POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Hyperparameter Tuning for Pairs Trading: an Online Learning Approach

Tesi di Laurea Magistrale in
Computer Science and Engineering - Ingegneria Informatica

Nahuel Coliva, 10748675

Advisor:
Prof. Marcello Restelli

Co-advisors:
Luca Sabbioni
Pierre Liotet
Lorenzo Bisi

Academic year:
2021-2022

Abstract: In finance, pairs trading is a market strategy that profits from two correlated assets by betting on the mean-reversion of their spread. Thus, the best profits are made by an underpriced and an overpriced asset which revert to their correct price after a short time. Modern technologies allow pairs trading to look for profitable pairs through statistical tests on a custom universe of assets, rather than limiting the research to those that present industrial similarities. However, removing this limitation comes with an exponential increase in the number of potential candidates: in order to overcome the issue, Machine Learning techniques, such as Principal Component Analysis and clustering, are employed to reduce the pairs research to subsets of assets. In this thesis, we start by applying a state-of-the-art pairs selection strategy to the context of the S&P500 daily dataset. We test the procedure with the suggested hyperparameters and evaluate the correlation between a newly proposed measure of cluster consistency and the selected pairs. After that, we assess the sensitivity of the method to the hyperparameters' variation and identify a subset of highly influential ones. Finally, they are optimised through the application of Online Gradient Descent with Momentum: the method achieves good performance with respect to both the suggested static parameters and the average of the tested parametrizations in a transaction fees-free environment, while profits heavily decline when costs are taken into consideration.

Key-words: hyperparameter tuning, pairs trading, online learning, OGDM

1. Introduction

Pairs trading is a statistical arbitrage which tries to exploit market inefficiencies that lead to over/underpricing of assets. Specifically, two assets are selected if their movements are similar, with respect to some metric, during the *pair formation* period: in the following period, the *trading* one, the arbitrage waits for the two assets price difference to deviate from the expected trend. Once a profitable threshold is reached, the strategy prescribes to bet on the mean-reversion of the two assets spread, which is likely to happen due to their historical movement similarity, achieving returns. While, in the past, pairs were selected on industrial similarities (e.g. very similar product type, same industrial group, ecc.), modern technology allows for extensive search beyond these bounds, looking for empirical relationships not always easily explainable in economic terms. At the same time, however, enlarging the search horizons brings computational challenges, since the number of pairs to be checked for

similarities grows exponentially as the number of assets increases. Moreover, spurious relationships may arise, introducing noise and possibly reducing final returns.

In order to overcome these issues while keeping the benefits of a data oriented approach, [Sarmento and Horta, 2020] relied on the application of Principal Component Analysis (PCA) and clustering to identify suitable subsets of assets. Then, the pair selection procedure would be run considering only pairs of assets that belonged to the same subset. While the contributions of the authors were also on the trading period, we will focus on the selection step and optimise its hyperparameters: in fact, even though they are few, their optimal value may change significantly during time, following the market trends. In order to do so, we apply an online learning approach: in particular, we handle each parametrization as an asset (called *expert* for the rest of the document due to the similarities with the expert learning field) and optimise our budget distribution over them. Thus, we are in the context of the *online portfolio optimisation* framework: among the available algorithms, we will apply Online Gradient Descent with Momentum (OGDM, [Vittori et al., 2020]), not only because of its theoretical guarantees on both the regret bound and the computational complexity, but also because of its empirical results on dealing with transaction costs. The experiments are performed both with and without transaction costs, while also measuring the impact of the rebalancing interval.

The remainder of the document is structured as follows: in Section 2 we recall some mathematical background that will be used throughout the research, followed by Section 3, where we provide an in-depth description of the [Sarmento and Horta, 2020] methodology and elaborate on which hyperparameter is worth the optimisation. In Section 4 we present the details of the optimisation framework and experiment performed, while in Section 5 an overview of the related works can be found. For what concerns the experimental results, Section 6 reports the exploratory analysis of the dataset we are dealing with: we perform the pair selection step using the suggested hyperparameters of [Sarmento and Horta, 2020] and present a measure of cluster consistency. As a result, we find out that selected pairs are usually clustered together few times. Then, in Section 7, the performance of the OGDM optimisation is shown: it achieves good results without considering transaction costs, while the pairs trading itself seems to struggle when costs are factored in. Nevertheless, the configuration with the longest rebalancing interval, that is, the number of days between each OGDM budget distribution update, managed to actually profit from the strategy. Moreover, we analysed the underlying distribution of the budget, finding out that, although an overfitting-like phenomenon emerged, the algorithm managed to generalise well on unseen data. Finally, in Section 8 we draw the conclusions and present possible directions for future research.

The present work is based upon the results of the cooperation between Politecnico di Milano and MDOTM.

2. Preliminaries

In the following, we provide some definitions and notations that will be used throughout the document. We will start from some statistics that allow us to determine whether a price series is likely to be a *mean-reverting* one or not, that is, a time series where zero-crossings happen frequently. Later on, we will describe the essential elements of online portfolio optimisation and, finally, we will reserve Section 2.1 to give an overview of pairs trading.

General Statistics.

Definition 2.1. (Integration and Co-integration) *We will call the time series* $X_t = [x_1, \dots, x_T]$ integrated of order 1 *[Engle and Granger, 1987] if the process* $X_t - X_{t-1}$ *is stationary and we will denote it as* $X_t \sim I(1)$. *Moreover, given another time series* $Y_t \sim I(1)$, *we will call the pair* co-integrated *if there exists a coefficient* β *such that their difference* $U_t = Y_t - \beta X_t$ *is stationary, that is,* $U_t \sim I(0)$.

Notice that, when two time series are co-integrated, not only their spread will have constant mean, but it will also be a mean-reverting process. Another interesting property of mean-reverting processes is the Hurst exponent [Hurst, 1951], here reported in the generalized version (as suggested in [Barunik and Kristoufek, 2010]).

Definition 2.2. (Generalized Hurst Exponent) *Given a time series* $X_t = [x_1, \dots, x_T]$, *consider the statistic* $K_q(\tau)$ *defined as*

$$K_q(\tau) = \sum_{t=0}^{T-\tau} \frac{|x_{t+\tau} - x_t|^q}{(T - \tau + 1)}, \tag{1}$$

which can be interpreted as the q-order moment of the distribution of the increments with τ *lag ([Di Matteo et al., 2003]). We will call* generalized Hurst exponent $H(q)$ *the one which approximate* $K_q(\tau) \approx c\tau^{qH(q)}$ *the best. In particular, we will refer to* $H(2) = H$ *as the* Hurst exponent, *since its estimate is comparable to the original one of [Hurst, 1951].*

To further elaborate on $K_q(\tau)$, let us remark that such exponential approximation comes from [Barabási and Vicsek, 1991] and is related to a fractal interpretation of the time series X_t. In any case, in all the aforementioned researches it is taken for a known fact, thus we will follow the same line.

Finally, notice that this parameter gives insight on the evolution of the process: in fact, $H \in [0, 0.5)$ suggest a mean-reverting trend, while it is likely to be divergent when $H > 0.5$. Finally, values around 0.5 suggest a Brownian motion.

The last relevant parameter to assess the mean-reverting properties of a process is the mean-reversion half-life, described in the following Definition.

Definition 2.3. (Mean-Reversion Half-Life) *Given an Ornstein-Uhlenbeck process defined as*

$$dX_t = \alpha(\mu - X_t)dt + \sigma dW_t, \tag{2}$$

where $\alpha > 0$ and W_t is the Wiener process, we will call the time needed (on average) to halve the distance from its historical mean as mean-reversion half-life $t_{1/2}$. *Such a quantity can be obtained starting from the ordinary differential equation derived from the process equation $\dot{x} = \alpha(\mu - x)$, which admits*

$$x(t) = \mu + e^{-\alpha t}(x_0 - \mu) \tag{3}$$

as solution. With Equation 3 we can solve

$$x(t_{1/2}) - \mu = \frac{x_0 - \mu}{2} \tag{4}$$

obtaining $t_{1/2} = \frac{log2}{\alpha}$.

Online Portfolio Optimisation. The problem of online learning can be described as a sequential prediction problem [Cesa-Bianchi and Lugosi, 2006] where the learner tries to predict a value y_t from the observation of a previous sequence $[y_0, \dots, y_{t-1}]$. The online nature resides in the fact that the problem is solved as new information become available.

