Multi-Objective Reinforcement Learning for Portfolio Allocation

Jakub Lewkowicz 2781247, Supervisor: Floris den Hengst

Vrije Universiteit Amsterdam

Abstract. In recent years, the application of Artificial Intelligence in the financial domain has become a crucial area of research and innovation, benefiting financial corporations and individual traders alike. Research into portfolio management has long recognized the sequential nature of the problem, making it amenable to various approaches, including reinforcement learning. Recent work in this field focuses on optimizing the Sharpe ratio, which balances risk and return. However, this approach overlooks the complex relationships between these objectives, offering investors limited choices regarding their investment preferences. This thesis redefines portfolio optimization as a Multi-Objective Reinforcement Learning (MORL) problem and applies Pareto Conditioned Networks (PCN) to rebalance a portfolio of S&P 500 stocks and cryptocurrencies. Additionally, it includes a robust comparison of PCN with established RL algorithms such as PPO and DQN. Results indicate that PCN delivers very similar results to PPO and consequently outperforms baseline strategy in S&P 500 markets. This work provides guidance on applying MORL to portfolio optimization and discusses key differences between multi-objective and single-objective approaches.

1 Introduction

In recent years, the application of Artificial Intelligence (AI) into the financial domain has emerged as a crucial area of research and innovation, for financial corporations as well as for individual traders. This intersection holds particular importance for addressing complex problems such as portfolio allocation, where optimal asset distribution is the principal to achieve satisfying returns. Portfolio allocation involves the strategic distribution of an investor's capital among a finite set of various assets, to maximize return while managing associated risks. This problem can be defined as a sequential decision-making task.

Research into portfolio management has long recognized the aforementioned sequential nature of the problem, making it amenable to various approaches, including reinforcement learning (RL). The financial market can be simulated as an RL environment and the asset manager can be expressed as an agent acting in such an environment choosing the asset's distribution based on the current market situation. RL offers a robust framework for modeling portfolio allocation as a dynamic decision-making problem. However, existing reinforcement learning approaches often struggle to effectively balance competing objectives, such as

profit maximization and risk management. Most of the studies in the application of reinforcement learning to portfolio optimization focus on the optimization of a single utility function known as the Sharpe ratio. The aforementioned function represents the balance between risk and return, nevertheless, it is a simplification of a problem because the relation between risk and return is not always linear, it is time-varying and also dependent on the level of the risk considered[1].

Moreover, the traditional approach of optimizing a single utility function fails to capture the diverse objectives and risk preferences of individual investors. While fundamental objectives like return and risk remain central, there is a growing recognition of the importance of other factors, such as maximal drawdown, portfolio diversification, liquidity maintenance, and consideration of Environmental, Social, and Governance (ESG) factors. These multifaceted objectives underscore the need for a more nuanced and flexible optimization framework.

While prior research revolving around the application of RL to portfolio allocation, focused on the direct application of well-known single-objective RL algorithms, it is beneficial to explore modeling portfolio allocation as a multi-objective problem. Existing approaches often fall short in this regard, neglecting complex relations between various objectives and risk factors.

This study addresses the challenge of applying Multi-Objective Reinforcement Learning (MORL) to portfolio optimization. Multi-Objective Reinforcement Learning extends the traditional RL framework by introducing simultaneous optimization of multiple objectives, rather than a single utility function. This approach is well suited to portfolio management, where investors seek to balance various goals.

The key research question in this study aims to answer is how to apply Multi-Objective Reinforcement Learning to portfolio optimization problem. This study describes how to define portfolio optimization as a Multi-Objective Markov Decision Process (MOMDP). Moreover, it explains how to define a simulated trading environment to apply MORL algorithms. The study discusses what is the suitable optimization criterion for applying MORL to portfolio optimization. Additionally, it poses the question if the MORL algorithm can find a better policy that can balance risk and return, than classical RL methods. The question is answered by conducting a robust comparison of Pareto Condition Networks (PCN) with DQN and PPO in both equities and cryptocurrency markets.

2 Related Work

This chapter explains how Reinforcement Learning has been applied to trading and portfolio optimization problems so far. It presents RL algorithms that were applied to the problem and explains what results were achieved. Consequently, it introduces utility functions for portfolio optimization problems in the single-objective setups.

Most of the research in the domain of Reinforcement Learning for Portfolio Allocation focuses on single-objective methods. Zhang et. al (2019) adopt Deep Reinforcement Learning (DRL) algorithms to design trading strategies for

futures contracts [2]. The authors apply 3 different DRL methods: Deep Q Network (DQN), Policy Gradient (PG), and Actor-Critic (A2C). They compare those methods with classical momentum strategies and conclude that the DQN agent outperforms both classical strategies and other agents. Although the authors do not directly tackle the portfolio allocation problem, they show positive results in applying DRL to sequential decision-making problems in trading and discuss the potential of applying those methods to portfolio allocation. Those conclusions indicate the need for further investigation of the application of RL to portfolio allocation.

