Lisbon School of Economics & Management
Universidade de Lisboa

MASTER

MASTER IN FINANCE

MASTER´S FINAL WORK

DISSERTATION

PORTFOLIO SELECTION FOR BENCHMARK TRACKING: A COMPARISON BETWEEN MIXED-INTEGER LINEAR PROGRAMMING AND REINFORCEMENT LEARNING

LUCAS FIRMINO MELO PEREIRA

JULY - 2026

Lisbon School of Economics & Management
Universidade de Lisboa

MASTER

MASTER IN FINANCE

MASTER´S FINAL WORK

DISSERTATION

PORTFOLIO SELECTION FOR BENCHMARK TRACKING: A COMPARISON BETWEEN MIXED-INTEGER LINEAR PROGRAMMING AND REINFORCEMENT LEARNING

LUCAS FIRMINO MELO PEREIRA

SUPERVISION:

RAQUEL M. GASPAR

JULY - 2026

GLOSSARY

EIT Enhanced Index Tracking.

FNN Feedforward Neural Network.

GAE Generalized Advantage Estimation.

GRU Gated Recurrent Unit.

IT Index Tracking.

MILP Mixed-Integer Linear Programming.

PPO Proximal Policy Optimization.

RL Reinforcement Learning.

S&P Standard & Poor's.

TE Tracking Error.

ABSTRACT

The field of portfolio selection offers multiple challenges not only by the stochastic nature of asset prices but also by the fact that even when a predictive model is considered, it can produces an objective and a domain that is often non-convex and highly dimensional and the task of searching for an optimal solution can be computationally prohibitive, requiring specific methodologies for the search.

The problem analyzed in this dissertation is the Index-tracking problem, traditional in the passive investment industry, which is when a portfolio is designed to mirror a given benchmark. This allows a series of methods to emerge with a few examples being deterministic numerical optimizations, stochastic metaheuristics and machine learning. Two distinct methods are evaluated here, a value-based Mixed Integer Linear Programming (MILP) and a Reinforcement Learning (RL) approach based on proximal policy optimization with an architecture based both on a feedforward neural network and a Recurrent Neural Network design. Regarding the benchmark, a synthetic index composed of 50 assets, with quarterly rebalances, is built from S&P 500 constituents for the period of 2006 - 2025.

Both methodologies are compared under equivalent data and computational capacity and evaluated on annual windows from 2018 - 2025 using weekly portfolio rebalance. Evaluation happens on both return-based (R-TE) and value-based(V-MAD) metrics.

The MILP achieves a mean V-MAD of approximately 0.10% and consistently outperforms the RL on the reported metrics, with an equivalent 1.48% V-MAD, while the difference in transaction costs is approximately 0.2%, indicating that the correct asset allocation plays a determinant role for the superior accuracy of the method of historical similarities.

KEYWORDS: Portfolio Selection; Index-Tracking; Passive Investment; Mixed-Integer; Reinforcement Learning.

JEL CODES: C61, G11, C45, C44, G23, C63

RESUMO

O campo da seleção de portefólios apresenta múltiplos desafios, não só pela natureza estocástica dos preços dos ativos, mas também pelo facto de que, mesmo quando se considera um modelo preditivo, este pode produzir um objetivo e um domínio frequentemente não convexos e de elevada dimensionalidade, onde a tarefa de busca por uma solução ótima pode ser computacionalmente proibitiva, exigindo metodologias específicas para a mesma.

O problema analisado nesta dissertação é o problema de index-tracking, tradicional na indústria de investimento passivo, que consiste em desenhar um portefólio que replique um determinado benchmark. Isto permite o surgimento de uma série de métodos, sendo alguns exemplos as otimizações numéricas determinísticas, as metaheurísticas estocásticas e a aprendizagem de máquina. São avaliados dois métodos distintos: uma programação linear inteira mista (Mixed-Integer Linear Programing - MILP) baseada em valor e uma abordagem de aprendizagem por reforço (RL) baseada em proximal policy optimization, arquitetada tanto numa rede neural feedforward como num desenho de Rede Neural Recorrente. Sobre o benchmark, é construído um índice sintético composto por 50 ativos, com rebalanceamentos trimestrais, a partir de constituintes do S&P 500 para o período de 2006 - 2025.

Ambas as metodologias são comparadas sob dados e capacidade computacional equivalentes e avaliadas em janelas anuais de 2018 a 2025, com rebalanceamento semanal do portefólio. A avaliação ocorre tanto em métricas baseadas em retorno (R-TE) como baseadas em valor (V-MAD).

A MILP atinge um V-MAD médio de aproximadamente 0.10% e supera consistentemente o RL nas métricas reportadas, com um V-MAD equivalente de 1.48%, enquanto a diferença nos custos de transação é de aproximadamente 0.2%, indicando que a alocação correta de ativos desempenha um papel determinante na precisão superior do método de similaridades históricas.

PALAVRAS-CHAVE: Replicação de Índices; Investimento Passivo; Mixed-Integer Linear Programming; Reinforcement Learning.

JEL CODES: C61, G11, C45, C44, G23, C63

TABLE OF CONTENTS

List of Figures

List of Tables

ACKNOWLEDGEMENTS

To my family, for their unconditional support and encouragement to pursue my goals.

I also would like to thank my supervisor for the guidance provided and all the meaningful discussions throughout the project.

Finally, I would like to express my gratitude to Instituto Turismo de Portugal for awarding me the Bolsa Luís Patrão, whose support has contributed significantly to this work.

1 INTRODUCTION

The index tracking problem consists in searching for an asset combination that best replicates an index performance. The existence of costs such as broker fees and bid-ask spreads or even the necessity of a certain investor to hold a reduced number of assets is sufficient to introduce the problem of searching for an optimal allocation in a subsample of the original assets. The very definition of index performance has been subject to debate through the years where simply replicating returns may be seen as insufficient, giving space to metrics based on value tracking.

Even setting aside the stochastic nature of price evolution, search for an optimal subset in problems like this is a hard combinatorial task where computational needs can grow exponentially as the number of assets increases or the number of constraints change and the search for methods that can answer for this optimal allocation are in constant development.

A lot of the literature also goes beyond simple tracking performance introducing more complexity into the model-environment dynamics such as cash withdrawals, expected shortfall constraints and others.

This work aims to compare two of these methods, a Mixed-Integer Linear Programming (MILP) (Strub & Baumann (2018)) that optimizes an objective function over a past window of data determining the next portfolio weights and a Machine Learning (ML), more specifically a Reinforcement Learning (RL) (Peng et al. (2024)) that train over historical data in order to predict the next holdings, both methods are tested under weekly (5 working days) protfolio rebalances[1].

The index chosen is a synthetic index, customized with 50 assets sampled from the S&P 500, distributed among the GICS[2] sectors, and with the requirement that those assets were part of the index since 2006, when the training window began.

This work tries to put both methods in terms of computation capacity, which means bringing the ML to a low-cost basis compared to its usual domain of high-computation demands. Considering a Intel Core i7-12700 (12 cores, 2.1 GHz), 32 GB RAM, Windows 11 Enterprise setup, the epoch limitation was set to fit within a 90-minute training window.

It is found that the MILP consistently achieves lower tracking error across all sampled years, displaying that the optimization over historical data is a strong method, the RL

[1] not to be confused with the index rebalance itself
[2] Global Industry Classification Standard

approach demonstrates competitive value-path tracking although its deviations are still at a higher magnitude.

The remainder of this dissertation is organized as follows. Section 2 reviews the relevant literature. Section 3 describes the data, index construction, and the formulation of both methods. Section 4 presents the results, beginning with a standalone analysis of both the MILP and the RL - this is done with zero fixed costs in order to reduce computational costs and the idea is to understand how both methods perform in the tracking task individually, allowing to analyze how lookback windows, training windows and model architectures affect the performance and finally both of the methods are put together in an out-of-sample comparison between the best-performing configurations of each method. Section 5 concludes with a discussion of strengths and limitations and potential for future work.

2 LITERATURE REVIEW

There is a large spectrum of methods for index-tracking problems which fundamentally varies along two dimensions. The first one is how to define the objective formulation - often called tracking-error - which can be quadratic, linear, downside-biased, return-based or value-based, and may operate over a historical window or attempt to forecast future deviations. The second main dimension is, once this objective is defined, how to search for it, i.e. what optimization procedure to use: ranging from deterministic numerical methods such as mixed-integer programming to stochastic heuristics and machine learning.

Roll (1992) used the traditional mean-variance theory (Markowitz (1952)) to elaborate on the benchmark tracking problem, with the proposal where the variance between portfolio returns and benchmark returns must be minimized while maintaining a certain level of excess returns, its also shown that this solution lies in the so called TEV (Tracking Error Variance) frontier, a region that is characterized by suboptimal portfolios when compared with the efficient frontier. Rudolf et al. (1999) depart from Roll's quadratic formulation of mean squared deviation of returns to develop four linear optimizations extending from the absolute deviation of portfolio returns to adaptations with focus on downside risk. The argument that investors are more sensitive to downside risk, leading to specific downside risk metrics, is further analyzed by Rockafellar et al. (2006) and Stoyanov et al. (2008).

With the argument that correlation methods are incapable of capturing non-stationary behavior Alexander & Dimitriu (2005) approaches the construction of the portfolio via cointegration with the benchmark index. The method is then evaluated with the Dow Jones Industrial Average (DJIA) as benchmark over different portfolio rebalance schedules ranging from 2 weeks to 6 months.

Beasley et al. (2003) investigates a non-linear return-based objective, over a historical window, making use of a population heuristic for the optimization, using hard cardinality[3] constraint for the number of asset held and explicitly accounting for transaction costs, the authors also contribute with the constitution of five benchmark datasets (Hang Seng, DAX 100, FTSE 100, S&P 100, Nikkei 225). Canakgoz & Beasley (2009) adopts a similar set of constraints but reformulate its objective as the regression between portfolio and benchmark returns, minimizing the deviation of the intercept to zero and the slope to one, also linearizing the problem, which is then optimized via Mixed-Integer Programming, it is also possible to switch the problem to an enhanced-indexation by adjusting the intercept

[3] A cardinality constraint is called hard when the number of assets held must be exactly equal a certain amount k.

in this setup, a different category of research that attempt to produce an excess return over the benchmark. In this case, the benchmark makes use of the one created by Beasley et al. (2003) and extends it to the S&P 500, Russell 2000 and Russell 3000.