In particular, we will deal with *online portfolio optimisation (OPO)* where, given a set of N tradable assets (stocks, in our case, to which we will also refer as tickers), we will call the sequence $r_t = [r_{1,t}, \dots, r_{N,t}]$ *price relatives* at time t, where $r_{1,t} = \frac{p_{i,t+1}}{p_{i,t}}$ and $p_{i,t}$ is the price of asset i at time step t. Then, the objective of the OPO framework is to allocate its resources as the vector $x_t = [x_{1,t}, \dots, x_{N,t}]$, trying to maximise the total wealth obtained, called *cumulative wealth* and defined as

$$W_T(x_{1:T}, r_{1:T}) = \prod_{t=1}^{T} \langle x_t, r_t \rangle, \tag{5}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. In order to face a simpler problem, we can rewrite the objective function as a convex loss function: following [Vittori et al., 2020], the chosen one for our setting is

$$f_t(x_t) = -log(\langle x_t, r_t \rangle), \tag{6}$$

and our new objective will be its minimisation.
In order to evaluate the performance of the algorithm, its loss is compared to the one of the *best constantly rebalanced portfolio (BCRP)*: its distribution over the assets will be denoted as $x^*_{1:T}$, while the metric actually used is the *regret on wealth*, formally defined as

$$R_T = log(W_T(x^*_{1:T}, r_{1:T})) - log(W_T(x_{1:T}, r_{1:T})). \tag{7}$$

Moving to a real world scenario, let us acknowledge that allotting resources means buying and selling assets. Thus, in order to deal with transaction costs, we can introduce the *total regret* as

$$R^C_T = R_T + \gamma \sum_{t=1}^{T-1} ||x_{t+1} - x_t||, \tag{8}$$

where γ is the transaction rate, fixed throughout the investment horizon and gives the transaction costs when multiplied by the amount of money involved.

2.1. Pairs Trading

Pairs trading is a market neutral trading strategy, meaning that it can potentially profit from both upwards and downwards market trends. It does so with a two-steps procedure:

1. firstly, a pair of assets is selected, according to some criteria, so that they are strongly correlated during a window that is called *pair formation period*;
2. secondly, the pair is traded during the *pair trading period*, speculating on the fact that the price spread between the two assets will converge to its historical mean.

The technique is thus based on the assumption that the two assets in a selected pair are structurally related to each other: their difference in price will be stable over time and the profits will come from the exploitation of market inefficiencies.

Effectively, these inefficiencies come in the form of an overpriced asset and an underpriced asset: the strategy then prescribes to sell short the former and to buy the latter. Notice that, introducing the *spread process* as the time series given by the difference of the two assets, we can interpret these positions from a different point of view: we can either buy or sell short the spread, depending on whether it is under or above its historical mean. Thresholds are usually defined (statically or dynamically) in order to decide when to open and close the position (long-short or vice versa).

We can identify three main metrics to select related pairs, as reported in the survey [Krauss et al., 2015]:

1. the first and most studied is the average square distance between the two normalized price series $P_i := [p_{i,1}, \dots, p_{i,T}]$ and P_j (see e.g. [Gatev et al., 2006]), where $p_{i,t}$ is the price of asset $i \in I$ at time step $t \in [1, T]$. Thus, formally:
$$d_{i,j} = \frac{1}{T} \sum_{t=0}^{T} (p_{i,t} - p_{j,t})^2. \tag{9}$$
Even though this metric may be appealing due to its simplicity, its generated pairs may be not very profitable (e.g. a zero-spread pair);
2. Pearson correlation between returns has also been used, e.g. in [Chen et al., 2012], that is, for two asset series X, Y,
$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y}, \tag{10}$$
where $cov(X, Y)$ is the two assets covariance and σ_X the standard deviation of X. As reported in [Krauss et al., 2015], although it proved better final profit than [Gatev et al., 2006], the lack of theoretical guarantees that the spread will reverse to its mean is an issue, along with possible spurious correlations among selected pairs;
3. finally, the cointegration relationship (see Definition 2.1) has both econometric foundations (the spread is expected to reverse to the mean) and empirical good results, e.g. in [Huck and Afawubo, 2015]; in practice, a pair is selected only when its components are cointegrated and, usually, the cointegration coefficient β is used as the ratio for the quantities actually traded: this means that, for each dollar invested in the first asset, we invest β dollars in the second one.

In the rest of the document, we will focus on the pair selection phase, especially considering the cointegration approach, while static but significant thresholds are set for the trading one.

3. Sarmento&Horta Pipeline

Since our work is built upon the one from [Sarmento and Horta, 2020], we will now describe the steps of their framework. Later, in Sections 3.1 and 4, we will describe how our research integrates and optimises it.
The main idea of the authors was to deploy an unsupervised learning algorithm in order to reduce the computational effort of an exhaustive pairs search: by grouping the assets in clusters, the possible generated pairs drop considerably, with a largely effective reduction in the computational costs. Unfortunately, this comes with an issue: clustering techniques are well-known to struggle when applied to high dimensional data (namely, our assets price series).

Principal Component Analysis (PCA). To overcome the aforementioned issue, the authors applied PCA to the normalized return series $r_{i,t}$, that is:

$$r_{i,t} = \frac{p_{i,t} - p_{i,t-1}}{p_{i,t-1}}. \tag{11}$$

Using PCA, an orthonormal basis of the data is extracted, allowing us to rank the directions in order of explained variance. Then, by keeping only a subset of the most informative ones, we can perform a dimensionality reduction. In particular, the authors chose to keep the 5 most explicative features: thus, assuming that there are N assets with T prices each, the dataset got restricted from $N \times T$ to $N \times 5$ in their experiments, where T was approximately seventy-five thousands, that is, three years of intraday data with 5 minutes frequency.

The meaning of the extracted features is not easily explainable, since they are linear combinations of the original features. While in [Sarmento and Horta, 2020] we read: "[PCA is] an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables, the principal components. Each component can be seen as representing a risk factor.", an alternative possible explanation could be that they represent the general trend of the series, since PCA selects the directions along which there is the greatest amount of variance in the dataset.

Let us notice that the number of features was selected empirically in [Sarmento and Horta, 2020], quoting again: "We adopt 5 dimensions since we find adequate to settle the ETFs' representation in a lower dimension [w.r.t. 15 as suggested by [Berkhin, 2006]] provided that there is no evidence favouring higher dimensions".

Clustering: OPTICS. OPTICS [Ankerst et al., 1999] is a clustering algorithm which was created to fix some of the weaknesses of DBSCAN [Ester et al., 1996]: in particular, the latter searches the data space for areas where data points are more dense. In order to do so, it uses a concept of density, obtained from two parameters:

- *MinPts*, which is, roughly, the minimum number of points needed to define the neighbourhood of a point as "dense";
- ε which is the radius of the neighbourhood.

Indeed, when DBSCAN faces datasets with highly variable density, it may result in poor performance.
To overcome this issue, OPTICS dynamically selects ε inside the feature space, so that sparser regions have a broader concept of core points (that is, the main data points of a cluster). Thus, the only parameter left is *MinPts*: although the authors did not write it explicitly, we assumed $MinPts = 2$ in their research, since a figure of the paper reports a cluster composed of only two elements.

Once transformed with PCA, data are fed to OPTICS which, in turn, outputs the asset labels, including the possibility to mark as noise. Among the pairs within the same cluster, the candidates for trading are then selected according to a set of hand-crafted rules.

Selection Rules. To make sure that a pair equilibrium persists, the authors put together ideas from different researches. Thus, in order to be selected, each pair must comply to the following rules:

1. first of all, the pair should be cointegrated: the authors propose the Engle-Granger test [Engle and Granger, 1987] due to its simplicity;
2. since the mean-reversion property of the spread series is of crucial importance, the pair should also have a Hurst exponent $H < 0.5$;
3. mean-reversion half-life is also taken into consideration: since we would like to generate profit in a reasonable time lapse, pairs with mean-reversion half-life less than a day and more than a year are discarded;
4. finally, to assure that there are enough profit opportunities, the spread process of each pair should have crossed its mean at least 12 times each year in the pair formation period.