In more recent work Sood et. al (2023) compare deep RL methods with traditional financial techniques such as Mean-Variance Optimisation (MVO)[3]. MVO is the mathematical process of allocating capital across a portfolio of assets that is widely used by finance professionals. MVO optimizes risk-adjusted returns either for single or multi-objectives. The authors choose the Sharpe ratio as their desired objective for MVO. For the RL methods they apply, they decided to use the Differential Sharpe Ratio as a reward function, because the Sharpe ratio itself is inappropriate for online learning. The superiority of Differential Sharpe Ratio for online learning was first recognized by Moody and Saffel (1998) [4]. Sood et. al (2023) apply a Proximal Policy Optimization (PPO) that optimizes risk-adjusted returns. They train those models in a multi-asset trading environment that simulates the US Equities market. They emphasize that much research that revolves around using DRL for portfolio optimization compares applied models only to other RL methods or long-hold strategies. The PPO approach outperforms the MVO baseline in an experiment constructed on S&P500 data collected from 2006 to 2021. From that research it can be derived that there is a potential for DRL methods for the portfolio allocation problem.

It can be observed that there is a rising trend in research revolving around RL applied to portfolio optimization and trading in general. However, there is a lack of attention to the multi-objectivity of the portfolio optimization problem itself. In recent work, it can be observed researchers purely focus on the application of known RL models to the problem, where they use reward functions directly derived from the Sharpe ratio. It assumes a trade-off made between risk and reward, without a separate focus on the optimised objectives. In real life, investors might have different risk preferences and are willing to accept disproportionate higher risk for higher return, which leads to a worse, non-optimal Sharpe ratio. Contrary to that, some investors have low-risk tolerance and are content with smaller returns, given smaller risks. That observation identifies a gap in the current approaches and gives space to redefine the problem in a multi-objective way.

Hayes et. al (2022) discuss various approaches for multi-objective problems [5]. The authors divide approaches into ones optimizing the expected utility of the returns and the utility of the expected returns. If the utility is derived from single outcomes of performing the policy, the utility function has to be applied to the returns, and then the expected utility of the returns (ESR) has to be optimized. The authors give as an example a medical treatment planning setting,

where the patients derive their utility from their specific treatment outcomes. In such an example user will apply the learned policy once or several times and each outcome of the policy is critical. If the trading strategy operates as a low-frequency strategy, it means that holding periods last from days to weeks, and each outcome of the strategy execution is also critical and the aforementioned approach can be applied. Low-frequency trading is mostly practiced by retail investors and on this particular group, this research tends to focus. Multi-objective reinforcement learning is a novel approach by itself and has not been applied to trading problems.

Its application to portfolio optimization shows potential to advance the domain by accommodating the diverse preferences of investors. Optimizing multiple objectives simultaneously can provide more suited and effective portfolio allocation strategies. Addressing a gap in current research can offer practical benefits to different investors and can be a foundation for further study. Moreover presented RL methods such as DQN and PPO form already successful approaches that this study uses as baseline models that are compared with the multi-objective RL approach.

3 Methodology

In this section, we detail how multi-objective reinforcement learning can be applied to the problem of portfolio optimization. It starts with Multi-Objective Markov Decision Process formalization and the definition of its components. Furthermore, this section includes a description of trading environment implementation. It presents the description of Pareto Condition Networks and explains how it is used for portfolio optimization.

3.1 Multi-Objective Markov Decision Process

The portfolio optimization problem can be defined as a Multi-Objective Markov Decision Process (MOMDP) where an agent interacts with an environment in discrete time steps. At each time step t, the agent receives a representation of a state S_t, based on which the agent takes action A_t. Based on the taken action a reward vector R_{t+1} is returned to the agent at the next step and the agent finds itself in a state S_{t+1}. The length of the reward vector is equal to the number of objectives. The interaction between the agent and the environment produces a trajectory $\tau = [S_0, A_0, R_1, S_1, A_1, R_2 \dots]$.

3.2 Actions

For the portfolio optimization problem agent decides which assets to keep in the portfolio at each time step t. The decision is made by selecting a discrete action that specifies which assets to keep. The action space consists of 2^N discrete actions, where each action is represented by a binary vector. For N assets there are 2^N portfolio composition combinations. In this binary vector, each element

indicates the presence of the particular asset: the value of 1 means the asset is included in the portfolio, and the value of 0 means the asset is excluded. Once an action is chosen, the binary vector is transformed into an allocation vector that indicates how much of the current portfolio value should be distributed into a given asset.

- **Binary Vector Representation:**
  Let $a \in \{0,1\}^N$ be a binary vector where $a_i = 1$ if the i-th asset is included in the portfolio, and $a_i = 0$ if it is excluded.
- **Transformation to Allocation Vector:**
  The binary vector a is transformed into an allocation vector $w \in [0,1]^N$ as follows:
  $$w_i = \frac{a_i}{\sum_{j=1}^{N} a_j} \quad \text{for all} \quad i \in \{1, 2, \ldots, N\}$$
  The allocation vector w must satisfy:
  $$\sum_{i=1}^{N} w_i = 1 \quad \text{and} \quad 0 \leq w_i \leq 1 \quad \text{for all} \quad i \in \{1, 2, \ldots, N\}$$
- **Example:**
  If there are 4 assets and the binary vector is $[1, 0, 1, 0]$, the allocation vector is $[0.5, 0, 0.5, 0]$.