Guastaroba et al. (2009*b*) evaluate multiple scenario generation techniques (historical, bootstrap, Monte Carlo and GARCH-based) under a CVaR framework for a single-period problem using the FTSE100 as benchmark. Guastaroba et al. (2009*a*) extend this to multi-period portfolio rebalancing schedule, modelling the trade-off between tracking accuracy and transaction costs across successive rebalancing dates.

Guastaroba & Speranza (2012) objective formulation minimizes the absolute deviation between the normalized trajectories of portfolio value and index, subject to a cardinality limit of the type at most of ($\leq$) and uses proportional and fixed costs, the optimization is done via kernel search, a procedure that introduce an intermediate step of selecting a candidate group of assets between the relaxation and the mixed integer, in order to optimize the MILP search.

Mezali & Beasley (2014) evaluates two objective formulations for the MILP, the MINIMAX and MINIAVERAGE which minimize the maximum(worst case) and the average difference returns between portfolio and benchmark respectively, it also distinguish transaction costs by buy and selling movements and finally, evaluates it under Canakgoz & Beasley (2009) database.

Strub & Baumann (2018) redefines the value-based formulation in order to directly account for transaction costs. The model also designed to account for investors cash deposit and withdraws, turning the environment more dynamic than the simple index rebalance. The model is calibrated over a 104 week period and solved sequentially at a 52-week investment horizon over a multiple range of instances, which represents different indexes and cash injection/withdraw status, aside from the standard cardinality constraints imposed, the paper also presents the non-restrictive formulation, where the number of assets is unconstrained in order to search for the minimum deviation.

Some solutions are designed to target specific types of portfolios, like when portfolios get too large, often mentioned as sparse portfolios, an approach called regression with regularization appears as a strong candidate. Benidis et al. (2018) points that methods like the above mentioned Mixed-Integer can start failing as running time increases, in this case, a quadratic return based objective function is adapted by adding a penalty component that helps controlling portfolio sparsity. It is then solved by a majorization-minimization framework. Chen et al. (2022) also adopts a quadratic return based optimization but adapts the coefficient of the penalty component in a way that recent data weights more than past data, this time-weighted function allows to prioritize information coming from

recent data and the data used for the empirical analysis is the one of Canakgoz & Beasley (2009).

On the machine learning branch, Peng et al. (2024) formulates the problem as a dynamic programming where a state vector comprised of stock prices, volumes and other metrics is fed into a feed-forward neural network in order to access the next portfolio weights. The problem is also stated with cash injections and a banach fixed point algorithm is used for the computation of costs, which follows the interative brokers cost formulation. The network is trained via Proximal Policy Optimization (PPO) (Schulman et al. (2017)), a reinforcement-learning framework, and tested in a rolling window scheme. The study uses three indexes: S&P500, S&P 500 Equally Weighted and the DJIA.

This work contributes to the current literature by addressing an existing gap in methodological comparison. Both approaches, the MILP based on the work of Strub & Baumann (2018) and the RL originally based on the work of Peng et al. (2024), are evaluated under similar processing capacity constraints, under the same available data and the same testing windows.

3 DATA AND METHODOLOGY

3.1 Overview and training scheme

This section describes the data and index construction common to both methods, and introduces the two methodological frameworks. The MILP formulation, presented in Section 3.2, follows Strub & Baumann (2018) and is described in both a non-restrictive (when no cardinality constraint is imposed) and restrictive modes (when a cardinality constraint k is imposed, which means that a maximum of k assets is allowed), both of this sets can also be described as Self-Financed (SF), which means that the transaction costs must be paid with its own funds. The Reinforcement Learning formulation, presented in Section 3.3, follows Peng et al. (2024) formulation and is evaluated in two configurations: a long-term model (RL-LT) trained over 12 years of past data with a 1-year lookback window[4], and a short-term model (RL-ST) trained over 2 years with a 30-day lookback window, the latter motivated by the findings of Dai & Li (2024) on the effectiveness of short observation windows. A third RL variant, RL-ST-GRU, extends the short-term model with a Gated Recurrent Unit (GRU) encoder, described in Section 3.3.1. The two methods are then evaluated on a common set of metrics, defined in Section 3.4.

A few assumptions: no short selling is allowed and both methods assume infinite share divisibility, in accordance with Strub & Baumann (2018), which is an approximation that holds under large portfolios.

The two methods differ fundamentally in how they incorporate historical information. The MILP is re-solved from scratch at each portfolio rebalance date over a fixed past window, retaining no information beyond the previous portfolio weights. The RL agent, by contrast, accumulates knowledge in the parameters of its neural network throughout training, so that experience from earlier periods shapes the policy applied at each subsequent rebalance. Despite this difference, both methods share the same operational structure at inference time: a lookback window of past returns is used to determine the portfolio weights held until the next rebalance date, the illustration in Figure 1 provides an idea on how the Reinforcement Learning is trained.

3.1.1 Index and Stock Universe

For this analysis a 50-stock index is built based on top constituents of the S&P 500, selected following the criteria of highest market-cap with at least one representative per GICS sector. The reason behind this choice is to reduce the computation time needed.

[4]for a given point in time, a lookback window is how much the method consume from past data

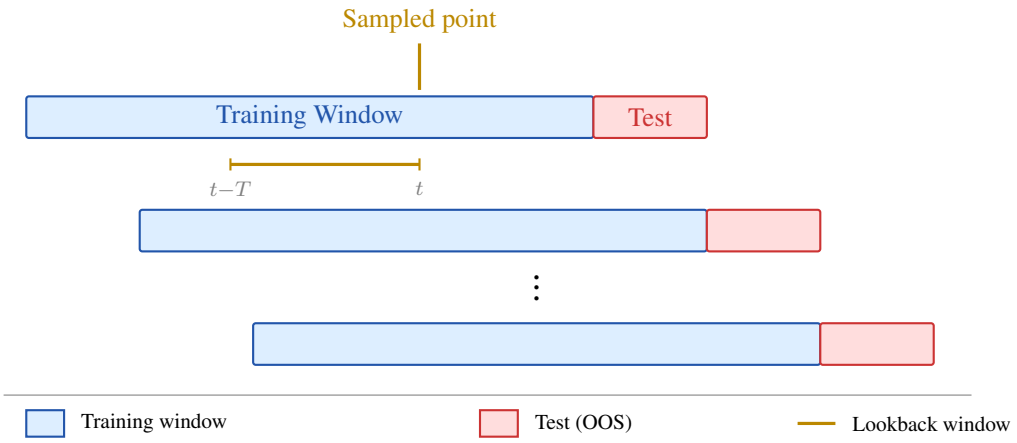

FIGURE 1: rolling windows scheme

Another consideration is that its stocks have been listed since 2006 (this guaranties that it is comprised of stocks with a complete price history over the training window) and distributed among sectors. Although this introduces survivorship bias to index it is not considered to be a concern, given that the concentration existance of long-living stocks, observed on the way the index overcomes the S&P 500 on Figure 2, is by itself an interest fact to evaluate how the models adapt. The index is also built to rebalance quarterly, with a total of 76 rebalances, in order to offer some transient behavior to test the adaptability of the methods, the rebalance dates occur mostly at the closing of March, June, September and December of each year.

The index is constructed as a capitalisation-weighted index using daily closing prices[5] over the period from January 2006 to December 2025.

$$w_{j,\tau_k} = \frac{\text{MC}_{j,\tau_k}}{\sum_{i \in J} \text{MC}_{i,\tau_k}}, \qquad j \in J, \tag{1}$$

where MC_{j,τ_k} denotes the market capitalization of stock j on date τ_k, computed as the product of its adjusted closing price and shares outstanding. In order to ensure continuity to the index when weights are rebalanced, it is then linked to its previous values as shown in Equation (2)

$$I_t = I_{t-1} \times \sum_{j \in J_{\tau_k}} w_{j,\tau_k} r_{j,t}\,, \tag{2}$$

where I_t is the index value at a point t in time, τ_k is the last rebalance date and $r_{j,t}$ is the asset j returns ensuring no discontinuous jumps. Figure 2 below displays the index

[5] All data was obtained from Bloomberg terminals

evolution through 2006 until 2025.

FIGURE 2: index evolution vs. S&P 500

No auxiliary market variables - such as the VIX index, Treasury bill rates, or trading volumes - are used in this database construction, which is an attempt to put both methods in the same ground with respect of data availability. It was also a deliberate choice to not adjust for dividends given the multiple stock-splits, reducing complexity.

The volatility profile (30 day vol.) can be seen in Figure 3, it remains similar to the S&P 500.

FIGURE 3: 30 day index volatility

Observing its large cap weights - above 5% - in Figure (4) it is possible to see the post-pandemic concentration into the tech mega-cap, another characteristic to evaluate on how both methods approach it.