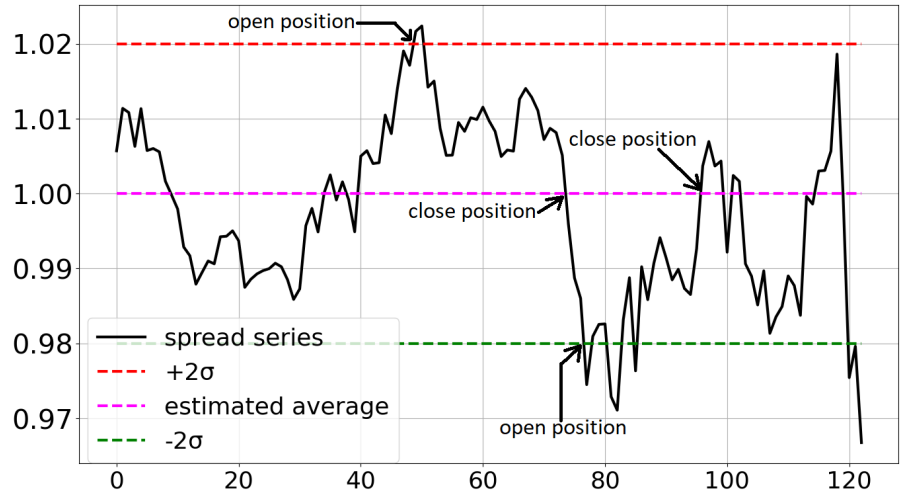


Figure 1: Visual representation of the trading strategy: when the spread series (black line) hits a distance of 2σ (red and green dotted lines) from the estimated average (magenta dotted line) a position is opened, that is, the overpriced asset is sold short while the underpriced one is bought; later on, once the spread mean-crosses, it is closed, generating profits.

Trading Strategy. During the pair formation step, the standard deviation of each pair spread process σ and its average μ are estimated. Then, in the trading period, such estimations are used to determine fixed thresholds: when the spread process gets to a distance of more than 2σ from μ, a position is opened, meaning that an equal amount of money is invested in the under and overpriced asset, buying the former while selling short the latter. Later on, when the spread crosses the mean, the position is closed. A visualization of the strategy is reported in Figure 1.

Lastly, let us remind that, in order to obtain strong statistical evidence for their findings, the framework was tested in three different periods (2012-2015, 2013-2016 and 2014-2017). Such periods were split as two year for pair formation and two for trading, where the first was for validation and the last one for testing.

3.1. Optimising the Pipeline

As explained in the previous sections, the framework from [Sarmento and Horta, 2020] has several hyperparameters that could possibly be tweaked to achieve better performance. In particular, the list boils down to:

1. length of the pair formation period;
2. number of features selected by PCA;
3. MinPts OPTICS parameter;
4. p-value threshold for the Engle-Granger test;
5. Hurst exponent upper bound;
6. mean-reversion half-life bounds;
7. minimum number of average mean-crossing per year.

To understand whether the impact of such hyperparameters were worth the optimisation, we performed a grid search exploration, paying attention to the sensitivity of an objective function. In the following, the main steps are reported:

- for each parametrization x in the grid, the *feedback function* $f(\boldsymbol{x})$ is defined as the mean daily return of the selected pairs over a two year rolling window, that is

$$f(\boldsymbol{x}) = \frac{1}{720} \sum_{t=1}^{720} \frac{daily\_return_t}{n\_selected\_pairs_t}; \tag{12}$$

- in order to consider only the most promising parametrizations, the analysis is then reduced on

$$\bar{X} = \{\boldsymbol{x} \in X | f(\boldsymbol{x}) > \frac{f_{max} + f_{min}}{2}\}, \tag{13}$$

  where f_{max} and f_{min} are the maximum and minimum, respectively, of function $f(\boldsymbol{x})$;

- as far as one dimension of $\boldsymbol{x}$, said h, is considered (i.e. a single hyperparameter), the goal is to assess how narrow is the range in which the best feedback values are found: hence, for each tuple of values (h_1, h_2) available in the grid, we define the *range feedback* $g(h_1, h_2)$ as

$$g(h_1, h_2) = \mathbb{E}[f(\boldsymbol{x}) | \boldsymbol{x} \in \bar{X}, \boldsymbol{x}^h \in [h_1, h_2]], \tag{14}$$

  where $\boldsymbol{x}^h$ is the value of hyperparameter h in parametrization $\boldsymbol{x}$. $g(h_1, h_2)$ represents the average of function $f(\boldsymbol{x})$ among parametrizations that have $\boldsymbol{x}^h$ in the considered range.

By plotting the range $[h_1, h_2]$ that attains the maximum over $g(\cdot, \cdot)$ and studying how it varies, we can get a deeper understanding of how much tuning the parameter may influence the final result: narrower intervals suggest high sensitivity (selecting the correct value is crucial to achieve almost-optimal performance), while, in turn, wider ones suggest low sensitivity (being in a neighbourhood of the optimal value is already enough). In Table 1 we report the whole grid.

Hyperparameter name	Values
formation period length (in days)	360, 400, 450, 500, 550, 600, 700, 800, 1080
PCA selected features	1, 2, 5, 8, 10, 14, 18, 22, 50
OPTICS MinPts	2, 4, 6, 8, 10, 12, 16, 20
p-value threshold	0.01, 0.03, 0.04, 0.05, 0.08
Hurst exponent upper bound	0.1, 0.3, 0.4, 0.5
mean-reversion half-life lower bound	1, 10, 100
mean-reversion half-life upper bound	20, 100, 365, 1000
average mean-crossing per year lower bound	5, 12, 24, 100

Table 1: Initial hyperparameters optimisation grid.

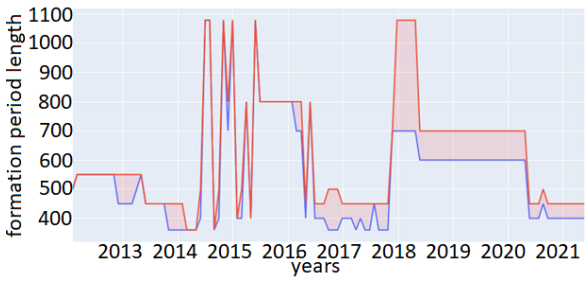
The analysis was performed on S&P 500 daily data from 1st January 2010 to 12th June 2021 and in Section 6 you can find an in-depth analysis of the clusters identified during the pair formation step with the parametrization suggested in [Sarmento and Horta, 2020].
As we can see from Figure 2, we can divide the hyperparameters into two groups:

- the first one, including the first four hyperparameters, is quite promising; in fact, optimal areas are usually narrow and, moreover, their included values change over the considered period;
- on the other hand, the second group, including the last four hyperparameters, is characterized by wide areas, generally the whole set of tested values, suggesting that the exact value is of little importance to the overall final result.

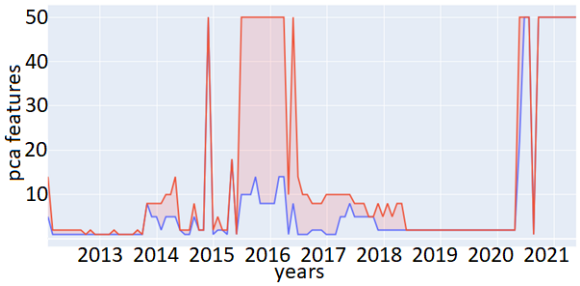
Therefore, the hyperparameters grid that will be considered during the rest of the document is the one presented in Table 2.

Hyperparameter name	Values
formation period length (in days)	360, 400, 500, 600, 700, 800, 1080
PCA selected features	1, 2, 4, 10, 14, 20
OPTICS MinPts	2, 4, 6, 8, 10, 12, 16, 20
p-value threshold	0.03, 0.05, 0.08

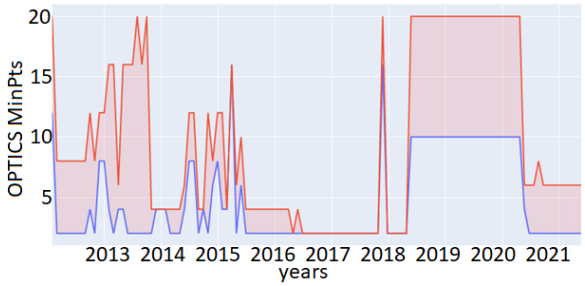
Table 2: Final hyperparameters optimisation grid.



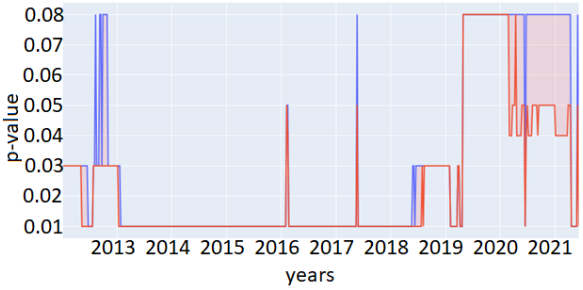
(a) Formation period length.



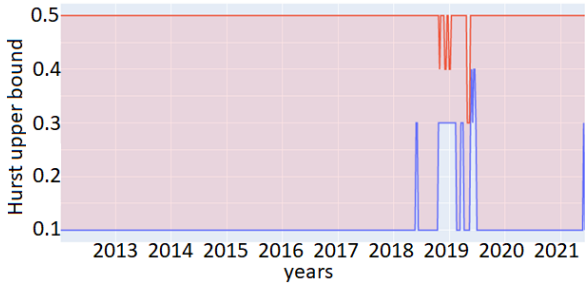
(b) PCA features.