3.3 States

The observation space is defined as a matrix where each row contains flag a indicating if the asset is currently present in the portfolio and 30 last observed p asset returns at a step t. Asset price vectors are expressed as daily log returns observed until step t in the current episode. A similar observation space was also used by Sood et. al (2023). [3]

$$obs = \begin{bmatrix} a_1 & p_1^{t-29} & p_1^{t-28} & \ldots & p_1^t \\ a_2 & p_2^{t-29} & p_2^{t-28} & \ldots & p_2^t \\ a_3 & p_3^{t-29} & p_3^{t-28} & \ldots & p_3^t \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_N & p_N^{t-29} & p_N^{t-28} & \ldots & p_N^t \end{bmatrix}$$

3.4 Rewards

In the multi-objective setting, the objective is to maximize the return and minimize the risk of the portfolio, both objectives need separate reward functions, so the agent is either rewarded or penalized according to the taken action. Log returns are the usual way of expressing the portfolio's profit and loss over time. Log returns are additive, the sum of daily log returns over a given period directly reflects portfolio profit and loss over that time. Using daily log returns

as the reward function that corresponds to portfolio return allows for a direct understanding of how the current state of the portfolio contributes to the overall portfolio's return. Log return is expressed as $Return_t = log(\frac{V_t}{V_{t-1}})$, where V_t is a portfolio's value at a time step t. To reflect overall profit and loss by the end of the episode agent receives an additional signal in the form of the log return negative value or positive value concerning the percentage return. Returning the signal as a reward at the terminal step of the episode helps with the learning process in maximizing the value of the portfolio. The terminal signal serves as a clear indicator of success or failure, providing a learning signal that can enhance the agent's ability to improve its policy. This helps the agent understand the long-term effects of its actions that are not reflected in immediate rewards over the episode. In real-world portfolio management, the end-of-the-period performance is often more important than day-to-day fluctuations of the portfolio value. For example, an agent can allocate money into assets that are currently in the downfall but are showing promising potential to be profitable in the future. This kind of behavior would lead to immediate negative log returns, but in the long term would be an important factor that makes the portfolio profitable. Adding a signal at the end-of-episode makes the return reward function more aligned with real-world evaluation criteria.

Risk is expressed as negative portfolio volatility. Volatility is a square root of the variance and is a standard measure for modeling risk. It also indirectly measures portfolio diversification, by considering the covariance of stocks in the formula.

The choice of such a reward function is motivated by the persistent use of differential Sharpe ratio for DRL. Splitting components of the Sharpe ratio into two separate objectives to optimize, reflects the initial goal in DRL to optimize the ratio of those, however is not based on the assumption of linear relation between those components.

During training, the reward vector is scaled such that each of its elements is between -1 and 1.

The profit reward R_{profit} is defined as:

$$R_{\text{profit}} = \begin{cases} \log\left(\frac{V_t}{V_{t-1}}\right) & \text{if not terminated} \\ pctg_{return} * 1000 & \text{if terminated and final pnl is positive} \\ -10 & \text{if terminated and final pnl is negative} \end{cases}$$

The risk-reward R_{risk} is defined as the negative variance of the portfolio returns:

$$R_{\text{risk}} = -\sigma(R_p) = -\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \text{COV}(R_i, R_j)}$$

Where:

- V_t is the portfolio value at time t
- V_{t-1} is the portfolio value at time $t-1$

- $pctg_{return}$ is the final percentage return of the portfolio
- w_i and w_j are the weights of the assets in the portfolio
- R_i and R_j are the returns of assets i and j
- $\text{COV}(R_i, R_j)$ is the covariance between the last 30 step returns of assets i and j

3.5 Scalarized Expected Returns vs Expected Scalarized Returns

In contrast to single objective MDP, there are multiple optimality criteria for MOMDP. Optimization under each criterion can lead to completely different policies being learned. The choice of the optimality criterion depends on how the policy is aimed to be used in practice. The two optimality criteria are known as Scalarized Expected Returns (SER) and Expected Scalarized Returns (ESR). The SER criterion is calculated as follows:

$$V_u^\pi = u\left(E\left[\sum_{i=0}^{\infty}\gamma^i r_i \mid \pi, s_0\right]\right)$$

The SER framework is aimed at maximizing average return over multiple trading episodes. This approach is suitable when the policy can be executed repeatedly and what matters is the final average reward within the finite horizon. Contrary, if the focus is solely on the outcome of a single policy, ESR has to be optimized. ESR optimization is suitable when the policy is executed infrequently and each outcome is critically important. A good example of such an application would be an individual investor who would like to use such an algorithm to manage one's life savings. In such a scenario, each outcome of the policy execution is critical e.g. in case of a big drawdown investors may get scared and pull out the capital. Another example would be the execution of a trading policy over long trading windows e.g. annual portfolio rebalancing, where the policy is executed once in a significant period. The portfolio optimization problem formulated as the ESR problem is also suitable when market conditions are highly volatile e.g. cryptocurrency markets, making each outcome of each execution significantly more important than the average outcome over multiple runs. ESR criterion is formalized as follows.

$$V_u^\pi = E\left[u\left(\sum_{i=0}^{\infty}\gamma^i r_i\right) \mid \pi, s_0\right]$$

Given the focus of this thesis solely on retail investors, optimization of ESR criteria is more suitable. Individual investors typically have limited resources and each loss caused by an executed policy can be critical in terms of further investments. In contrast, optimization of SER is more suitable if solely average return over multiple policy executions is taken into consideration. This kind of approach is suitable for large financial institutions that have higher investment capital in their disposition. Thus optimization of SER is not suitable for retail investors. Pareto Condition Networks (PCN) are known MORL algorithm that

optimizes ESR. The following chapter will elaborate on the functionality of PCN and its application within this thesis.