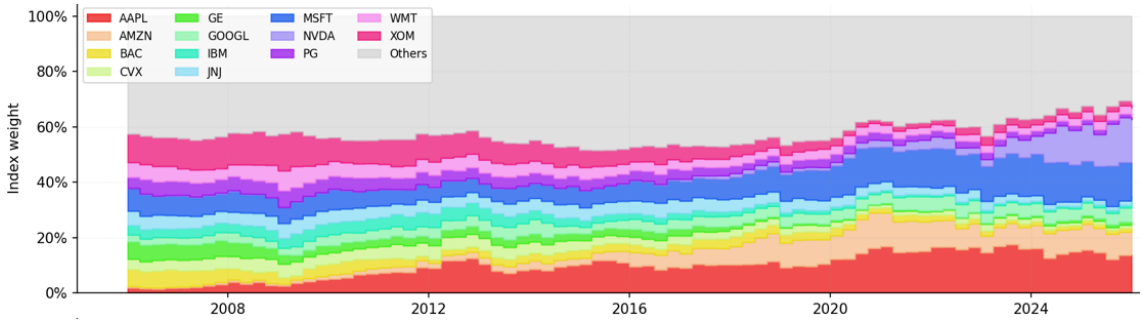

FIGURE 4: synthetic index: constituents weights

3.2 The Mixed Integer Linear Programming Formulation

In the literature, programming problems fundamentally address the problem of optimizing an objective under a set of constraints, usually related to problems of scarce resources allocation. Multiple variations are possible but specifically the linear programming are the formulations where both the function to be optimized and the constraints are linear and Integer programming uses integer constraints into the optimization. When both of these constraints are applied together, the setup is defined as a Mixed-integer Linear Programming (MILP) model. With an example on Equation (3)

$$\begin{aligned} \min_{x} \quad & c^{\top} x \\ \text{subject to} \quad & Ax \geq b \\ & x \geq 0 \\ & x_j \in \mathbb{Z}, \quad j \in \{1, \ldots, p\}, \end{aligned} \tag{3}$$

These problems are usually solved by a procedure that involves a relaxation followed by a branch and bound mechanism, this concept is not developed here but the reader is advised to the works of Cornuejols & Tütüncü (2007) for more details. For the current work, the whole setup is written in python, using Pyomo (Bynum et al. (2021) for more details), which is an algebraic modeling API, to instantiate the objective function and constraints and then call Gurobi®, the solver.

The first two optimizations performed derive from the work of Strub & Baumann (2018), the non-restrictive and restrictive formulation. In the non-restrictive case, the solver is let free to find the optimal number of constituents - in contrast with the restrictive case, where a cardinality constrain is imposed. Both models are also self-financed, which means that the transaction costs must be deduced from the capital available under investment, eroding the funds value over time, through the text both can be read under the acronyms of NR-SF and R-SF respectively.

3.2.1 *The Objective Function*

Both the Non-Restrictive and Restrictive settings are solved using the same optimization criterion,

$$f(\mathbf{x}) = \sum_{t=1}^{T} \left| Q_t(\mathbf{x}) - I_t \frac{Q_T(\mathbf{x})}{I_T} \right| + \sum_{t=1}^{T} I_t \frac{C_T - Q_T(\mathbf{x})}{I_T}, \tag{4}$$

where $\mathbf{x} = [X_{1T}, \ldots, X_{nT}]$ is the number of units of stock j held in the portfolio and $Q_t(\mathbf{x}) = \sum_{j \in J} q_{jt} X_{jT}$ denotes the value of the tracking portfolio at time t, where q_{jT} is the closing price of stock j at time T. I_t is the index value at time t, the ultimate objective to be tracked and C_T the fund capital at T, the sum of all positions deduced by the acumulated transaction costs at the end of the experiment T as displayed on Equation (8).

The first term of Equation (4) measures the distance between the historical portfolio trajectory Q_t and the target I_t normalized by $Q_T(\mathbf{x})/I_T$, this ensures for example, that a portfolio of million dollars and an index of thousands of points are normalized to a similar scale. the summation across all point in time t ensures that all deviations weight the same. So minimizing this deviation is maximizing the similarity between trajectories of the portfolio value and the index. The second term is similar to the index re-scaling done in the first component, but this time it is scaling the index trajectory by the transaction costs. It penalizes the gap between the fund capital C_T and the invested portfolio value $Q_T(\mathbf{x})$ that is the accumulated transaction costs G_T, accounting for both variable and fixed costs incurred during transactions.

The objective is linearized by introducing non-negative auxiliary variables $u_{tT}, d_{tT} \geq 0$ representing the positive and negative deviations respectively:

$$\min_{\mathbf{x}} \quad \sum_{t=1}^{T} \left(d_{tT} + u_{tT} + I_t \frac{C_T - Q_T(\mathbf{x})}{I_T} \right) \tag{5}$$

$$\text{s.t.} \quad u_{tT} - d_{tT} = Q_t(\mathbf{x}) - I_t \frac{Q_T(\mathbf{x})}{I_T}, \quad t \in \{1, \ldots, T\} \tag{6}$$

$$d_{tT},\, u_{tT} \geq 0, \quad t \in \{1, \ldots, T\}, \tag{7}$$

where u_{tT} and d_{tT} are strictly positive, so $u_{tT} + d_{tT} = |Q_t(\mathbf{x}) - I_t Q_T(\mathbf{x})/I_T|$ minimizing the set above is equivalent to minimizing Equation (4). Both the Restrictive and Non-Restrictive formulations are solved using the Gurobi optimizer with a time limit of 3,000

seconds per portfolio rebalancing decision.

Summarizing, at each rebalancing date both MILP variants solve an optimization problem over the set of index constituents $J = \{1, \ldots, n\}$. and over the past window of time $t = \{0, \ldots, T\}$, the solution are the weights to be held until the next portfolio rebalance date, and the process is repeated.

3.2.2 *The Non-Restrictive set of Constraints*

Equations (8) to (17) present the set of non-restrictive constraints. Equation (8) enforces full investment. The fund capital C_T is allocated in assets $Q_T(\mathbf{x})$ or acumulated in transaction costs $\sum_{j \in J} G_{jT}$.

With Y_{jT} representing the amount of stocks currently held, Equation (9) decomposes the net change in holdings into a gross purchase flow v^b_{jT} and a gross sale flow v^s_{jT}, both continuous and non-negative. Their difference equals the change in dollar value of stock j: a positive value indicates a net purchase, a negative value a net sale. It can be seen as a similar mechanism as the done with variables u_{tT} and d_{tT} in Equation (7)

Non-Restrictive set

$$
\begin{cases}
\displaystyle\sum_{j \in J} (q_{jT} X_{jT} + G_{jT}) = C_T & (8) \\
v^b_{jT} - v^s_{jT} = q_{jT}(X_{jT} - Y_{jT}) (j \in J : Y_{jT} > 0) & (9) \\
G_{jT} = c^b_j v^b_{jT} + c^s_j v^s_{jT} + c^f_j (w^b_{jT} + w^s_{jT}) (j \in J : Y_{jT} > 0) & (10) \\
G_{jT} = q_{jT}\, c^b_j X_{jT} + c^f_j w^b_{jT} (j \in J : Y_{jT} = 0) & (11) \\
\zeta_{jT} C_T w^b_{jT} \leq v^b_{jT} \leq \eta_{jT} C_T w^b_{jT} (j \in J : Y_{jT} > 0) & (12) \\
\zeta_{jT} C_T w^b_{jT} \leq q_{jT} X_{jT} \leq \eta_{jT} C_T w^b_{jT} (j \in J : Y_{jT} = 0) & (13) \\
\zeta_{jT} C_T w^s_{jT} \leq v^s_{jT} \leq \eta_{jT} C_T w^s_{jT} (j \in J : Y_{jT} > 0) & (14) \\
w^b_{jT} + w^s_{jT} \leq 1 (j \in J : Y_{jT} > 0) & (15) \\
v^b_{jT} \geq 0,\ w^s_{jT} \in \{0, 1\},\ v^s_{jT} \geq 0 (j \in J : Y_{jT} > 0) & (16) \\
w^b_{jT} \in \{0, 1\},\ X_{jT} \geq 0,\ G_{jT} \geq 0 (j \in J) & (17)
\end{cases}
$$

Transaction costs for each stock j include proportional costs for purchasing (c^b_j) and selling (c^s_j) as a fraction of the transaction value, and a fixed cost (c^f_j) per trade. The total transaction cost for stock j is G_{jT} (Equation (10)). G_{jT} reflects the exact transaction cost rather than an upper bound. This exact computation is what allows minimum transaction values to be modeled correctly.

A minimum transaction value $\zeta_{jT}C_T$ for any stock that is traded, alongside the existing maximum $\eta_{jT}C_T$ is introduced in Equations (12), (13) and (14), this bounds act as protection in case of infeasible solutions.

Two binary variables: (w^b_{jT}) for purchases and (w^s_{jT}) for sales, in a mutual exclusion constraint $w^b_{jT} + w^s_{jT} \leq 1$ ensures that v^b_{jT} and v^s_{jT} cannot both be strictly positive simultaneously, preventing buy and sell actions at the same asset at the same rebalance period.

The formulation further exploits the observation that any stock not currently held ($Y_{jT} = 0$) cannot be sold under the no-short-selling restriction. For such stocks, the sale variables v^s_{jT} and w^s_{jT} are structurally zero and are removed. Moreover, since any units acquired must come entirely from a new purchase, the purchase value equals $q_{jT}X_{jT}$ directly, eliminating the need for the auxiliary variable v^b_{jT} for these stocks.

The mutual exclusion constraint $w^b_{jT} + w^s_{jT} \leq 1$, defined for currently held stocks, ensures that purchases and sales cannot occur simultaneously for the same stock in a single rebalancing period. Together with the separate buy and sell binaries, this guarantees that G_{jT} reflects the exact transaction cost rather than an upper bound — a property that is necessary for the minimum transaction value $\zeta_{jT}C_T$ to be modeled correctly. The final two lines (Equations (16) and (17)) state the variable domains: all flow variables are non-negative continuous, trade indicators are binary, the decision variable $X_{jT} \geq 0$ enforces the no-short-selling restriction and $G_{jT} \geq 0$ guarantees costs are positive.