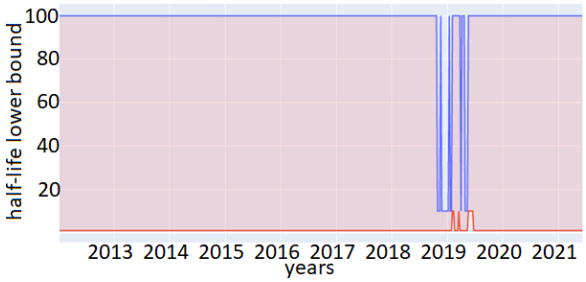
(c) OPTICS MinPts.



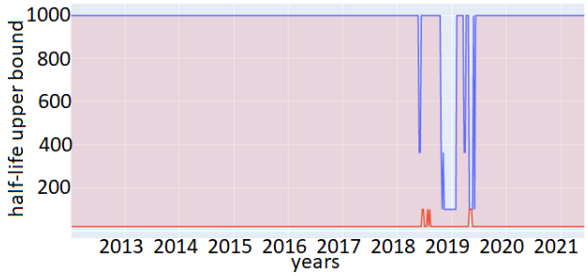
(d) p-value threshold.



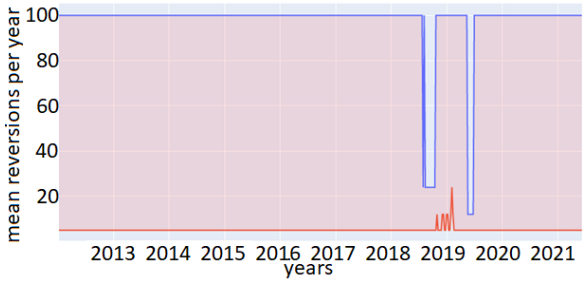
(e) Hurst exponent upper bound.



(f) Mean-reversion half-life lower bound.



(g) Mean-reversion half-life upper bound.



(h) Average mean-crossing per year lower bound.

Figure 2: Hyperparameters optimal areas: the blue lines represent h_1 while the red ones h_2; the area between the two is also in red.

4. Online Optimisation Framework

In the following Section we present the chosen framework, namely Online Portfolio Optimisation, starting from the reasons that lead us to adopt it. Later on, in Section 4.1, we introduce the chosen algorithm, that is, Online Gradient Descent with Momentum, while in Section 4.2 the trading strategy is reported. Finally, in Section 4.3, we describe the experiments performed, along with providing some last technical details.

As already stated in Section 1, our approach is in an online fashion: as the trading strategy unfolds, we would like to move our budget in order to maximise the final revenue or, alternatively, to reach the lowest possible loss with respect to some form of oracle. Such a problem seems to translate easily to the expert learning setting (see [Cesa-Bianchi and Lugosi, 2006] for a complete introduction to the problem): for each parametrization (i.e. each expert), we have perfect information on both the actions performed (i.e. the underlying hyperparameters combination) and the performance obtained (i.e. the returns obtained by the pairs selected). Therefore, given such information, expert learning algorithms try to minimise a *regret*, that is, some measure of the performance gap between an oracle (e.g. using look-forward bias) and the algorithm itself. The performance measure, in this case, would be the wealth obtained: by dynamically deciding the best allocation of its budget over the experts or, alternatively, by sampling, from such a distribution, the expert to follow at each time step, the algorithm tries to keep a low regret. Usually, the budget is initially uniformly distributed among the available experts: then, at each update, resources are moved from an expert to another, following those that performed better than others. Unfortunately, the problem is more convoluted than it seems: in order to assess the performance of the various configurations, we would have to wait for the trading period to end, then perform the budget distribution update and so on. Assuming that we stick to the one year testing trading period, the dynamics of the underlying system would be way faster than our capability to adapt: in other words, the optimal distribution would change too rapidly with respect to our feedback from the market.

Two different possible solution approaches were identified but, unfortunately, resulting in a trade-off:

- increasing the decision making frequency, with the drawback of introducing correlation among samples (due to the overlapping trading periods);
- reducing the trading period, thus avoiding the correlation among samples, but with the drawback of modifying the trading strategy, thus evaluating the hyperparameters on a different strategy.

In order to both increase the feedback frequency and keep the trading period of optimal length, we would like to be able to receive feedback as the positions are opened/closed. In this way, we are both increasing the feedback frequency and keeping the pairs in a trading phase for as long as they should be. However, such an idea comes with an obstacle: we have to keep track of the pairs selected by each expert, since they will be inherited by the same expert in the next time step. Unfortunately, such a solution would bound together the performance of an expert to its choices in previous time steps, which is outside of the expert learning setting, where prediction and evaluation are somewhat independent.

A workaround to stick to an online setting without introducing new features (that is, the information about "which pairs are selected by which expert"), that would be similar to a meta-reinforcement learning setting, is the online portfolio optimisation framework: in this case, we represent each expert (i.e. each parametrization of the underlying pair selection strategy) as an asset, with its price variation due to the performance achieved by the trading strategy. Although the complete explanation of the experts portfolio management is reported in Section 4.2, we will now address its main aspects in this new framework, along with their practical consequences:

- experts performances are bounded to their past decisions: in fact, each pair is kept in an expert portfolio for six months (save for some cases discussed in Section 4.2), while the interval between budget distribution updates will be generally lower (see Section 4.3);
- transaction costs play a crucial role: they are money to be paid in order to open/close a position and determine whether we need to acknowledge the dependency on past decisions or not; in fact, in a scenario where they are negligible, agents are free to move any amount of budget to the most promising assets, hence the gain of each expert is not depending on previous choices; on the other hand, when they need to pay fees to move previously invested money, a dependency is introduced;
- furthermore, let us remark that transaction fees are related to both our management of the budget distribution among experts (moving money from an expert to another one may result in buying/selling assets), as well as the decisions of the expert itself (where to invest money and when); while the impact of the former can be smoothed by some internal computation (see equation 17), the latter is what truly bounds the performance of an expert to its previous choices.

Having considered these facts, we are not really transitioning to a stateless setting, but rather limiting the impact of the state while keeping the representation simple: indeed, the OPO framework is substantially simpler than a reinforcement learning setting, both from a computational and design standpoint.

Lastly, the new objective is to find the best allocation for our budget to the various assets. Notice that this framework allows us to:

- have a dynamic idea of how well a strategy is performing, since we provide feedback at a higher frequency, but without correlation among samples and without truncating the trading period in an artificial way;
- explicitly model the cost of transactions, which is a key quantity in a strategy that is often rebalancing its budget distribution, especially considering our complex scenario, where rebalancing our portfolio could mean closing part of a position way too early, potentially even losing money on top of the transaction costs.

4.1. Dealing with Transaction Costs: OGDM

In order to keep the transaction costs low, we chose Online Gradient Descent with Momentum (OGDM, [Vittori et al., 2020]) as the optimisation algorithm, which provides also some interesting properties. In particular:

- it provides good guarantees on the regret w.r.t. the BCRP, both theoretically and empirically; specifically, it keeps R_T and R_T^C in the order of $\mathcal{O}(\sqrt{T})$;
- it is computationally efficient, scaling well both on the time horizon and on the size of the portfolio, with a per-iteration complexity of $\Theta(N)$ number of assets; notice that the latter will be especially important, since the grid grows exponentially in the number of hyperparameters considered;
- even though transaction costs are not specifically embedded in the algorithm, it manages to keep them low.

Moreover, the only assumption needed by the algorithm is the following basic one:

- there exist two finite constants ϵ_l, $\epsilon_u \in \mathcal{R}^+$ such that $r_{i,t} \in [\epsilon_l, \epsilon_u]$ with $0 < \epsilon_l \leq \epsilon_u < +\infty$, $\forall i \in [1, N]$, $\forall t \in [1, T]$.