3.6 Approaches

Portfolio optimization is a problem where a solution that balances multiple objectives needs to be found. A solution for which it is not possible to improve one objective without hampering the other is said to be Pareto-efficient. In a multi-objective setting finding all Pareto-efficient policies is known to be a computationally expensive process. The set of optimal solutions can grow exponentially with the number of objectives. Pareto Conditioned Networks is a known approach that scales efficiently with several objectives. It transforms an optimization problem into a classification one by training a single neural network. PCN associates past transitions with episode rewards that lead to it. Neural Network is trained in a way, that when it is conditioned on a given return it should emulate transitions that lead to it. The coverage set is a subset of all solutions that comprise Pareto-efficient policies. PCN can efficiently and effectively learn coverage sets in multi-objective sequential problems and it was demonstrated that it can do so in various environments [6]. Pareto Conditioned Networks are a well-suited method for portfolio optimization problems due to their scalability with the number of objectives. The use of PCN in this thesis provides a robust foundation for future experimentation beyond the optimization of risk and return. Additionally, PCN produces a set of Pareto-efficient solutions. It empowers retail investors to choose options that align with their objectives such as risk tolerance. Investors can choose suitable solutions from the set of produced policies that align with their investment strategies.

3.7 Environment

To facilitate the application of multi-objective reinforcement learning a trading environment was implemented [8]. The implemented environment allows for the use of already existing DRL and MORL methods. It was implemented using the MO-Gymnasium library, which provides a standard API to communicate between learning algorithms and environments. The aforementioned library allows for the creation of custom environments and also includes standard MORL algorithms that can be used further in the process. The environment allows for loading trading data which is explained in the next chapter. It takes a discrete action from the agent and transforms it into an allocation vector as described in one of the previous sections. Further, based on the allocation vector it re-balances portfolio in the following way.

1. It determines which assets have to be sold entirely and which asset allocations have to be lowered due to the new allocation vector.
2. It buys new assets and increases asset allocations if necessary based on the new allocation vector.

After performing an action, adequate rewards are calculated and returned to the agent with new state observation.

Several assumptions are made to simplify the trading environment and make it practical for experimentation.

- Liquidity of assets traded in such a trading environment is as high as possible. It implies that all orders at a certain price are filled immediately.
- The environment incorporates fixed transaction costs of 0.2%. In a standard exchange, each transaction comes at a cost.
- The environment is adjustable in terms of loading different data. Such flexibility alleviates the problem of handling data to conduct diverse experiments.
- The environment can be used in the single and multi-objective modes, allowing for comparison of DRL and MORL methods.
- Agent represents a retail investor that is managing relatively small or medium capital and executed trades do not impact the market state.
- The environment allows for using different episode lengths.

4 Experiments

4.1 Data

For the experiments, daily adjusted close price data of selected stocks from the S&P 500 index and cryptocurrencies is used. The S&P 500 stocks data spans from June 2012 until the end of 2023, while the cryptocurrencies data covers the period from January 2018 until the end of 2023. Data was extracted with the Python library Yahoo Finance. Data is divided into three experiment scenarios. The first scenario was created to effectively examine an episode's length impact on a PCN model's performance. The extended scenario was introduced to allow for an understanding of how the MORL method can scale with the increased complexity of the problem. Additionally, a second and third scenario was created to conduct a robust comparison of the MORL method with classical RL methods. Multiple scenarios allow for a thorough analysis of models' behavior in different markets and different market conditions.

S&P 500 - base scenario The base scenario consists of 8 selected stocks from the S&P 500 index. The scenario consists of highly-correlated stocks, high\low volatility stocks, and stocks from uncorrelated sectors.

- Highly correlated stocks from the tech sector: Apple Inc. (AAPL), Microsoft Corporation (MSFT). These two stocks are known to have a high correlation due to their involvement in the technology sector. Bot are large market capitalization companies that are often influenced by similar market factors.
- Uncorrelated stocks: Johnson & Johnson (JNJ), The Procter & Gamble Company (PG). Both companies operate in different sectors (healthcare and consumer goods, respectively) and are likely to have little correlation with each other and technology stocks.

- High volatility stocks: Tesla, Inc. (TSLA), Netflix, Inc. (NFLX). Both stocks show high volatility, with significant price fluctuations over short periods.
- Low volatility stocks: The Coca-Cola Company (KO), Visa Inc. (V). Both stocks are relatively stable and experience less volatility in their price movements.

S&P 500 - extended scenario To assess how algorithms perform having more stocks to choose from the base scenario has been extended by an additional 12 stocks. It creates a set-up where a total of 20 stocks are available. It adds complexity to the base scenario.

Cryptocurrencies Cryptocurrencies are known for high volatility with high price swings in short periods. This market is relatively new and rapidly evolving. The period of available data is also shorter than for S&P 500. The cryptocurrency market is known for high risks that can lead to high returns or extreme losses. It is beneficial to assess how MORL and RL algorithms can adapt to it.