3.2.3 The Restrictive constraints

The R-SF setting, Equations (18) to (25), imposes the additional cardinality constraint limiting the number of stocks in the portfolio to $\leq k$, and requires the value of each held stock to lie within a defined range $[\varepsilon_{jT}C_T,\ \delta_{jT}C_T]$, where ε_{jT} and $\delta_{jT}C_T]$ are respectively lower and upper bounds for the amount held on a single stock j. The binary variable z_{jT} equals one if stock j is held, and w_{jT} equals one if it is traded. The value purchased and sold are denoted v^b_{jT} and v^s_{jT} respectively.

$$\textbf{Restrictive set} \begin{cases} \sum_{j \in J} (q_{jT} X_{jT} + G_{jT}) = C_T & (18) \\ v^b_{jT} - v^s_{jT} = q_{jT}(X_{jT} - Y_{jT})(j \in J) & (19) \\ G_{jT} = c^b_j v^b_{jT} + c^s_j v^s_{jT} + c^f_j w_{jT}(j \in J) & (20) \\ v^b_{jT} + v^s_{jT} \leq \eta_{jT} C_T w_{jT}(j \in J) & (21) \\ \varepsilon_{jT} C_T z_{jT} \leq q_{jT} X_{jT} \leq \delta_{jT} C_T z_{jT}(j \in J) & (22) \\ \sum_{j \in J} z_{jT} \leq k & (23) \\ w_{jT} \in \{0,1\},\ z_{jT} \in \{0,1\}(j \in J) & (24) \\ v^b_{jT} \geq 0,\ v^s_{jT} \geq 0,\ X_{jT} \geq 0,\ G_{jT} \geq 0(j \in J) & (25) \end{cases}$$

The Table I presents the parameters for both non-restrictive and restrictive sets.

Parameter	**NR-SF**	**R-SF**
Data		
Lookback window T	252 days	252 days
Portfolio Rebalance frequency	Weekly (5 working days)	
Out-of-sample period	2,000 trading days (2018–2025)	
Initial capital C_T	\$100	
Transaction Costs		
Proportional buy cost c^b_j	0.001 (0.1%)	
Proportional sell cost c^s_j	0.001 (0.1%)	
Fixed cost per trade c^f_j	$0,00005$ (\$5.0 per \$10 million)	
Trade Size Limits		
Min. transaction fraction ζ_{jT}	0.0	
Max. transaction fraction η_{jT}	0.1 (10% of C_T)	
Position Size Limits		
Min. position fraction ε_{jT}	0.0001	0.0001
Max. position fraction δ_{jT}	1.0	1.0
Cardinality limit k	—	$\{10, 20, 30, 40\}$
Solver		
Solver	Gurobi	
Time limit	3,000 seconds	
MIP gap tolerance	0.01%	

TABLE I: MILP Model Parameters — Non-Restrictive (NR-SF) and Restrictive (R-SF) Configurations

3.3 The Reinforcement Learning Formulation

In machine learning there are different approaches on how to train a model, the reinforcement learning is one, that instead of trying to predict the next immediate response tries to predict a sequence of actions that will lead to a better final outcome, this approach is particularly suitable to tasks that involve sequence of action-responses like games or dynamic control, for example.

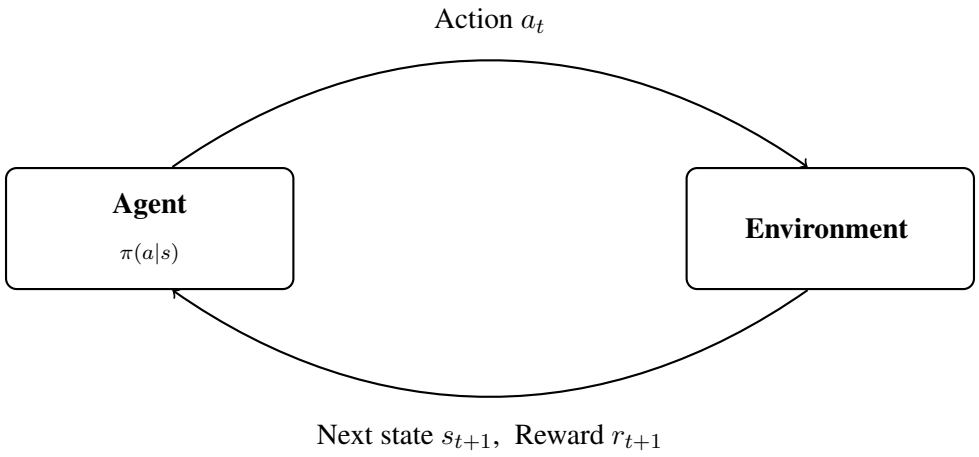

FIGURE 5: agent–environment interaction in reinforcement learning.

Figure 5 illustrates an agent reading a state, which is a representation of all possible information available to him at that point and deciding an action, this is done via a policy - π - which is fundamentally a mapping from a state to an action, this action then interacts with the environment and the machine receives feedback, called a reward.

Part of the challenge is how to allow the machine to explore enough of the environment collecting enough experiences and learning patterns from the data and as this training evolves also being capable of converge towards a good solution (reducing exploration and becoming deterministic).

The method chosen here is the Proximal-Policy Optimization (PPO) of Schulman et al. (2017). In simple terms, PPO is an implementation of the actor-critic framework, which consists in two entities working side-by-side: the actor (the above mentioned policy), as the decision maker, a function that maps states to actions and the critic, a function that tries to predict the value of this decisions.

While training, the policy run with two networks in parallel, the first one produces a mean signal μ while the second produces a σ (standard deviation) output, these two outputs are combined and sampled to characterize an action, this stochasticity is what allows the method to explore the possibilities and do different actions every time it sees

the same state, here defined in accordance to Peng et al. (2024):

$$a_t = clip(\mu_{\theta_1}(s_t) + \sigma_{\theta_2}(s_t) \odot z, -b, +b)\,, \tag{26}$$

where $z \sim \mathcal{N}(0, I)$, $\odot$ is the element-wise multiplication $(\sigma \odot z) = \sigma_j.z_j$ and $clip(x, -b, b)$ is a clipping (or a truncation) applied on x between $-b$ and b, each asset have its own noise pertubation. Also worth noting the clipping mechanisms, several of these mechanisms can be implemented in order to truncate outputs and provide stability to the methods. The portfolio weights are then obtained via the softmax transformation,

$$w_t = \text{softmax}(a_t) = \left(\frac{e^{a_{1t}}}{\sum_j e^{a_{jt}}}, \ldots, \frac{e^{a_{Nt}}}{\sum_j e^{a_{jt}}} \right)\,, \tag{27}$$

This transformation is used to represent actions in a probability space for a broad class of machine learning problems and particularly fit the necessities of a weight distribution. The softmax mapping ensures $\sum_i w_i = 1$ by construction and that each $w_i \in [0, 1]$. A limitation of the post softmax output is that it is not capable of allocating exactly zero weights to an asset, it does not impose a cardinality constraint.

The policy performance then can be measured as a value-based tracking error (V-TE)[6], a measure of deviation between the daily portfolio value per unit and the index value, simplifying the displayed on Peng et al. (2024),

$$\text{V-TE}^q_{(t_1,t_2)} = \left[\frac{1}{(t_2 - t_1)} \sum_{t=1}^{(t_2-t_1)} \left| \frac{V_t}{N_0} - I_t \right|^q \right]^{\frac{1}{q}}\,, \tag{28}$$

V_t is the portfolio value at time t, I_t is the index level at time t, and $N_0 = V_0/I_0$ is a normalization constant that anchors the portfolio value to the index scale at the start of the evaluation window, ensuring that deviations are measured in comparable units. With $q = 2$ used throughout this work. The reward is then scaled by a β factor,

$$r_t = -\beta \cdot \text{V-TE}^q_{(t,\,t+\Delta)}\,, \tag{29}$$

important to note that here the return is made negative by definition, considering that V-TE is a tracking deviation that needs to be minimized towards zero and that the PPO surrogate objective, to be introduced in Equation (33), is by convention maximized, this negative signal aligns both.

This cycle of action–rewards is repeated successively through a path and a collection

[6]This optimization objective should not be confused with the V-MAD to be presented in Section 3.4

of these rewards (Figure 6):

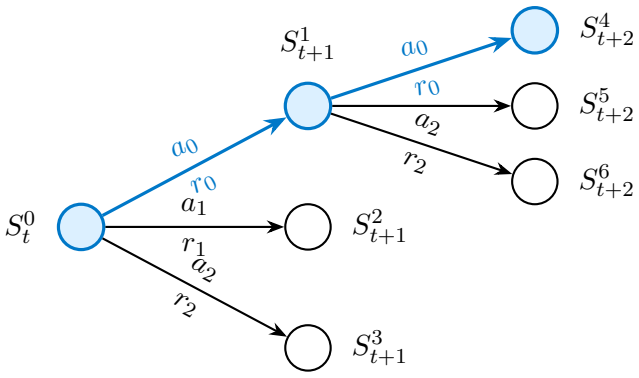

FIGURE 6: states, actions and rewards through a sampled path

This returns are then discounted by γ, as displayed on Equation (30), giving a notion of value for that given state, the role of the policy is to maximize this accumulated returns,

$$\max \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \cdot r_t\right] , \tag{30}$$

where $\gamma \in (0, 1]$ is a discount factor.

Searching through this multiple paths is a high variance task and among other things, different methods focus on how to stabilize the variance between this sampled path trajectories.

This observed value (Equation 30) then be contrasted with the predicted value made by the critic V_F, and this is the concept behind an advantage (Equation 31),

$$\hat{A}_t = -V(s_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t-1} r_{T-1} + \gamma^{T-t} V(s_T) , \tag{31}$$

The advantage represents the real value performed by the actor's choice (with its last signal bootstraped from the value function estimation) deduced from the value predicted by the critic and its idea is to provide a learning signal given the deviation from expected and observed values. The study uses then the concept of a Generalized advantage Estimation - GAE (Schulman et al. (2015)),

$$\hat{A}^{\text{GAE}}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l} , \tag{32}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the temporal difference (TD) residual at step t and

λ can be set to help control the bias-variance trade-off

The PPO surrogate objective then takes the shape of:

$$L_t^{\text{CLIP+VF+S}}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S[\pi_\theta]\left(s_t\right) \right] , \tag{33}$$

The first component L_t^{CLIP}:

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_{\theta_{\text{old}}}(s,a), \;\; \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}, 1-\varepsilon, 1+\varepsilon \right) A_{\theta_{\text{old}}}(s,a) \right) \right] \tag{34}$$

where $\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}$ is the ratio of probabilities under the updated and the previous policies and $A_{\theta_{\text{old}}}(s,a)$ the advantages. The loss of the value function - $L_t^{\text{VF}}(\theta)$ - is then measured by a simple squared deviation between expected and observed values,

$$L_t^{\text{VF}}(\theta) = \left(V_\theta(s_t) - V_t^{\text{targ}} \right)^2 , \tag{35}$$

Finally the Entropy, defined as

$$S[\pi_\theta](s_t) = -\sum_a \pi_\theta(a \mid s_t) \, \log \pi_\theta(a \mid s_t) \, , \tag{36}$$

During training, the collected episode buffer is divided into mini-batches and used to compute gradients of the composite loss, with advantage estimates normalized to zero mean and unit variance before each update. This mini-batch scheme is repeated for n_{ppo} cycles, updated jointly using the Adam optimizer (Kingma & Ba (2014)), the buffer is then discarded and new sampled episodes are collected in the next epoch, repeating the cycle. After training, the policy runs in deterministic mode - the σ and value network drop and only the mean signal $\mu_{\theta_1}(s_t)$ is used to produce the action.