The algorithm is described in the following:

Algorithm 1 OGDM(H,Λ)

input: learning rate sequence $H = \{\eta_1, \dots, \eta_T\}$, momentum parameter sequence $\Lambda = \{\lambda_1, \dots, \lambda_T\}$
Initialize $x_0 \leftarrow (0, \dots, 0)$ and $x_1 \leftarrow (\frac{1}{N}, \dots, \frac{1}{N})$
for $t \in 1, \dots, T$ **do**
  Receive r_t from the market
  Store the obtained wealth $\langle x_t, r_t \rangle$
  Compute the new budget distribution: $x_{t+1} \leftarrow \prod_{\Delta_{N-1}} \left(x_t + \eta_t \frac{r_t}{\langle r_t, x_t \rangle} - \frac{\lambda_t}{2}(x_t - x_{t-1}) \right)$
  for $i \in 1, \dots, N$ **do**
    Update the pairs portfolio $\mathcal{K}_{i,t+1}$ of expert i
    Distribute budget $x_{i,t+1}$ evenly among the pairs belonging to $\mathcal{K}_{i,t+1}$
  end for
end for

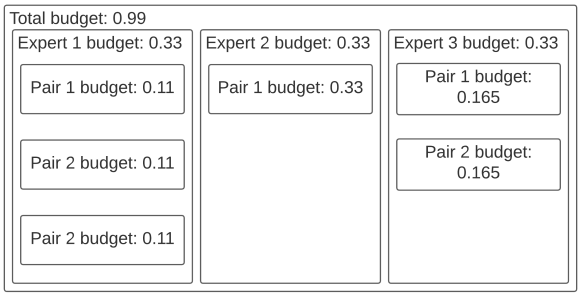
Let us notice that $\prod_X(y) = \underset{x \in X}{\arg\inf} \, ||y - x||_2^2$ is the standard projection of the vector y onto X, which can be computed in linear time as in [Duchi et al., 2008]. Finally, hyperparameter sequences H, Λ will be constant and optimized through a validation period: see Section 4.3 for more details.
Indeed, the trickiest part of the algorithm is the computation of r_t, which is reported next:

$$r_{i,t} = 1 + \frac{1}{K_{i,t}} \sum_{k \in \mathcal{K}_{i,t}} g_{k,t}, \tag{15}$$

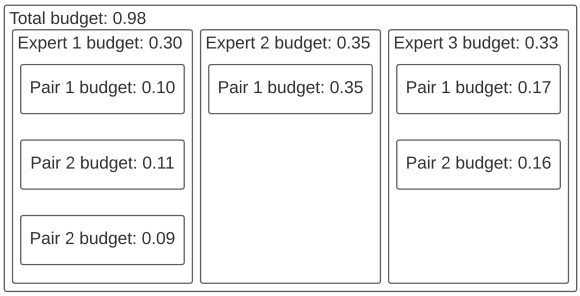
where $\mathcal{K}_{i,t}$ is the set of the $K_{i,t}$ tradable pairs for expert i at time step t and $g_{k,t}$ is the percentage gain on pair k at time step t; hence, $g_{k,t} = 0$ if the amount of money did not change once the position was closed or the pair was not opened in the first place.

Notice that we do not force any position to close before rebalancing, although we do compute $g_{k,t}$ with the actual available money (thus, in a certain sense, as we closed the position just before rebalancing): in this way, we stick to the trading strategy of each expert while providing a feedback to OGDM. Anyway, after obtaining x_{t+1}, it is possible that portions of open positions will be forced to be closed, due to the budget movement. Another thing to notice is that each expert i will distribute its budget $x_{i,t}$ uniformly over its pairs: this procedure is performed at rebalance time and may be needed even when $x_{i,t} = x_{i,t+1}$. Finally, in order to keep the computation

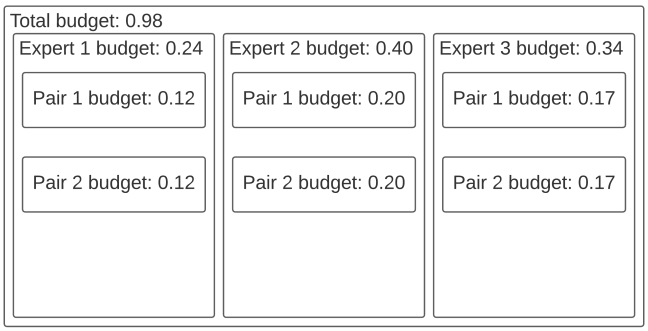
straightforward, the update of the budget distribution provided by OGDM is synchronized with the update of the pairs in each expert portfolio. An explicative example of update is provided in Figure 3.



(a) Starting budget distribution.



(b) Budget distribution after market changes and just before the OGDM update.



(c) Budget distribution after the update.

Figure 3: OGDM budget update example: starting from Figure 3a, where the budget is uniformly distributed among the experts and pairs, the open pairs are subject to the market changes. Then, in Figure 3b, we can see the budget of each expert and pair just before computing the update: in particular, expert 2 is gaining, while expert 1 is losing and expert 3 remains stable. Finally, in Figure 3c, the update is performed: the budget distribution over experts is computed as prescribed by OGDM and each expert homogeneously spreads it among its newly computed pairs.

It is also worth mentioning that the transaction computation will be slightly different from [Vittori et al., 2020]; namely, here we perform a transaction only if:

- a position is opened/closed;
- the OGDM update involves moving money from or to an expert which has still open positions: in fact, upon receiving a budget increase, the amount received is immediately invested in open positions; similarly, it may happen that experts that see their budget decreased have to partially close open positions.

In particular, performing a budget update does not necessarily trigger a transaction. Thus, for the aforementioned reasons, the explicit formula to obtain transaction costs C is:

$$C = \gamma \sum_{t=1}^{T} \left(\sum_{i=1}^{N} \sum_{k \in (\mathcal{O}_{i,t}^{-} \cup \mathcal{O}_{i,t}^{+})} ||x_{i,k,t} - x_{i,k,t-1}|| \right), \tag{16}$$

where $\mathcal{O}_{i,t}^{-}$ is the set of open pairs of expert i that receive a budget decrease from the update, similarly $\mathcal{O}_{i,t}^{+}$ but for budget increment. Lastly, notice that, with some internal computation, we can obtain an asset-level transaction costs formula, where the portfolio of the ensemble of experts is seen as a whole, moving the focus from the costs payed by each expert to the costs payed for each asset:

$$C = \gamma \sum_{t=1}^{T} \left(\sum_{j \in A} \left(\left| \sum_{i=1}^{N} \delta_{i,j,t}^{+} \right| + \left| \sum_{i=1}^{N} \delta_{i,j,t}^{-} \right| \right) \right), \tag{17}$$

where A is the set of tradable assets, $\delta_{i,j,t}^{+}$ is the variation of *invested* budget on a long position of asset j by expert i and similarly $\delta_{i,j,t}^{-}$ for a short position.

4.2. Pairs Portfolio Management and Trading Strategy

After having described what the pairs selection phase looks like, we now give the main details of the trading period, that is, both how to deal with the pairs selected and how to trade them.

Pairs Portfolio Management. Each expert has its own portfolio of tradable pairs obtained incrementally from the various pair selection steps. Notice that, in contrast with most of the literature, here we overlap pairs selection and trading periods: in fact, we do not alternate the two steps, but operate in a more continuous way, by updating partially but frequently the tradable pairs. As upside, we make the feedback loop tighter, boosting the performance of OGDM. However, such an overlap comes with a cost: we have to manage how pairs are inserted in/dropped from the portfolio of each expert.

Specifically, the common policy is given in the following:

- at each rebalancing step, the pairs selection algorithm is run, but only 5 pairs can be added to the expert portfolio: namely, the top 5 pairs in order of cointegration p-value significance (the lower, the better);
- the maximum portfolio size is 30: if the new pair selection step obtains different pairs, we need a policy to deal with it, that is, how and which pair to be added/dropped in order to keep the 30 pairs limit;
- we will call the number of pairs that are dropped in favor of new pairs *turnover* and will cap it at 5;
- pairs are dropped in order of returns: thus, in the case of a complete turnover, the dropped pairs would be those that have generated the lowest return.

Let us notice that, even though each expert has a uniform budget distribution over its pairs, the overall distribution among the underlying pairs will not be so, for two main reasons:

1. the budget associated to each pair is proportional to the number of experts that selected it: although this phenomenon is more evident at the beginning, it will still influence the overall return until the end;
2. OGDM updates tend to put money to experts that have positive returns, but since a pair can be selected by multiple experts, the amount put on that pair is actually amplified.

The overall wealth of the algorithm is thus determined by both factors and can be interpreted as an overall pairs portfolio.

Trading Strategy. In order to assess the chosen pairs, we need to implement a trading strategy: the chosen one is an adaptation from [Caldeira et al., 2013]. Although very simple, it was proven to be effective, thus a good choice in our setting, where the focus is on the pairs selection.