Scenario	Tickers
Base S&P 500	AAPL, MSFT, JNJ, PG, TSLA, NFLX, KO, V
Extended S&P 500	AAPL, MSFT, JNJ, PG, TSLA, NFLX, KO, V, GOOGL, AMZN, META, JPM, UNH, HD, VZ, DIS, NVDA, MA, ADBE, IBM
Cryptocurrencies USD pairs	BTC-USD, ETH-USD, LTC-USD, XRP-USD, XMR-USD, DASH-USD, ETC-USD, ZEC-USD, DCR-USD, WAVES-USD

Table 1. Scenarios and their respective tickers

Data was split into two sets: training data and test data with a 70-30 split. Test data for S&P 500 data was split into two 2-year periods. The first period represents the 2020-2022 market, second one 2022-2024 market. Both market periods were very different. Thus the separation of those creates an additional evaluation layer in the process. It helps to assess how algorithms can perform in different market periods. Cryptocurrencies data followed the same ratio of a data split without additional split of test data.

4.2 Evaluation procedure

Pareto Condition Networks are used to address multi-objective problems in portfolio optimization by balancing risk and return. PCN generates a set of diverse strategies where each represents a different point on the Pareto front. Those

strategies can be used depending on the preference of the investor, all leading to different trade-offs between optimized objectives. To effectively measure the performance of learned strategies following metrics are used in the evaluation procedure.

- Final Return: refers to the total return generated by trading strategy during the evaluation period. Return is expressed as a percentage.
- Maximal Drawdown: is the maximal peak-to-trough decline in the portfolio during the evaluation period. It represents the maximal loss an investor can experience in the portfolio before reaching a new peak. It helps to understand the worst-case scenario of the strategy.
- Annualized Volatility: the measure used to quantify portfolio variability over the evaluation period.
- Sharpe Ratio: measures the risk-adjusted return of the portfolio. It represents an additional amount of return per unit of increase in risk.
- Calmar Ratio: assesses risk-adjusted return by comparing annualized return to the maximal drawdown.

4.3 Training process

During the training process each starting point of the episode window was sampled from the training dataset concerning the split explained in the previous section. Data points were sampled uniformly. In reinforcement learning the main metric that indicates the model's final performance and progress in training is the ability to maximize cumulative reward over the episode. In the MORL, since there are multiple objectives hypervolume is the metric that provides a comprehensive measure of convergence. Hypervolume calculates the volume in the objective space that a set of solutions dominates, given the specified reference point. By calculating the volume, the hypervolume metric offers a single scalar that summarises the solution's performance. PCN used in this thesis follows implementation from MORL-baselines library [7]. PPO and DQN used in the experiments follow StableBaselines3 implementation. Table 2 and table 3 present used hyperparameters for PPO and DQN respectively. PPO and DQN hyperparameters were tuned on S&P 500 data using grid search.

Parameter	Value
ppo_batch_size	64
ppo_learning_rate	1e-3
ppo_n_steps	100
ppo_time_steps	1e6
gamma	0.99

Table 2. PPO hyper-parameters

Parameter	Value
dqn_batch_size	32
dqn_exploration_final_eps	0.04
dqn_exploration_fraction	0.6
dqn_learning_rate	5e-4
dqn_learning_starts	500
dqn_time_steps	1e6

Table 3. DQN hyper-parameters

Choice of the reference point Reference point is a metric, based on which hypervolume is calculated. To train PCN using hypervolume it is crucial to set a reference point first. In the training process, a reference point was set based on the estimate of the minimal acceptable cumulative reward that can be achieved by the model. The profit reward was calculated based on the assumption that the model cannot lose more than 50% of the portfolio at each step. This assumption leads to the following results e.g. for an episode with a 100-day trading window. $R_{profit} = 99 * \log 0.5 - 10 \approx -40$ For volatility, the assumption is that the model should not exceed the average volatility of 55%, thus $R_{risk} = -100 * 0.55 = -55$. The final reference point is expressed as $[-40, -55]$.

Hyper-parameters optimization PCN hyper-parameters were optimized using grid search. Models were trained using 2 different seeds and the best model's hyper-parameters were chosen for all of the experiments. Reusing hyper-parameters in different training scenarios is justified by the computationally expensive process of hyper-parameters optimization.

Parameter	Value
max_buffer_size	720
num_er_episodes	20
num_model_updates	15
batch_size	128
hidden_dim	64
learning_rate	0.01
num_model_updates	15
time_steps	150,000

Table 4. PCN hyper-parameters

Most RL problems are episodic, which means a problem can be defined as a finite-horizon task with a terminal state. On the other hand, trading is recurrent and does not have explicit termination conditions. However, in a problem of algorithmic trading, an investor can define the length of a trading period by choosing how long he or she wants a learned policy to trade for. Normally investors do

not invest indefinitely but define horizon either in terms of the desirable returns or the allowed period to trade for. To directly answer the key research question in this thesis, which is how to model portfolio optimization as a MORL task it is crucial to define the horizon of a trading episode. Authors of Deep Reinforcement Learning for Optimal Portfolio Allocation paper define trading horizon as fixed 5-year windows [3]. Such an approach is proven to be effective during the training process. However to fully facilitate retail investors shorter trading horizons of 30, 100, and 252 days were used in the experiments. To answer the question of modeling the portfolio optimization problem as MOMDP it is crucial to understand how different episode length impacts trading performance and agent's actions. PCN was trained for 3 different episode lengths: 30, 100, and 252 days. Different trading window set-ups were compared with each other and the baseline. A baseline is defined as a simple strategy that allocates the entire initial capital equally over all assets available in the scenario. Models were compared on the test data. The best-performing set-up was used in further comparison with RL methods on the extended S&P 500 and cryptocurrency data. The PCN produces a Pareto front with non-dominated sets of policies. In the evaluation procedure for each configuration single policy was used from the Pareto front. Used policies reflect the best balance in our opinion in terms of expected return and acceptable level of risk.