Table II summarizes the set of hyperparameters discussed.

TABLE II: RL Model Parameters — Long-Term (LT) and Short-Term (ST) Configurations

Parameter	**LT**	**ST**
Data		
Lookback window M	1 year	30 days
Rebalance frequency	5 working days	
Rebalance steps per episode	4	
Network Architecture		
Policy hidden layers	128×8	64×4
Value hidden layers	128×6	64×2
Activation function	$\tanh(.)$	
Reward norm q	2	
PPO Hyperparameters		
Clip ϵ	0.2	
Discount factor γ	0.99	
GAE λ	0.95	
Policy learning rate	1×10^{-5}	
Value learning rate	1×10^{-5}	
SGD batch size $\tilde{n}$	200	
Optimization epochs per update	5	
Optimizer	Adam	
Value loss weight c_1	0.5	
Entropy weight c_2	0.005	
Reward scale β	10	
Episodes per epoch K	300	
Training cut-off (epochs) H	1,000	

on the hyperparameters, ϵ, γ and λ are chosen in accordance to standard PPO literature and the remaining parameters are design choices. The parameters $c2$, β, the learning rate and the number of portfolio rebalances must be adjusted jointly.

3.3.1 State Representation and Network Architecture

In this implementation, the state is simplified to include only the daily return history of the index and the N constituent stocks over a lookback window of M trading days, together with the current portfolio weights $w_t \in \mathbb{R}^N$,

$$s_t = \left[r^{\text{idx}}_{t-M:t},\ r^{\text{stocks}}_{t-M:t},\ w_t \right] \quad \in \mathbb{R}^{M(N+1)+N}, \tag{37}$$

where $r^{\text{idx}}_{t-M:t} \in \mathbb{R}^M$ are the daily index returns over the lookback window, $r^{\text{stocks}}_{t-M:t} \in \mathbb{R}^{M \times N}$ the corresponding stock returns, w_t the portfolio weights held entering period t. The idea of not using VIX, Stock volumes and T-Bill returns, for example, is deliberate: it ensures that any performance difference between the MILP and RL methods reflects the

dynamic versus static nature of the two approaches rather than differences in information availability.

The state vector s_t is processed by a Feedforward Neural Network (FNN) composed of L hidden layers, where each layer applies a linear combination of inputs followed by a $\tanh$ activation:

$$\begin{aligned} h^{(l+1)} &= \phi\big(W^{(l)}h^{(l)} + b^{(l)}\big)\,, \\ h^{(0)} &= \mathbf{x} \quad , \\ z &= h^{(L)} \end{aligned} \tag{38}$$

where $W^{(l)}$ and $b^{(l)}$ are the weight matrix and bias vector of layer l, and z is the output vector that determines the action, x is the feature[7] vector and ϕ the tanh.

The Recurrent Neural Network (RNN), exemplified in Figure 7 by a sequence of linked processing units, has structure developed to work with temporally or spatially organized data.

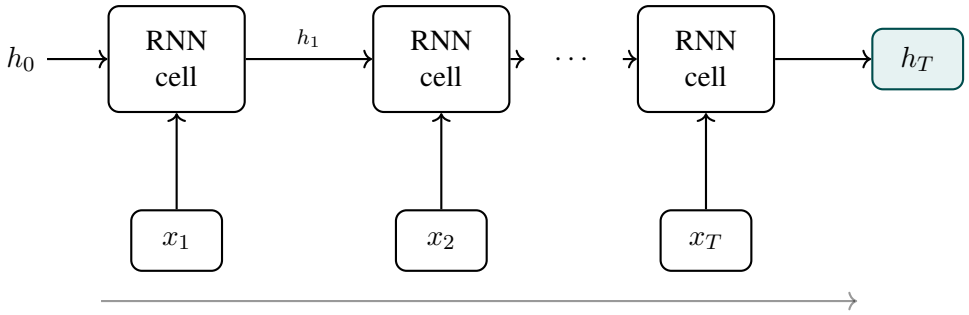

FIGURE 7: single-layer RNN over T time steps.

Instead of a flattened vector x comprised of a flattened stream of all daily data t of every asset j, this daily data is partitioned and each cell receives a daily input x_t and the previous hidden state h_{t-1} that represents the encoded information of previous days. The final hidden state h_T, defined on dimension 32, summarizes the encoded information of the sequence.

A specific type of processing cell is the Gated Recurrent Unit (GRU), introduced by Cho et al. (2014), in which the fundamental unit consists of two gates, a reset gate r_t (Equation (39)) and an update gate z_t (Equation (40)), both receives the current state of information at that cell x_t and the past information encoded until that moment h_{t-1}, but differ in their weigts W and U. The reset gate then controls how much of the previous state is passed forward, as displayed in Equation (41) by $r_t \odot h_{t-1}$ (where $\odot$ element-wise multiplication) controlling the balance between x_t and h_t. This post-reset output $\tilde{h}_t$ is

[7] the name 'feature' is often used to describe an input variable

then processed again by the update gate as shown in Equation (42), the encoded vector h_t then become the input for the next cell and the process repeats.

$$\begin{cases} r_t = \sigma(W_r x_t + U_r h_{t-1}) & (39) \\ z_t = \sigma(W_z x_t + U_z h_{t-1}) & (40) \\ \tilde{h}_t = \tanh(W x_t + U(r_t \odot h_{t-1})) & (41) \\ h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t & (42) \end{cases}$$

The final hidden state h_T (Figure 7) serves as an encoded (compressed) representation of the sequence, which is fed into the Feed-forward network.

3.4 Evaluation Metrics

Two metrics are adopted to evaluate the out-of-sample tracking performance of each strategy. The Return-based Tracking Error (R-TE),

$$\text{R-TE} = \sigma(r_{p,t} - r_{b,t}) \times \sqrt{252} \tag{43}$$

measures the annualised standard deviation of daily returns between the replicating portfolio and the benchmark index, where $r_{p,t} = V_t^p / V_{t-1}^p - 1$ and $r_{b,t} = V_t^b / V_{t-1}^b - 1$ are the daily returns of the portfolio and the benchmark index at time t, respectively, and $\sqrt{252}$ annualises the daily average.

The Value-based Mean Absolute Deviation (V-MAD),

$$\text{V-MAD} = \frac{1}{T} \sum_{t=1}^{T} \left| \frac{V_t^p}{V_1^p} - \frac{V_t^b}{V_1^b} \right| \tag{44}$$

introduced with alignment to the optimization objective by Strub & Baumann (2018), measures the average absolute distance between the normalized cumulative value paths of the portfolio and the index over the evaluation window $[1, T]$, where V_t^p and V_t^b denote the portfolio and index values at time t, and V_1^p, V_1^b are the respective values at the start of the window, both normalized to 1.0.

Unlike R-TE, V-MAD accumulates the divergences between portfolio and benchmark paths: a strategy will only report zero V-MAD if its value paths are the same, including the value consumed by transaction costs (TC). Considering that the optimization under both methodologies is value-focused, this is the most consistent metric to be applied.

4 RESULTS

4.1 Configuration Selection

This sub-section analyzes both methods individually in order to assess how they perform under different lookback and training windows and cardinality constraints (applicable to the MILP), This standalone analyze, where MILP is compared with MILP and RL with RL - is done with zero fixed costs, reducing computational costs and evaluating pure tracking capacity. It then evolves to the comparison of both.

The MILP analysis proceeds in two steps. In the first, the Non-Restrictive (NR) formulation is evaluated across five lookback windows (3 year, 2 year, 1 year, 180 days and 30 days) in order to identify the configuration that best balances tracking fidelity and sensitivity to the estimation window. Running the full grid of cardinality constraints across all three windows would be computationally prohibitive, so the Restrictive formulation is evaluated only under the window selected in this first step. In the second step, the NR and R-SF results are compared under that common window, motivating the choice of the NR configuration, carried forward into the comparison with the RL in Section 4.2.

Regarding the selection of a lookback window, Tables III and IV reports the R-TE and V-MAD for the Non-Restrictive formulation under the five lookback windows over the 2018-2025 evaluation period. The 1-year (252 trading days) window consistently achieves the best balance across both metrics. The 30-day window produces elevated V-MAD, thinking on this method as a path similarity fit, this can be seen as an insufficient window to allow precise weight allocations the 2 and 3-year window have a large index path to be match and allow low adaptation to transitions, degrading performance during structural breaks such as 2020. The performance does not degrade so drastically with window length, suggesting that the optimizer is not simply overfitting to longer historical periods but rather that medium-term windows offer a better balance between short-term movements and stability. The 1-year window is therefore adopted for all subsequent comparisons. The numbers also provide an interesting observation on how return track and value track differs, it is observable how the 30 days window displays relatively low R-TE, even though they go the V-MAD performance is the worst.