Its main steps are:

1. for each pair, extract the spread time series of the pairs selection period, that is

$$S_t = X_t - \beta Y_t, \tag{18}$$

   where X_t and Y_t are the two assets series, while β is the cointegration coefficient;
2. compute its mean μ and its standard deviation σ;
3. during the trading period, monitor the value of the spread s_t: a position is opened (investing an equal amount of money on the two assets) when $|s_t - \mu| > 2\sigma$, while it is closed when $|s_t - \mu| < \frac{\sigma}{2}$.

Additionally, the strategy employs a stop-loss threshold: when a pair return goes below -7%, it is closed and dropped from the expert portfolio.

4.3. Experimental Framework Wrap-up

We will now summarise the main features of the framework adopted, along with some useful details:

1. about the generation of the experts: due to computational issues, not all the possible combinations of hyperparameters have been considered (for the explicit list, see Appendix A); nonetheless, 428/1008 experts already give an idea of the range of optimisation: besides, experts which differ only by a parameter are likely to be highly correlated, providing little additional information;
2. for each expert, the pair selection step is the one from [Sarmento and Horta, 2020] with the expert own parametrization, while the trading strategy is common to all of them, as it is described in the previous Section;
3. once defined the pairs selection and trading strategy, which contribute not only in generating the price relatives r_t but also in defining the budget assigned to each pair, OGDM is run;

4. since the time interval between OGDM updates influences not only the frequency of the OGDM feedback but the whole problem too (e.g. pairs selected and their returns), we decided to test different values for it; specifically, the *rebalancing intervals*, in days, were selected in the set $\{7, 14, 21, 42, 63\}$, which roughly translate to a week, three weeks, a month, two months, three months respectively (due to the absence of Saturdays and Sundays in the dataset).

One of the baselines that will be used to assess the performance of OGDM is the one of *independent experts*: it is obtained by splitting the initial budget to each expert as in OGDM, but here they trade without the possibility to move money from one to another (thus, the independence). Given this definition, the experiments run are of two types:

1. for each rebalancing interval, we perform a validation-test split ($25\% - 75\%$) in order to extract the best values for H and Λ: then, the performance of OGDM with the resulting parameters is compared with the average performance of independent experts and the performance of the two most similar available experts to the parametrization of [Sarmento and Horta, 2020]; notice that here we do not account for transaction costs; moreover, the grid of tested values is reported in Table 3;
2. we repeat the experiments to assess the impact of considering transaction costs: in particular, we kept the optimal values for H and Λ found previously and run the experiments on the whole dataset; this time, the benchmark is just the average performance of independent experts, transaction costs included.

For both of them, we considered the same dataset as in 3.1, but from 3rd January 2000 to 31st December 2021.

OGDM parameter name	Values
H **(learning rate)**	0, 0.05, 0.1, 0.5, 1, 5, 10, 20
Λ **(momentum)**	0, 0.05, 0.1, 0.5, 1, 5, 10, 20

Table 3: OGDM hyperparameters optimisation grid.

5. Related Works

The application of machine learning to pairs trading is still a quite unexplored topic, mainly because most of the research is proprietary. Nonetheless, many recent researches (e.g. [Krauss et al., 2017] and [Huck, 2019]) report that pairs trading strategies experience a performance reduction from around 2000 onward, suggesting that the market has become more efficient and validating the previous claim.

The documents available in literature can be roughly divided into two categories: those applying machine learning to pairs selection and those applying it to the trading phase. Starting from the former, we have the already cited [Krauss et al., 2017], where machine learning models try to predict whether an asset will perform better or worse than the market: then, assets are ranked and pairs are generated by associating the k-th asset with the (n-k)-th (thus, the expected best with the worst and so on). A similar idea was already presented by [Huck, 2009] and [Huck, 2010], where neural networks are employed to predict the asset performance: however, such predictions are one of the inputs of ELECTRE III, a fuzzy system which actually establish the final ranking. The ranking idea to generate pairs is also used in [Han et al., 2021], where PCA and clustering are used, similarly to our research, to identify groups of similar assets, which are then paired on the basis of their last one month return (again, best with worst and so on). For what concerns the Hurst exponent, [Ramos-Requena et al., 2021] employed it to select pairs among low volatility stocks, which are then traded with fixed thresholds on their spread series. Finally, the concept of cointegration is expanded in [Clegg and Krauss, 2018] with the introduction of Partial Co-Integration (PCI), a similar relationship which takes also into consideration a random walk component in the spread series.

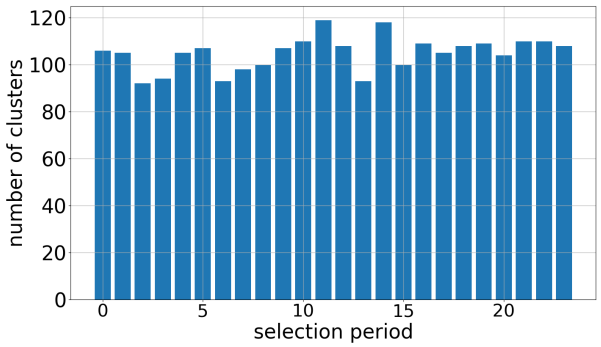
The machine learning applications to the trading phase include trying to predict the asset trends (as already reported in, e.g., [Krauss et al., 2017]), dynamically decide the opening/closing/stop-loss thresholds, as in [Kim and Kim, 2019], or directly the actions to be performed, as in [Wang et al., 2021]. Notice that deep reinforcement learning is usually employed in the last two categories, as reported in [Kim et al., 2022], that is, deep neural network are used among the inputs of reinforcement learning algorithms: in particular, in this last research, both actions and thresholds are predicted, while PCA and clustering are employed to generate the inputs of the DNN.

Finally, some online learning literature is available, although only for the trading part, as in [Fallahpour et al., 2016], where trading is modelled by a Multi-Armed Bandit problem (MAB) on pairs selected on several rules, including a cointegration test.

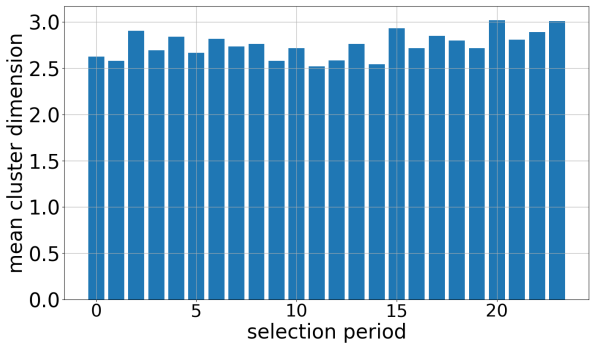
6. Exploratory Analysis

Before running the experiments described in Section 4.3, some more exploratory analysis have been conducted. Specifically, we considered the same 1st January 2010 - 12th June 2021 period as in Section 3.1 and extracted some statistics about the clusters and pairs selected by the [Sarmento and Horta, 2020] pipeline. However, let us point out that, although the suggested hyperparameters were used, the formation and trading periods are slightly different: the former lasts one year and is followed by six months of trading. The two periods are then shifted six months ahead, thus obtaining disjointed trading periods, amounting to a total of 23 pairs selection periods. The trading environment considered is formed by the components of the S&P500 index, which includes the top 500 largest companies listed on stock exchanges in the United States, quoted with a daily frequency.

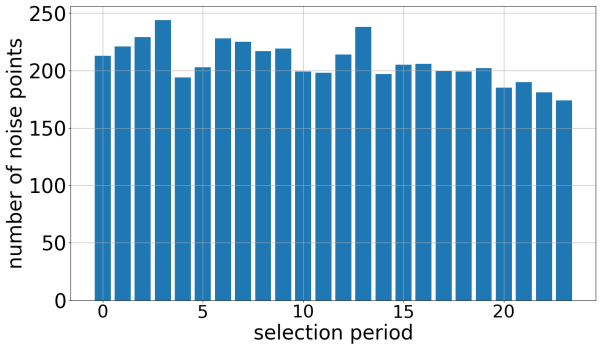
6.1. General Aspects



(a) Average number of clusters throughout the pairs selection periods.



(b) Average size of the clusters throughout the pairs selection periods.

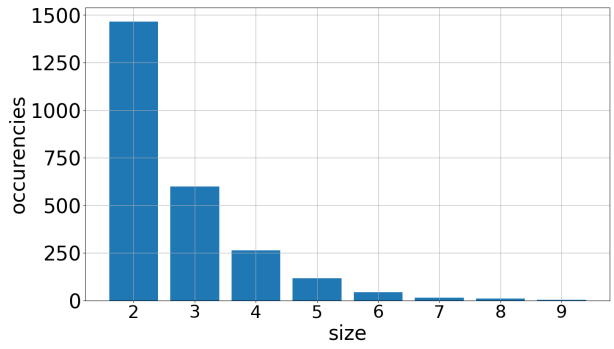


(c) Average number of noise points throughout the pairs selection periods.