4.4 Results

Choosing episode length To understand the impact of episode length on the trading performance and choose a trading window for further experiments PCN was trained for 3 different episode lengths. Set-ups were compared with each other using Base S&P 500 scenario data. The comparison was conducted in two different market periods. The first period is from 2020-2022 when selected stocks were first growing and at the end of the period started to plummet, the second one represents the market after 2022 when selected stocks mostly moved sideways. Model set-ups were examined by calculating average statistics from 30 sampled episodes. Table 5 represents the agent's average performance in episodes sampled from the 2020-2022 and 2022-2024 periods. To confirm the difference between strategies ANOVA test was conducted showing a significant difference between strategies with p-value ≤ 0.05. Exact differences showing the impact of episode length on episode return were confirmed with Tukey's range test. It can be observed in table 6 that a strategy with an episode length of 100 days consistently shows significant differences in episode returns compared to the other strategies across different market conditions.

Furthermore, due to uneven episode length set-ups were compared with each other in 2 year period where agents were trading continuously. Agents were compared in the 2020-2022 market which showed both bullish and bearish periods. Agents start with 100000 dollars as initial capital and at the end of each episode, the final portfolio value is calculated and passed to the agent as initial capital for the next trading window. In table 7 it can be observed that the strategy with 100 days 100-day-long episode window outperformed other strategies, including the

baseline strategy that reflects market trends. Change of the portfolio value for different agents is reflected in the figure 1. Based on the results episode length of 100 days was used in further comparison with RL methods.

Episode length	Return	Maximal drawdown	Volatility	Sharpe Ratio	Calmar Ratio
2020-2022 Market					
30	0.023	-0.051	0.190	1.297	0.039
100	**0.200**	**-0.107**	**0.225**	**1.788**	**0.021**
252	0.193	-0.076	0.136	1.285	0.009
2022-2024 Market					
30	0.004	-0.068	0.211	0.981	0.032
100	**0.043**	**-0.149**	**0.249**	**0.681**	**0.007**
252	-0.044	-0.162	0.163	-0.244	0.001

Table 5. Average portfolio metrics for S&P 500 base scenario across different periods

Episode length comparison	Mean Difference	p-value	Significance
2020-2022 Market			
100 vs 252	-0.0065	0.984	Not Significant
100 vs 30	**-0.1761**	**0.0001**	**Significant**
252 vs 30	**-0.1696**	**0.0001**	**Significant**
2022-2024 Market			
100 vs 252	**-0.0882**	**0.0019**	**Significant**
100 vs 30	-0.0389	0.2685	Not Significant
252 vs 30	0.0493	0.1248	Not Significant

Table 6. Tukey HSD test results for comparison of episode returns between different strategies in different market conditions

Episode length	Return	Maximal Drawdown	Volatility	Sharpe Ratio	Calmar Ratio
30	0.2384	-0.1830	0.1909	0.7402	0.7756
100	**0.4043**	**-0.2258**	**0.2278**	**0.9981**	**1.0389**
252	0.1591	-0.0961	0.1448	0.6381	0.9990
Baseline	0.0967	-0.2422	0.2646	0.3107	0.5860

Table 7. Portfolio metrics for continuous trading evaluation for different episode lengths in 2020-2022 market

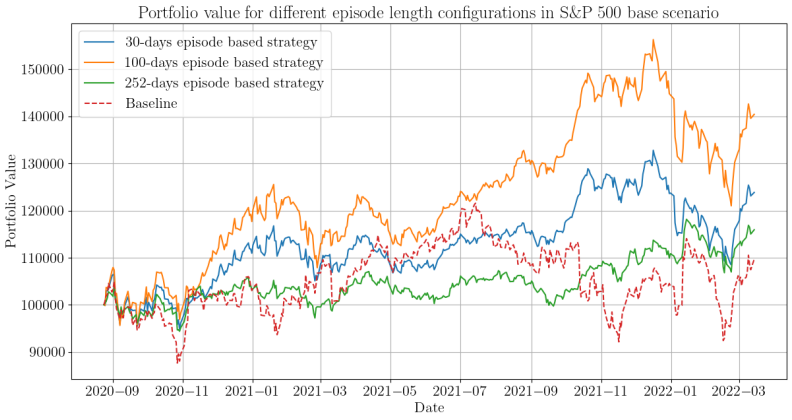

Fig. 1. Comparison of PCN managed portfolio value during 2020-2022 market for different episode length configurations.

