicente OF, Rebollo FF, Polo FJG (2022) Deep q-learning market makers in a multi-agent simulated stock market. In: Proceedings of the Second ACM International Conference on AI in Finance. Association for Computing Machinery, New York, NY, USA, ICAIF '21, 10.1145/3490354.3494448
23. Lokhacheva K, Parfenov D, Bolodurina I (2020/01) Reinforcement learning approach for market-maker problem solution. In: Proceedings of the International Session on Factors of Regional Extensive Development (FRED 2019). Atlantis Press, pp 256–260. https://doi.org/10.2991/fred-19.2020.52
24. Zhong Y, Bergstrom Y, Ward A (2020) Data-driven market-making via model-free learning. In: Bessiere C (ed) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20. International Joint Conferences on Artificial Intelligence Organization, pp 4461–4468. https://doi.org/10.24963/ijcai.2020/615, special Track on AI in FinTech
25. Spooner T, Fearnley J, Savani R, et al (2018) Market making via reinforcement learning. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '18, pp 434–442
26. Selser M, Kreiner J, Maurette M (2021) Optimal market making by reinforcement learning. SSRN Electron J. https://doi.org/10.2139/ssrn.3829984
27. Gasperov B, Kostanjcar Z (2021) Market making with signals through deep reinforcement learning. IEEE Access 9:61611–61622. https://doi.org/10.1109/ACCESS.2021.3074782
28. Lim YS, Gorse D (2018) Reinforcement learning for high-frequency market making. 26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning ESANN 2018 : Bruges, Belgium, April 25, 26, 27, 2018 : proceedings. https://www.esann.org/sites/default/files/proceedings/legacy/es2018-50.pdf
29. Zhang G, Chen Y (2020) Reinforcement learning for optimal market making with the presence of rebate. SSRN Electron J. https://doi.org/10.2139/ssrn.3646753
30. Mani MA, Phelps S (2019) Applications of reinforcement learning in automated market-making. In: GAIW, May 2019, Montreal, Canada. https://nms.kcl.ac.uk/simon.parsons/publications/conferences/gaiw19.pdf

31. Sutton RS, Barto AG (2011) Reinforcement learning: an introduction. MIT Press, Cambridge, MA
32. Ramírez J, Yu W, Perrusquía A (2022) Model-free reinforcement learning from expert demonstrations: a survey. Artif Intell Rev 55(4):3213–3241. https://doi.org/10.1007/s10462-021-10085-1
33. Otto F (2021) Model-free deep reinforcement learning—algorithms and applications. Springer International Publishing, Cham, pp 109–121. https://doi.org/10.1007/978-3-030-41188-6_10
34. Mnih V, Kavukcuoglu K, Silver D et al (2015) Human-level control through deep reinforcement learning. Nature 518:529–533. https://doi.org/10.1038/nature14236
35. Clifton J, Laber E (2020) Q-learning: theory and applications. Annu Rev Stat Appl 7(1):279–301. https://doi.org/10.1146/annurev-statistics-031219-041220
36. Mallozzi P, Pardo R, Duplessis V, et al (2018) Movemo: a structured approach for engineering reward functions. In: 2018 Second IEEE International Conference on Robotic Computing (IRC), pp 250–257. https://doi.org/10.1109/IRC.2018.00053
37. Hussein A, Elyan E, Gaber MM, et al (2017) Deep reward shaping from demonstrations. In: 2017 International Joint Conference on Neural Networks (IJCNN), pp 510–517. https://doi.org/10.1109/IJCNN.2017.7965896
38. Adams S, Cody T, Beling PA (2022) A survey of inverse reinforcement learning. Artif Intell Rev. https://doi.org/10.1007/s10462-021-10108-x
39. Behboudian P, Satsangi Y, Taylor ME et al (2022) Policy invariant explicit shaping: an efficient alternative to reward shaping. Neural Comput Appl 34(3):1673–1686. https://doi.org/10.1007/s00521-021-06259-1
40. Ganesh S, Vadori N, Xu M, et al (2019) Reinforcement learning for market making in a multi-agent dealer market. ArXiv abs/1911.05892
41. Byrd D, Hybinette M, Balch TH (2020) Abides: towards high-fidelity multi-agent market simulation. In: Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. Association for Computing Machinery, New York, NY, USA, SIGSIM-PADS '20, pp 11–22. https://doi.org/10.1145/3384441.3395986
42. Amrouni S, Moulin A, Vann J, et al (2022) Abides-gym: gym environments for multi-agent discrete event simulation and application to financial markets. In: Proceedings of the Second ACM International Conference on AI in Finance. Association for Computing Machinery, New York, NY, USA, ICAIF '21. https://doi.org/10.1145/3490354.3494433
43. Assefa SA, Dervovic D, Mahfouz M, et al (2021) Generating synthetic data in finance: opportunities, challenges and pitfalls. In: Generating Synthetic Data in Finance: Opportunities, Challenges and Pitfalls. Association for Computing Machinery, New York, NY, USA, ICAIF '20. https://doi.org/10.1145/3383455.3422554
44. Kuo CH, Chen CT, Lin SJ et al (2021) Improving generalization in reinforcement learning-based trading by using a generative adversarial market model. IEEE Access 9:50738–50754. https://doi.org/10.1109/ACCESS.2021.3068269
45. Zhang Z, Lim B, Zohren S (2021) Deep learning for market by order data. Appl Math Finance 28(1):79–95. https://doi.org/10.1080/1350486X.2021.1967767

Óscar Fernández Vicente possesses a wealth of expertise in computer science, finance, and industrial applications of artificial intelligence (AI). With a degree in Computer Science from Antonio de Nebrija University (2005) and more than thirteen years of experience in the banking sector as a data specialist and data scientist (2007-2020), he has established a strong foundation in algorithms and programming languages. Currently serving as a machine learning applied scientist in the industrial sector since 2022, he is actively involved in the development of AI systems utilizing reinforcement learning and deep learning techniques. Concurrently pursuing a PhD in Artificial Intelligence applied (RL) to financial markets (UC3M), his dedication to research and innovation in AI underscores his commitment to advancing knowledge.

Fernando Fernández is an associate professor of the Computer Science Department at Universidad Carlos III de Madrid, since october 2005. He received his Ph.D. degree in Computer Science from University Carlos III of Madrid (UC3M) in 2003. He received his B.Sc. in 1999 from UC3M, also in Computer Science. In the fall of 2000, Fernando was a visiting student at the Center for Engineering Science Advanced Research at Oak Ridge National Laboratory (Tennessee). He was also a postdoctoral fellow at the Computer Science Department of Carnegie Mellon University since october 2004 until december 2005. He is the recipient of a pre-doctoral FPU fellowship award from Spanish Ministry of Education (MEC), a Doctoral Prize from UC3M, and a MEC-Fulbright postdoctoral Fellowship as well as the 2020 and 2021 JPMorgan AI Research Awards. He has more than 80 journal and conference papers, mainly in the field of machine learning, automated planning and robotics. He is interested in intelligent systems that operate in continuous and stochastic domains. He is the Director of the Planning and Learning Group of the Computer Science Department at UC3M. He is also co-founder and CSO of Inrobocs Social Robotics, where social assistive robots are deployed in rehabilitation hospitals.

Javier García received his B.S. and Ph.D. degree in Computer Science from the Universidad Carlos III de Madrid (UC3M), Spain, in 2006 and 2013, respectively. Since 2021, Dr. García is Associate Professor in the Department of Electronics and Computer Science at the University of Santiago de Compostela, Spain. His research interests include robotics, machine learning, reinforcement learning and multi-agent systems.