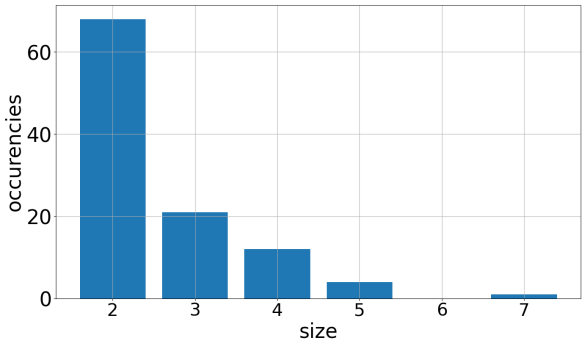
Figure 4: General cluster statistics: non-noise clusters are generally small (two to three elements) and about 40% of the assets (200/500) are classified as noise.

For what general statistics are concerned, in Figure 4a we can see the average number of clusters in each pairs selection period, which is stable around 100, while in Figure 4b the average size of such clusters is represented. It is stable at around 2.7 elements per cluster: thus, we can conclude that the clusters identified were generally small, with not much variance across selection periods.
Finally, Figure 4c reports the number of noise points in each period (i.e. the points that were not identified as belonging to any cluster): we can notice that almost half of the dataset is classified as noise, namely around 200 elements per period.

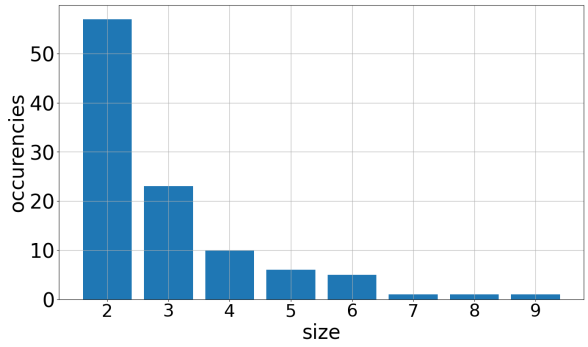
6.2. Size Frequencies



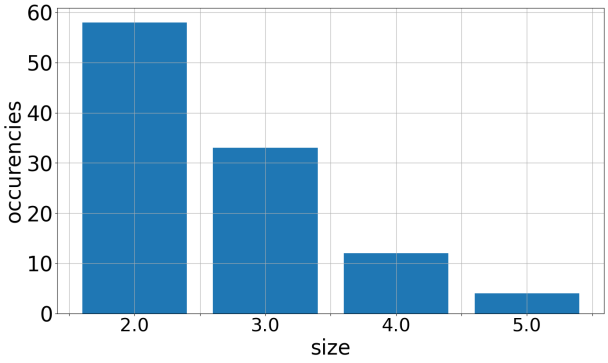
(a) Cluster size frequencies across all periods.



(b) Cluster size frequencies in period 0.



(c) Cluster size frequencies in period 20.

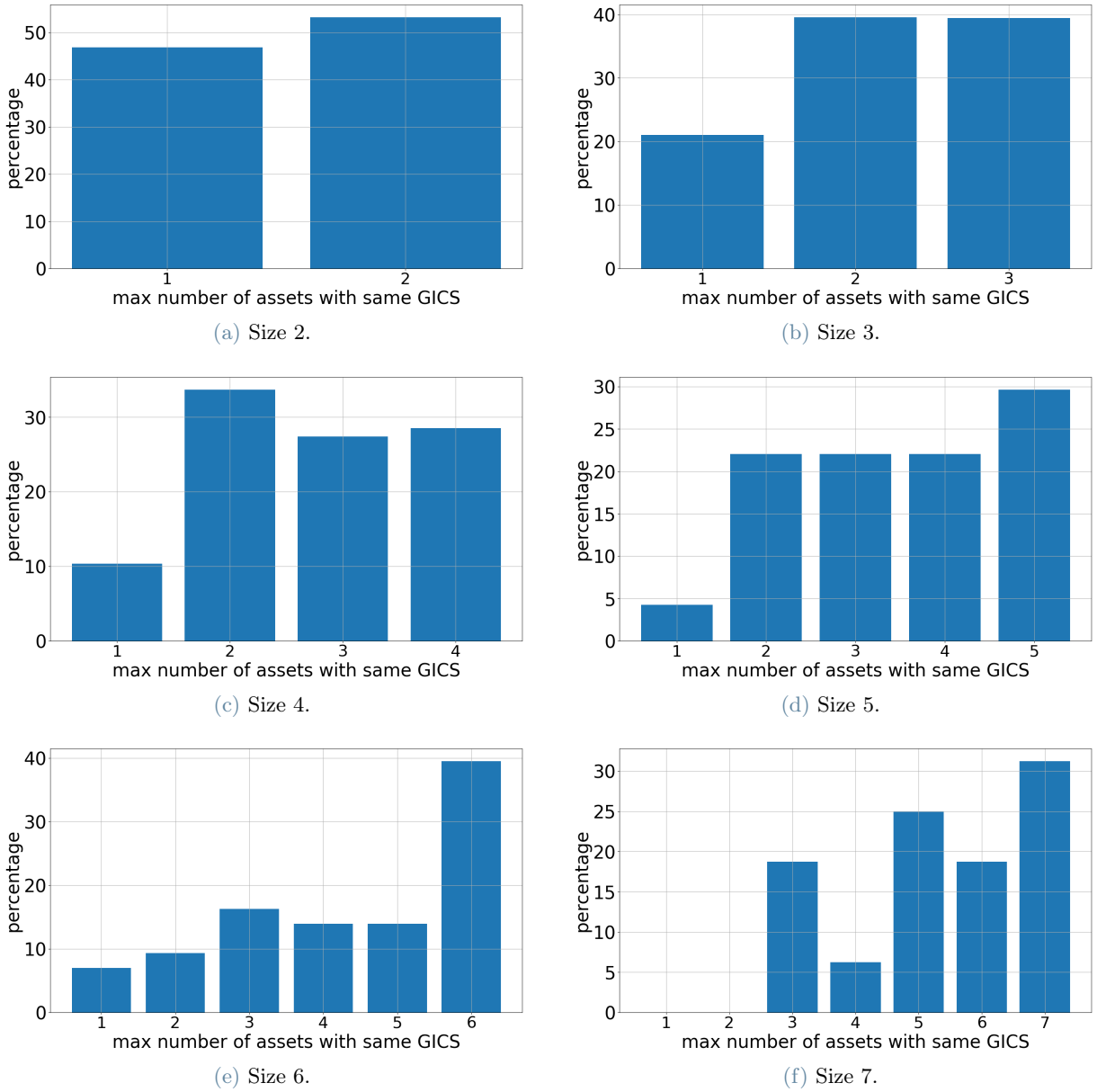


(d) Cluster size frequencies in period 5.

Figure 5: Cluster size frequencies summary: the majority of the clusters has 2 or 3 elements, exponentially decreasing as the size increases.

We can further explore the size of the clusters by plotting the frequencies of the sizes, which can be seen in Figure 5a: the most prominent column is the one for size 2, representing almost half of all the clusters, while the distribution steadily decreases as the size increases.
Going deeper, we can analyse the distribution of sizes in each period: although they would be too many to be reported here, some notable examples are reported in Figure 5b, 5c and 5d. In particular, the distributions were generally like period 0 (Figure 5b), with the most extreme being period 5 (Figure 5d), where the decrease is close to linear.
As a final note on the subject, let us notice how effective the technique is at reducing the computational effort: pairs generated from such small clusters are much less than a complete (sometimes unfeasible) search.

6.3. GICS Consistency



(a) Size 2. (b) Size 3.

(c) Size 4. (d) Size 5.

(e) Size 6. (f) Size 7.

Figure 6: Percentage of clusters w.r.t. maximum number of elements with same GICS code, divided by cluster size: the bigger the size of the cluster, the higher the probability that it will be consistent to the GICS classification.