Algorithm	Return	Maximal Drawdown	Volatility	Sharpe Ratio	Calmar Ratio
S&P 500 extended scenario					
2020-2022					
DQN	0.051	-0.070	0.156	0.847	0.009
PCN	0.088	-0.076	0.165	1.490	0.014
PPO	**0.105**	**-0.068**	**0.163**	**1.618**	**0.017**
2022-2024					
DQN	0.092	-0.100	0.195	1.461	0.021
PCN	**0.115**	**-0.102**	**0.207**	**1.655**	**0.023**
PPO	0.095	-0.104	0.204	1.456	0.021
Cryptocurrencies scenario 2022-2024					
DQN	-0.0095	-0.2584	0.4997	0.1637	0.0038
PCN	0.0222	-0.2393	0.4847	0.4310	0.0051
PPO	**0.0902**	**-0.2347**	**0.5290**	**0.7073**	**0.0082**

Table 8. Average portfolio metrics for S&P 500 extended scenario and cryptocurrencies scenario in different market periods

Models comparison To understand PCN scalability as well as MORL method performance with comparison to classical RL methods several experiments were conducted. The model was compared with PPO and DQN in the extended S&P 500 scenario as well as in the cryptocurrencies scenario. Experiments were conducted in two steps. The first step included sampling 30 trading periods from the data and then calculating average results for further comparison. The second step included the evaluation of models in a continuous fashion like for episode length comparison. PCN policy used for evaluation was selected from the learned Pareto front based on the best ratio between maximized objectives. Table 8 includes metrics obtained during sampling comparison. To understand the signif-

Algorithm	Return	Maximal Drawdown	Volatility	Sharpe Ratio	Calmar Ratio
S&P 500 Market					
DQN	0.1464	-0.1982	0.1853	0.4975	0.0018
PCN	0.2371	-0.1800	0.2029	0.7047	0.0032
PPO	**0.3322**	**-0.1653**	**0.1948**	**0.9626**	**0.0045**
Baseline	0.0832	-0.1993	0.2397	0.2869	0.0014
Cryptocurrency Market					
DQN	-0.5210	-0.6529	0.5521	-0.2765	-0.0009
PCN	-0.3358	-0.6769	0.5342	-0.0608	-0.0002
PPO	-0.0927	-0.5791	0.5443	0.1783	0.0007
Baseline	**0.0583**	**-0.6196**	**0.4381**	**0.2499**	**0.0007**

Table 9. Portfolio metrics of different algorithms in the extended S&P 500 scenario and the cryptocurrency market for continuous evaluation

Models Comparison	Mean Difference	p-value	Significance
S&P 500 Market			
2020-2022			
DQN vs PCN	-0.0375	0.0236	Not Significant
DQN vs PPO	0.0166	0.4646	Not Significant
PCN vs PPO	**0.0541**	**0.0006**	**Significant**
2022-2024			
DQN vs PCN	-0.0222	0.7035	Not Significant
DQN vs PPO	-0.0184	0.7846	Not Significant
PCN vs PPO	0.0038	0.9898	Not Significant
Cryptocurrencies Market 2022-2024			
DQN vs PCN	0.0318	0.7955	Not Significant
DQN vs PPO	0.0997	0.1123	Not Significant
PCN vs PPO	0.0679	0.3559	Not Significant

Table 10. Tukey HSD test results for comparison of episode returns between different models in the S&P 500 and cryptocurrency markets

icance of obtained results Tukey's range test was conducted and its results are presented in the table 10. Figures 2 and 3 present a change in portfolio value for continuous evaluation in the S&P 500 extended and cryptocurrencies scenario respectively. From conducted experiments, it can be concluded that the PPO algorithm demonstrates dominance in most market conditions. However, its superiority is most of the time not significant. The Pareto Condition Networks outperform PPO in the S&P 500 2022-2024 market. PCN also showcases its ability to scale with more stocks, generating a profit of 10% in both S&P 500 evaluation periods. Additionally, PCN finds a satisfying balance between profit and risk outperforming the baseline strategy that allocates investment capital among all stocks equally. S&P 500 Baseline strategy reflects market trend and PCN demonstrated its superiority to it. In the cryptocurrency market all tokens are correlated with bitcoin, the baseline strategy used for this comparison allocates all initial capital to bitcoin. All models, including PCN and PPO, failed to surpass the bitcoin baseline in the cryptocurrency scenario. This highlights

the inherent difficulty of the cryptocurrency market, which poses challenges for trading algorithms. It can be concluded that PCN can scale with more stocks and adapt to different market conditions. Additionally, PCN produces Pareto front of different policies. PCN uses obtained cumulative rewards to reenact transitions that lead to it. However, market behavior is stochastic and it can be hypothesized that learned policies during the training period cannot always be reenacted on evaluation data. PCN produces policies that can act differently on training data, however when policies are evaluated on evaluation data differences between them are minimal. This highlights a limitation where the training episodes might not always reflect future market conditions.