Regarding multiple cardinality constraints, Table (V) reports the out-of-sample R-TE and V-MAD for six replication scenarios solved under the Mixed Integer Linear Programming (MILP) framework - Full Replication (FR), the Non-Restrictive, and four cardinality-constrained cases ($k \in \{10, 20, 30, 40\}$)— evaluated annually from 2018 to 2025 with quarterly rebalances (63 days) and with a cut-off of 600 seconds on the computation of

TABLE III: MILP NR-SF Tracking Performance by Lookback Window (2018–2025): 3-year, 2-year, and 1-year windows

	3 years			2 years			1 year		
Year	R-TE	V-MAD	V-MAD^c	R-TE	V-MAD	V-MAD^c	R-TE	V-MAD	V-MAD^c
2018	0.20	0.06	0.06	0.15	0.09	0.09	0.10	0.06	0.06
2019	0.20	0.07	0.07	0.15	0.10	0.09	0.10	0.03	0.05
2020	0.35	0.44	0.19	0.27	0.30	0.16	0.20	0.18	0.09
2021	0.24	0.43	0.25	0.18	0.26	0.19	0.15	0.27	0.14
2022	0.25	0.41	0.28	0.20	0.41	0.23	0.14	0.26	0.16
2023	0.19	0.81	0.37	0.15	0.71	0.31	0.12	0.31	0.19
2024	0.18	1.47	0.53	0.15	1.32	0.46	0.13	0.65	0.25
2025	0.19	1.57	0.65	0.19	1.48	0.58	0.17	0.79	0.32
Mean	0.23	0.66	0.30	0.18	0.58	0.26	0.14	0.32	0.16
Std	0.05	0.55	0.20	0.04	0.50	0.17	0.03	0.24	0.09

R-TE: annualized return-based tracking error (σ of daily excess returns $\times\sqrt{252}$). V-MAD: mean absolute deviation of normalized cumulative value paths. V-MAD^c: Cumulative V-MAD. All values in percentages. Transaction costs: $c_j^b = c_j^s = 0.1\%$, $c_j^f = 0$.

TABLE IV: MILP NR-SF Tracking Performance by Lookback Window (2018–2025): 180-day and 30-day windows

	180 days			30 days		
Year	R-TE	V-MAD	V-MAD^c	R-TE	V-MAD	V-MAD^c
2018	0.09	0.03	0.03	1.18	0.54	0.54
2019	0.07	0.07	0.05	0.20	1.27	0.91
2020	0.16	0.13	0.08	0.15	1.85	1.22
2021	0.13	0.36	0.15	0.15	2.65	1.58
2022	0.10	0.37	0.19	0.13	3.03	1.87
2023	0.09	0.56	0.25	0.12	3.59	2.15
2024	0.13	1.01	0.36	0.17	5.23	2.59
2025	0.14	1.54	0.50	0.17	6.17	3.02
Mean	0.11	0.51	0.20	0.26	3.04	1.74
Std	0.03	0.49	0.15	0.31	1.88	0.82

R-TE: annualized return-based tracking error (σ of daily excess returns $\times\sqrt{252}$). V-MAD: mean absolute deviation of normalized cumulative value paths. V-MAD^c: Cumulative V-MAD. All values in percentages. Transaction costs: $c_j^b = c_j^s = 0.1\%$, $c_j^f = 0$.

each rebalance step. Several patterns emerge that motivate the design choices adopted in next session's main comparison. Also worth mentioning that the overperformance of the near full-replication strategies is expected once no fixed costs are being applied.

TABLE V: Tracking Performance Metrics (2018–2025)

	FR		**NR**		**k=10**		**k=20**		**k=30**		**k=40**	
Year	R-TE	V-MAD	R-TE	V-MAD	R-TE	V-MAD	R-TE	V-MAD	R-TE	V-MAD	R-TE	V-MAD
2018	0.05	0.02	0.12	0.06	3.04	3.99	2.26	0.68	0.97	0.34	0.50	0.29
2019	0.04	0.03	0.14	0.05	4.08	3.77	1.95	1.68	1.09	0.24	0.52	0.14
2020	0.09	0.19	0.33	0.14	4.86	6.70	2.68	4.70	1.74	0.82	0.90	0.49
2021	0.09	0.58	0.17	0.28	2.95	8.34	1.83	5.70	1.06	0.95	0.64	0.87
2022	0.07	0.75	0.24	0.47	3.86	7.38	1.89	3.12	1.17	1.93	0.69	1.49
2023	0.06	1.01	0.15	0.61	3.50	4.87	2.54	0.94	0.98	5.45	0.61	2.30
2024	0.07	1.74	0.15	1.18	4.12	8.52	1.51	4.65	0.96	11.33	0.48	5.31
2025	0.13	2.14	0.21	1.24	4.44	18.26	1.90	11.15	0.95	13.91	0.56	6.95
Mean	0.08	0.81	0.19	0.50	3.86	7.73	2.07	4.08	1.11	4.37	0.61	2.23
Std	0.03	0.79	0.07	0.48	0.66	4.64	0.39	3.41	0.26	5.40	0.14	2.54

All values in percentages. For transaction costs: c_j^b = (0.1%), c_j^s = (0.1%) and $c_j^f = 0$ FR: full replication; non-restrictive self-financed (NR-SF); k: cardinality constraint. Rebalanced quarterly with 600s of solver cut-off time per rebalance.

The relationship between tracking quality and additional assets is strong, moving from $k = 10$ to $k = 20$ reduces the mean R-TE from 3.86% to 2.07% and the equivalent step from $k = 30$ to $k = 40$ producea marginal gain of only 0.40%, from 1.11% to 0.61%. This concavity, also visible in the spread analysis of Figure 8, is consistent with the intuition that a small number of carefully selected constituents can span most of the explanatory capacity, while the remaining gain requires a large number of marginal positions whose individual contribution to replication is minimal. The Non-Restrictive also presents a long term track that overcomes the FR capacity, and it could be attributed to lower transaction costs that accumulate on the long run.

In contrast, V-MAD across cardinality levels reveals limitations of return-based diagnostics. At $k = 10$, R-TE remains in the range 2.95%–4.86% across the sample — already elevated, but interpretable as a moderate annualized deviation — whereas V-MAD rises from nearly 4.00% in 2018 to 18.26% by 2024. This is the error accumulation described in Equation (44): a systematic misallocation on the constituents and transaction costs accumulation, barely manifests at single-day deviations but become representative on the long run. It is also worth noting that the quality gain from $k = 30$ to $k = 40$ is more perceptible in V-MAD than in R-TE, indicating that smaller constituents can play a meaningful role in long-run tracking. This also provides an empirical motivation for adopting V-MAD as the primary metric for analysis.

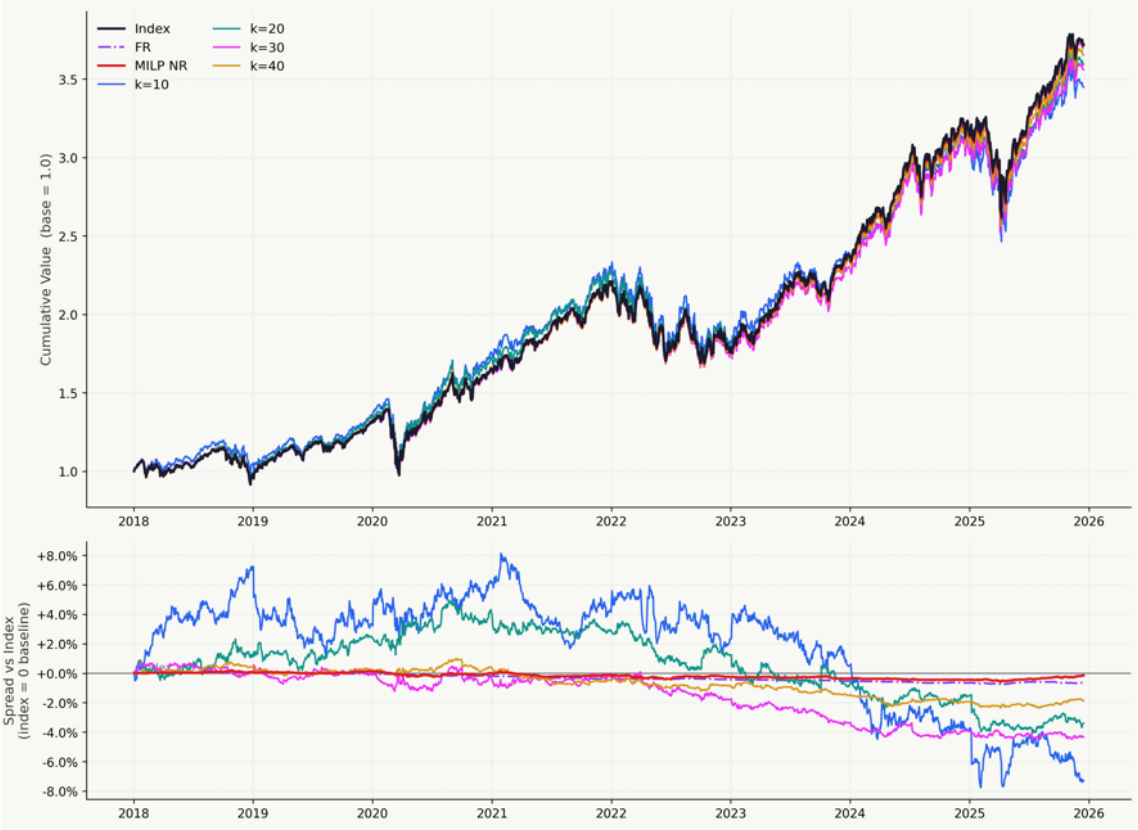

FIGURE 8: raw index view and spread-based view for multiple cardinality constraints. Quarterly rebalances and 600 seconds of cut-off time at the solver

An analogous evaluation is done to the RL method, Table VI presents the two training and lookback windows (Long-Term and Short-Term) and the third setup which makes use of the RNN. The RL-LT is by far the worst setup which among other things can be attributed to few training epochs for such a long training window and the incapacity of the model on capturing recent information from such a large time-span, despite some learning capacity can already be seen. Here the Equally-Weighted (EW) portfolio is compared as a benchmark for degeneration of the neural network, considering the year of 2020 for example where the EW portfolio underperform the index, a degenerated model incurring high transaction costs should only overperform that if it can display some learning capacity.