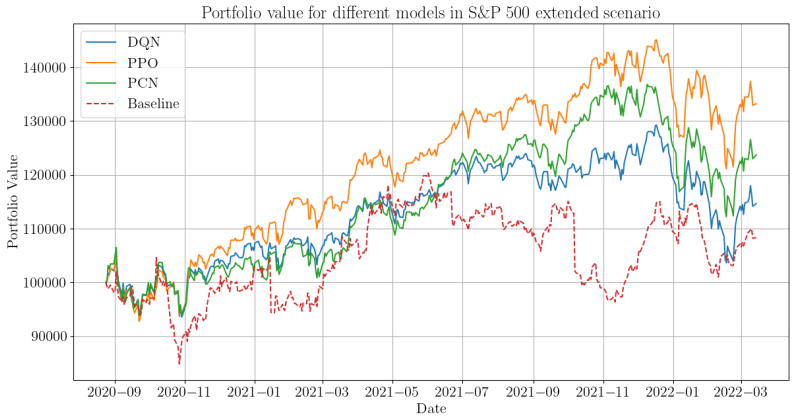

Fig. 2. Comparison of portfolio value for different agents in 2020-2022 market for S&P 500 extended scenario during continuous trading evaluation.

5 Conclusions

The application of Multi-Objective Reinforcement Learning represents a novel way of tackling financial decision-making problems. This thesis explored the integration of MORL frameworks, specifically Pareto Conditioned Networks (PCN), in addressing the objectives of profit maximization and risk minimization in the portfolio optimization problem. The portfolio optimization problem was defined across different markets including equities from S&P 500 index and cryptocurrencies. The study began by formalization of portfolio optimization problem as a Multi-Objective Markov Decision Process, introducing a reward function for MORL and state representation. Through experiments and evaluations, PCN demonstrated its efficacy in generating solutions that can manage the risk and profit of the portfolio. Key metrics such as final return, maximal drawdown,

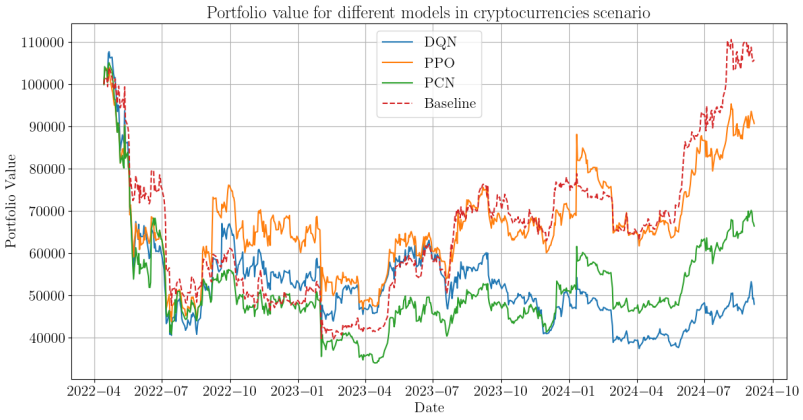

Fig. 3. Comparison of portfolio value for different agents in 2022-2024 market for cryptocurrencies scenario during continuous trading evaluation.

volatility, Sharpe ratio, and Calmar ratio were used to evaluate the performance of the MORL method and single objective methods such as PPO and DQN. The PCN agent showed its ability to learn policies that can balance risk and reward on a satisfactory level, however, did not manage to outperform PPO in the S&P 500 and cryptocurrencies market. The PPO agent outperformed the PCN agent in most of the scenarios, however the difference in performance was not significant. The study showed that the choice of the trading window significantly impacts the performance of the PCN agent and a 100-day window yields better results in comparison with a shorter 30-day window and a longer one that was 252 days. In conclusion, PCN demonstrates an alternative method to manage profit and risk in portfolio optimization problems. Continued exploration of the application of MORL to portfolio optimization holds the potential for addressing different investors' needs by experimenting with a bigger set of objectives.

References

1. John Cotter and Enrique Salvador, "The non-linear trade-off between return and risk and its determinants," *Journal of Empirical Finance*, vol. 63, pp. 1-23, 2022.doi.org/10.1016/j.jempfin.2022.03.002.
2. Zihao Zhang, Stefan Zohren, and Stephen Roberts, "Deep Reinforcement Learning for Trading,", 2019. doi.org/10.48550/arXiv.1911.10107.
3. Srijan Sood, Kassiani Papasotiriou, Marius Vaiciulis, and Tucker Balch, "Deep Reinforcement Learning for Optimal Portfolio Allocation: A Comparative Study with Mean-Variance Optimization," *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2023. icaps23.icaps-conference.org/papers/finplan/FinPlan23_paper_4.pdf.

4. John Moody and Matthew Saffell, "Reinforcement Learning for Trading," *Advances in Neural Information Processing Systems (NIPS)*, 1998. proceedings.neurips.cc/paper_files/paper/1998file/4e6cd95227cb0c280e99a195be5f6615-Paper.pdf.
5. C.F. Hayes, R. Rădulescu, E. Bargiacchi, et al., "A practical guide to multi-objective reinforcement learning and planning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 2, 2022. doi.org/10.1007/s10458-022-09552-y.
6. Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé, "Pareto Conditioned Networks," *arXiv*, 2022. doi.org/10.48550/arXiv.2204.05036.
7. Florian Felten, Lucas N. Alegre, Ann Nowé, Ana L. C. Bazzan, El Ghazali Talbi, Grégoire Danoy, and Bruno Castro da Silva, "A Toolkit for Reliable Benchmarking and Research in Multi-Objective Reinforcement Learning," 2023. arxiv.org/abs/2204.05036.
8. Jakub Lewkowicz, github.com/JLLEW/MORL-Thesis