TABLE VI: Out-of-Sample Tracking Performance: RL Configurations vs. EW (2018–2025)

	EW			RL-LT			RL-ST			RL-ST-GRU		
Year	R-TE	TC	V-MAD	R-TE	TC	V-MAD	R-TE	TC	V-MAD	R-TE	TC	V-MAD
2018	3.94	0.20	2.11	3.41	1.89	2.59	2.96	1.69	2.82	1.84	0.20	0.55
2019	3.09	0.19	1.16	2.83	1.86	0.56	2.08	1.62	0.99	1.09	0.19	0.60
2020	11.20	0.24	7.04	11.01	2.09	5.70	8.41	1.71	2.89	5.22	0.24	1.54
2021	6.76	0.21	3.20	6.16	2.01	2.05	3.51	1.55	1.17	1.52	0.20	0.49
2022	7.24	0.23	4.99	6.75	2.11	2.98	3.31	1.49	2.15	2.07	0.22	0.27
2023	6.89	0.20	10.21	5.92	2.15	8.93	2.31	1.32	1.70	1.73	0.20	1.42
2024	8.97	0.21	6.88	7.56	2.06	6.75	4.57	1.50	3.09	4.55	0.21	4.51
2025	9.59	0.23	2.92	7.28	2.10	2.96	4.31	1.54	1.46	2.49	0.23	1.04
Mean	7.21	0.21	4.81	6.36	2.03	4.06	3.93	1.55	2.03	2.56	0.21	1.30
Std	2.64	0.02	3.06	2.55	0.10	2.79	1.97	0.13	0.84	1.49	0.02	1.37

EW: equally-weighted portfolio. RL-LT: RL with long-term windows. RL-ST: RL with short-term windows. RL-ST-GRU: RL with short-term windows coupled with a gated recurrent unit network. All values expressed as percentages.

4.2 Comparative performance through the years

TABLE VII: Out-of-Sample Tracking Performance: RL-ST-GRU vs. Benchmarks (2018–2025)

	FR			MILP_{NR}			RL-ST-GRU		
Year	R-TE	TC	V-MAD	R-TE	TC	V-MAD	R-TE	TC	V-MAD
2018	0.06	0.11	0.01	0.10	0.12	0.05	1.92	0.32	0.99
2019	0.04	0.11	0.03	0.11	0.12	0.05	1.02	0.29	1.04
2020	0.09	0.11	0.09	0.20	0.13	0.24	5.00	0.36	1.33
2021	0.09	0.11	0.13	0.15	0.13	0.12	1.42	0.31	0.50
2022	0.07	0.11	0.03	0.15	0.13	0.12	2.19	0.36	0.45
2023	0.06	0.11	0.05	0.13	0.12	0.05	1.54	0.31	1.40
2024	0.07	0.11	0.08	0.13	0.12	0.09	4.66	0.32	4.66
2025	0.13	0.11	0.04	0.17	0.13	0.10	2.68	0.34	1.48
Mean	0.08	0.11	0.06	0.14	0.13	0.10	2.55	0.33	1.48
Std	0.03	0.00	0.04	0.03	0.00	0.06	1.49	0.03	1.34

R-TE: annualized return-based tracking error (σ of daily excess returns $\times\sqrt{252}$); TC: total transaction costs over the evaluation year; V-MAD: mean absolute deviation of normalized cumulative value paths. All values expressed as percentages. For transaction costs: c_j^b = (0.1%), c_j^s = (0.1%) and c_j^f = $\$5,0$

Table VII summarizes the tracking performance across the years of 2018-2025. With the two strategies: MILP NR and RL-ST-GRU and the FR as benchmark. The FR strategy

consistently achieves the lowest tracking error in every year (its transaction costs are basically the costs of building the portfolio, with residual costs for rebalancing), confirming its role as the theoretical upper bound on replication quality for the non-restrictive setup.

The NR MILP closely follows FR across all metrics, with slightly higher transaction costs reflecting its more frequent rebalancing schedule, its V-MAD observations remains below 0.3% on an annualized basis. The RL-ST-GRU transaction costs are in a similar order of magnitude to the NR MILP ones, and the performance difference can be attributed to asset allocation itself. Figure 9 display the methods across each evaluated year starting from a baseline of 1.0

FIGURE 9: raw index and methods

The year of 2018 is a mild volatility year while 2019 is characterized by a sustained bull market, both falling within the pre-pandemic period of the experiment. The NR MILP tracks closely to the FR in both years, displaying strong replication performance that can be partly attributed to the relative stability of index constituents during this period. The RL policy underperforms not only due to accumulated transaction costs but mostly due to an inability to find allocations that are close to the FR solution, it usually find the top one or second best allocation but is uncapable of correctly ranking all the constituents according its true weights. in this case for example, it is not capable of finding the correct allocation specially at Amazon, where the algorithm allocates nearly 6% when the index have it at nearly 10%, visible in Figure 10 this struggle to correctly rank less representative constituents have an opposite effect in 2019, as the EW portfolio overcomes the index, the RL, more prone to the EW end up overcoming it too.

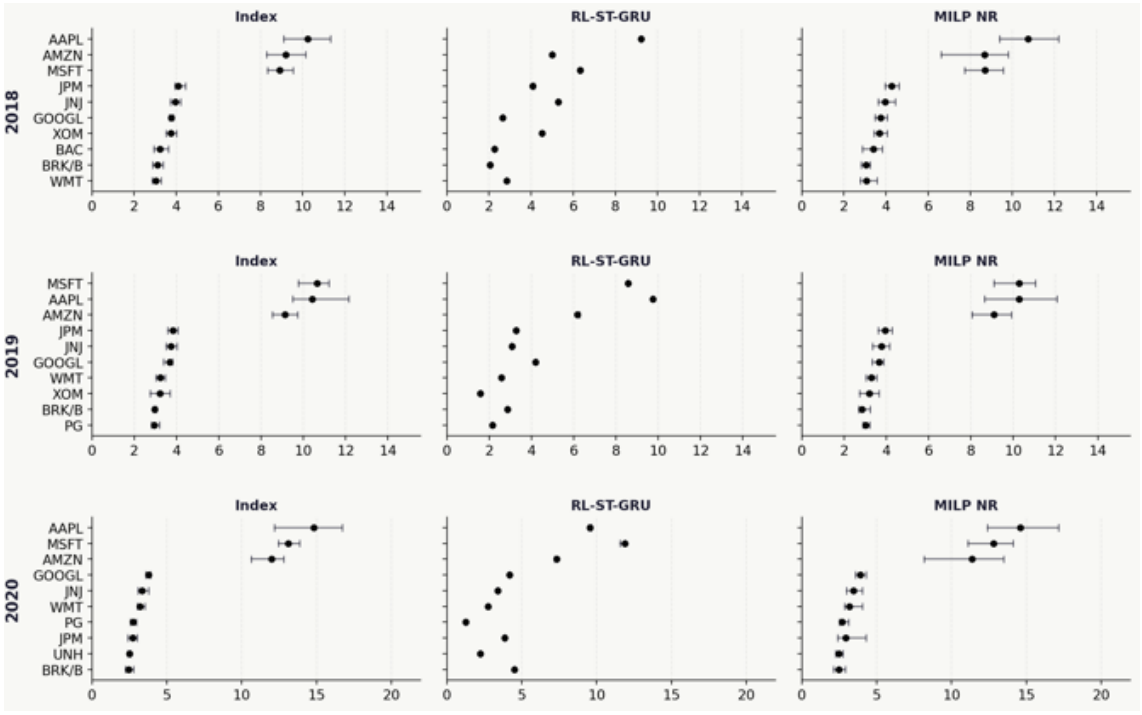

FIGURE 10: top 10 weight allocation per method, RL-ST-GRU and MILP NR. Years 2018 - 2020. x axis values in percentage

The year of 2020 is characterized by the pandemic, which produces the highest volatility across the serie as shown in Figure 3. The MILP react well to the shock, not displaying elevated transaction costs relative to other years and the steady transaction costs for this specific year indicates that the number of trades is constrained even with the shock. The 0.24% V-MAD, despite being the highest one for the MILP series, demonstrates the strength of the method.

At the same time, the RL policy displays limited capacity to track the index recovery following the crash. Being trained in the pre-pandemic world, it is incapable of finding the growth on tech drivers that drove the recovery. The year of 2021, can be seen as a continuation of this bull trend and so being trained with 2020 makes a good fit.

The years of 2022, 2023 and 2024 keep this trend of concentration in tech mega-cap, visible in Figure 11, where the RL, slightly more proned to the EW overperforms the index. Also, MILP second worst V-MAD without an increase in transaction costs can be seen as a product of high volatility even though the optimizer finds a solution avoiding large quantities of trading.

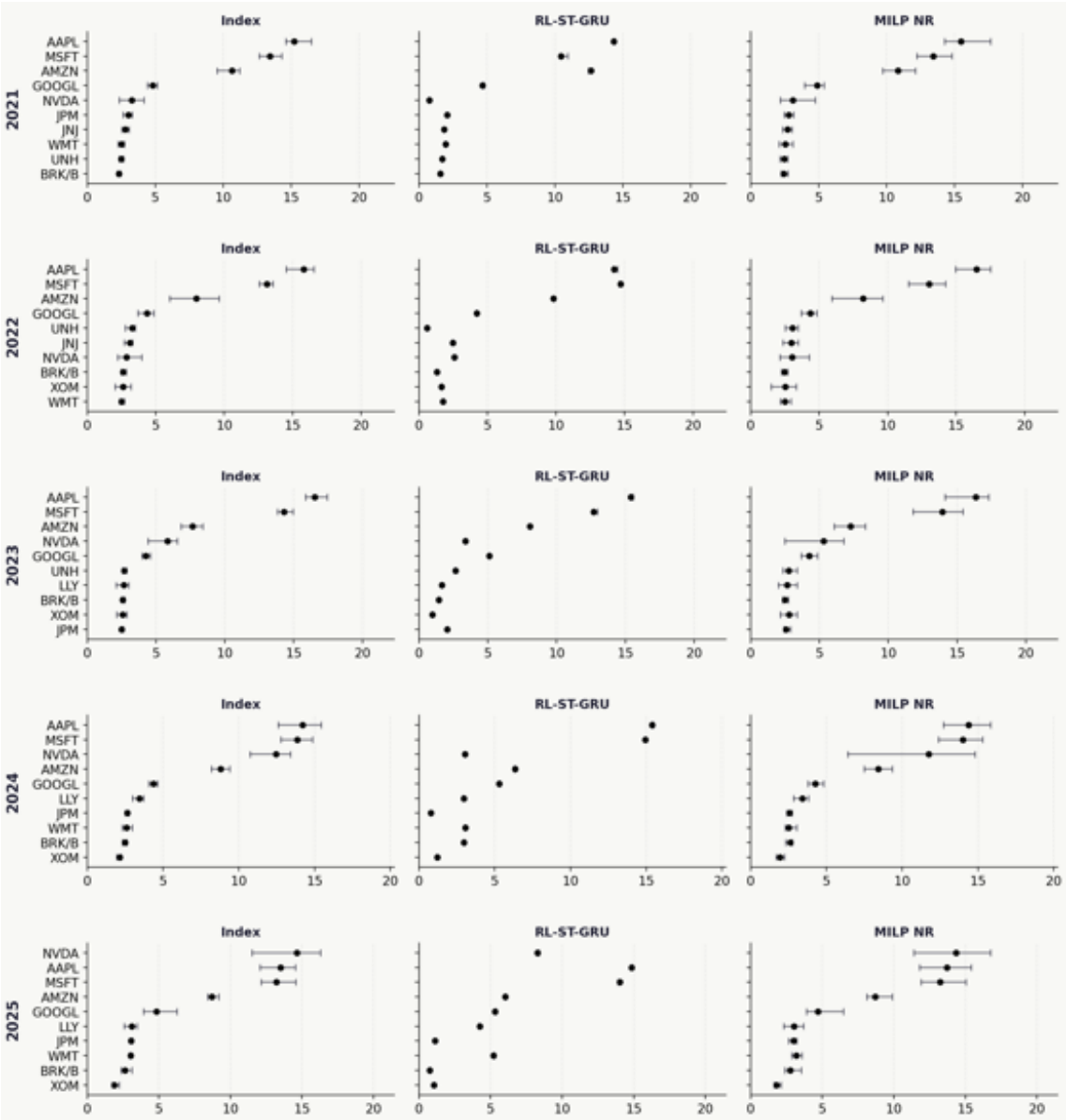

FIGURE 11: top 10 weight allocation per method, RL-ST-GRU and MILP NR. Years of 2021 - 2025. *x* axis values in percentage

The recovery from the 2022 drawdown becomes clear in 2023 and 2024, driven by the AI rally - with emphasis at the Nvidia growth in the index, this constituent grows nearly 7% in the index, and similar to what was observed on previous periods, a policy trained on historical data was incapable of adapting to the shift. And so the RL performance deteriorates reaching its worst levels across the full sample.

The year of 2025 combines both the April tariff shock, breaking the bull trend from the past two years, followed by a strong recovery. Differently from the 2020 shock, the MILP tend to hold the V-MAD at a 0.10% level, mostly due to correctly finding the top allocations. For the RL, the method heavily misallocate Nvidia, nearly 7% below the correct value.

5 CONCLUSION AND FURTHER IMPROVEMENTS

This study analyzes the problem of designing portfolios to track a given benchmark, a problem that belongs to a category of search tasks that are, most of the times, computationally prohibitive to be solved exactly and must be approached by searching algorithms capable of optimize it. Two methodologies are compared here, a deterministic optimization (MILP) and a reinforcement learning (RL), designed via Proximal Policy optimization (PPO), the portfolios obtained by each method are rebalanced on a weekly basis and tested on annual windows. Worth noting that the same data is available for both methodologies and the computational cost is also constrained, impacting mostly the RL, at nearly 90 minutes.

The MILP emerges as the dominant strategy throughout the full evaluation period as visible in Table VII, achieving a mean V-MAD of approximately 0.10% and consistently outperforming the RL policy on the reported metrics, while keeping transaction costs stable at approximately 0.13% annually showing that the method of historical similarities is a strong proxy for the tracking task. The reinforcement learning, despite reaching a transaction cost in the same order of magnitude as the MILP, demonstrates a lack of allocation capacity. This allocation is severely constrained by a structural limitation on the training epochs, which is done on purpose to match the same order of the MILP's computational cost.

On the MILP side, the lookback window analysis reveals a relationship between window length and tracking performance. The 1-year window shows a good balance between short and long-term data and deepening this analysis may enhance results.

Regarding the RL, the progression made to RL-ST and RL-ST-GRU, both due to narrowing the training and lookback windows and from architectural choices, such as using a RNN pre FNN produces a perceptible gain on the RL side: the GRU act as an encoder, compressing the state vector into a smaller hidden state, this bring statibility to what is fed into the subsequent neural network, the FNN, and helps stabilizing its output reducing transaction costs to the same order of magnitude as the MILP while improving tracking error.

Even so, the RL's tracking capacity remains constrained by two structural limitations. First, the number of training epochs is deliberately capped to match the MILP's computational cost, and the results suggest that convergence requires substantially more training. Second, there is a difficulty in capture within-year regime shifts, as visible in the 2024 Nvidia concentration episode, which indicate that a higher training frequency should be required.

A few directions are suggested for future works. The results display the MILP as superior in both runtime and tracking accuracy over for the benchmark replication problem, expanding the loookback window analysis can bring results even further.

The above mentioned difference also indicates that the RL solution makes a good candidate to be tested in dynamic decision-making tasks such as managing cash deposits and withdrawals. Also, there must be an entropy decay management in the RL training: the policy collapses to a deterministic mode too quickly, limiting exploration, and methods to control the exploration/exploitation of a solution are needed. Also, the decision to train the RL on price data alone (excluding cross-data signals, for example: VIX, volumes, and yield curves) represents a deliberate parity constraint with the MILP that simultaneously limits the network's ability to adapt to structural market changes, this richer data input along with other network architectures could improve the RL method.

REFERENCES

Alexander, C. & Dimitriu, A. (2005), 'Indexing and Statistical Arbitrage', *Journal of Portfolio Management* **31**(2), 50–63.

Beasley, J., Meade, N. & Chang, T.-J. (2003), 'An evolutionary heuristic for the index tracking problem', *European Journal of Operational Research* **148**(3), 621–643.

Benidis, K., Feng, Y. & Palomar, D. P. (2018), 'Sparse portfolios for high-dimensional financial index tracking', *IEEE Transactions on Signal Processing* **66**(1), 155–170.

Bynum, M. L., Hackebeil, G. A., Hart, W. E., Laird, C. D., Nicholson, B. L., Siirola, J. D., Watson, J.-P. & Woodruff, D. L. (2021), *Pyomo — Optimization Modeling in Python*, Vol. 67, Springer.

Canakgoz, N. & Beasley, J. (2009), 'Mixed-integer programming approaches for index tracking and enhanced indexation', *European Journal of Operational Research* **196**(1), 384–399.

Chen, Q.-a., Hu, Q., Yang, H. & Qi, K. (2022), 'A kind of new time-weighted nonnegative lasso index-tracking model and its application', *North American Journal of Economics and Finance* **59**, 101603.

Cho, K., Van Merriënboer, B., Gulçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014), Learning phrase representations using rnn encoder–decoder for statistical machine translation, *in* 'Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)', pp. 1724–1734.

Cornuejols, G. & Tütüncü, R. (2007), *Optimization Methods in Finance*, Cambridge University Press.

Dai, Z. & Li, L. (2024), 'Deep learning for enhanced index tracking', *Quantitative Finance* **24**(5), 569–591.

Guastaroba, G., Mansini, R. & Speranza, M. G. (2009*a*), 'Models and simulations for portfolio rebalancing', *Computational Economics* **33**(3), 237–262.

Guastaroba, G., Mansini, R. & Speranza, M. G. (2009*b*), 'On the effectiveness of scenario generation techniques in single-period portfolio optimization', *European Journal of Operational Research* **192**(2), 500–511.

Guastaroba, G. & Speranza, M. G. (2012), 'Kernel search: An application to the index tracking problem', *European Journal of Operational Research* **217**(1), 54–68.

Kingma, D. P. & Ba, J. (2014), Adam: A method for stochastic optimization, arXiv:1412.6980.

Markowitz, H. (1952), 'Portfolio selection', *Journal of Finance* **7**(1), 77–91.

Mezali, H. & Beasley, J. (2014), 'Index tracking with fixed and variable transaction costs', *Optimization Letters* **8**(1), 61–80.

Peng, X., Gong, C. & He, X. D. (2024), 'Reinforcement learning for financial index tracking', *arXiv preprint arXiv:2308.02820v2* .

Rockafellar, R. T., Uryasev, S. & Zabarankin, M. (2006), 'Generalized deviations in risk analysis', *Finance and Stochastics* **10**(1), 51–74.

Roll, R. (1992), 'A mean variance analysis of tracking error', *Journal of Portfolio Management* pp. 13–22.

Rudolf, M., Wolter, H.-J. & Zimmermann, H. (1999), 'A linear model for tracking error minimization', *Journal of Banking & Finance* **23**(1), 85–103.

Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. (2015), 'High-dimensional continuous control using generalized advantage estimation', *arXiv preprint arXiv:1506.02438* .

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. (2017), 'Proximal policy optimization algorithms', *arXiv preprint arXiv:1707.06347* .

Stoyanov, S. V., Rachev, S. T., Ortobelli, S. & Fabozzi, F. J. (2008), 'Relative deviation metrics and the problem of strategy replication', *Journal of Banking & Finance* **32**(2), 199–206.

Strub, O. & Baumann, P. (2018), 'Optimal construction and rebalancing of index-tracking portfolios', *European Journal of Operational Research* **264**(1), 370–387.

AI Disclosure Statement

During the preparation of this work, the author used language model tools to support code writing and debugging, and also with the research for references. The product of such tools were reviewed by the author, who is fully responsible for the text presented.