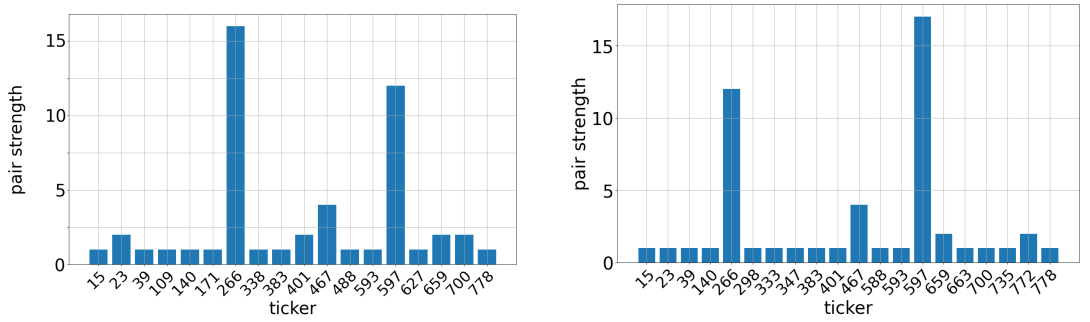
In this Section and in the following one, we will assess the consistency of the clusters from different points of view. The first and most trivial is the GICS classification: it is an industry taxonomy developed by MSCI and S&P to classify companies on the basis of their sector, that is, same GICS code means same industry group. Thus, in Figure 6 we can see the percentage of clusters w.r.t. their the maximum number of elements with the same GICS code inside them.

For example, Figure 6a shows that, among all size 2 clusters, almost half of them has at most 1 GICS code in common among the elements of which is composed (that is, every element has a different one), while the other half has 2 elements with the same GICS code (that is, being the cluster formed by 2 elements, it is fully consistent with the GICS classification).

We can observe that bigger clusters tend to be more consistent with the GICS classification, having the highest bars in the right part of the graph, so more elements with a common GICS code. On the other hand, smaller clusters have a more uniform-like distribution.

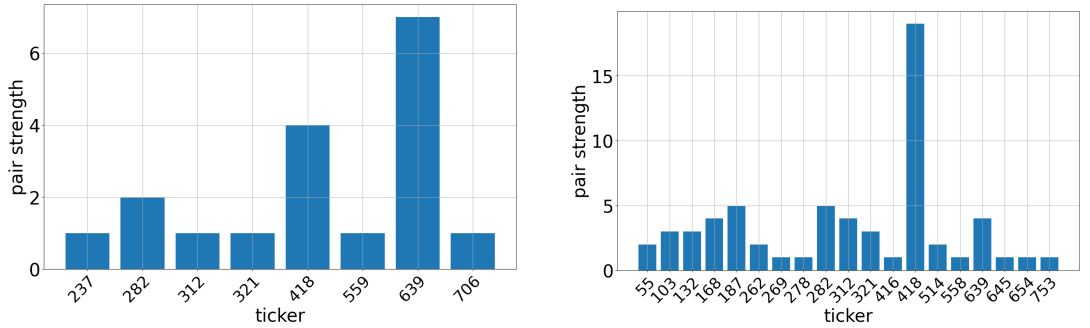
These results are consistent with [Sarmento and Horta, 2020], that is, clusters are generally composed of elements with the same GICS code, although this is not mandatory: thus, the algorithm is capable of identifying relationships even between structurally distant assets.

6.4. Cluster Consistency



(a) Pair strengths from asset #266 point of view. (b) Pair strengths from asset #597 point of view.

Figure 7: Example of symmetric pair.



(a) Pair strengths from asset #639 point of view. (b) Pair strengths from asset #418 point of view.

Figure 8: Example of asymmetric pair.

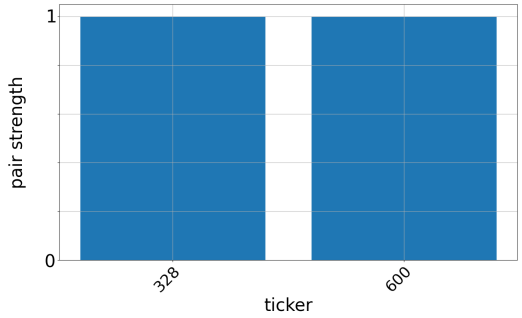
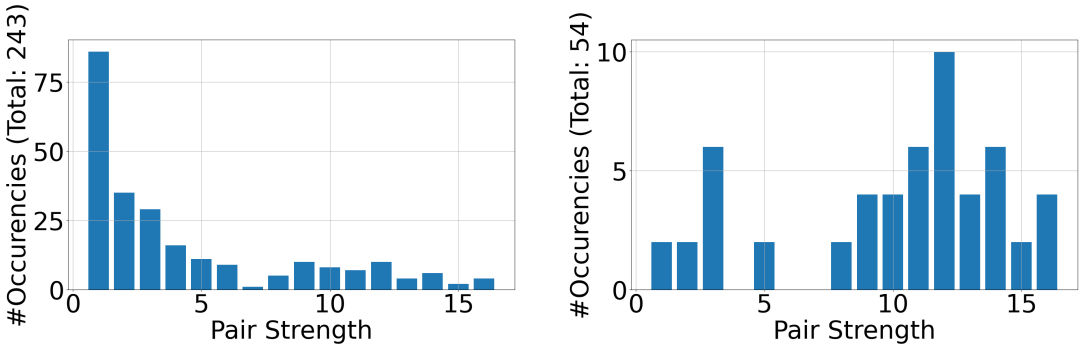


Figure 9: Pair strengths from asset #600 point of view.

Since OPTICS identified small clusters, the difference between a cluster and the pairs generated from it is narrow, at least most of the time. Therefore, we thought about looking for tickers (i.e. stock identification codes) that were in the same cluster most of the times. That is, if clustering actually helps and the relationships found are not spurious, it should consistently group together similar tickers. Thus, some definitions will be useful going forward:

- pair strength: number of times two assets were clustered together;
- cluster time: number of times an asset belong to a cluster (with at least two elements);



(a) Consistent pairs strength frequencies.

(b) Symmetric consistent pairs strength distribution.

Figure 10: Consistent vs Symmetric consistent pairs strength frequencies.

- consistent pair: (x,y) is a consistent pair from x's point of view if

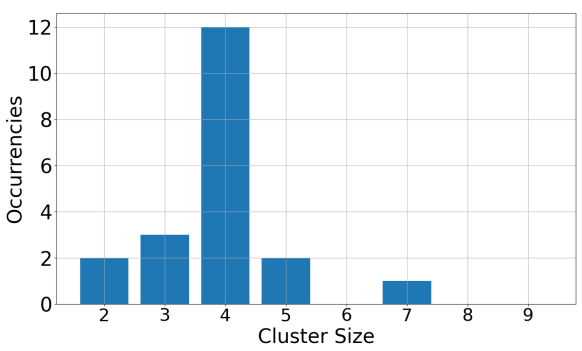
$$PairStrength(x,y) > 0.5 \cdot ClusterTime(x) = 0.5 \cdot PairStrength(x,x), \tag{19}$$

  where x and y are two assets; this means that we consider x and y consistent w.r.t. x if at least half of the time x belongs to a cluster, also y is in the same cluster: thus it is not a symmetric relationship;
- symmetric (consistent) pair: when both (x,y) and (y,x) are consistent pairs.

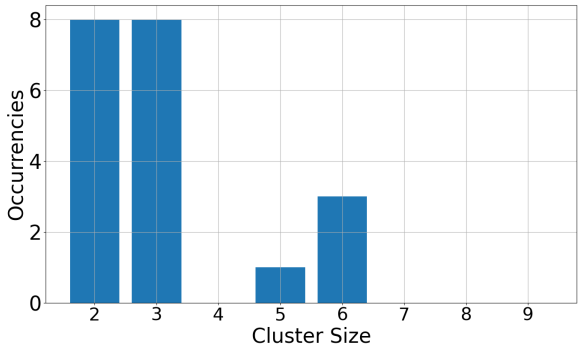
For example, assets #266 and #597 are in a symmetric pair, since their pair strength is greater than both their $0.5 \cdot ClusterTime$, as can be seen from Figure 7. However, this is far from being a common behaviour: for example, asset #639 is in a consistent pair (from its point of view) with asset #418, but not vice versa, as illustrated in Figure 8.

Finally, let us acknowledge that such a measure of consistency is subject to outliers: the most trivial example is when an asset is clustered just a single time, for example asset #600, as shown in Figure 9. In order to take care of such possible source of noise, we took a look at the pair strength distribution for consistent pairs: in Figure 10a we can see that these "outliers" are actually a third of the total. Moreover, we can notice that the distribution is decreasing until 8, but then it starts a singular behaviour. Plotting the same graph but for symmetric pairs, which can be seen in Figure 10b, we understand that a lot of them are in the $[8, 16]$ interval: thus, they generate the unusual growth seen in the previous graph. Notice that we were exactly looking for these pairs: assets that are not only frequently clustered, but also often with the same neighbour (at least one). Therefore, the next question arises naturally: are symmetric consistent pairs (which should be the most similar pairs from a structural point of view, as suggested by the clustering) also the selected pairs? Unfortunately, they were not, as can be seen at the end of Section 6.5.

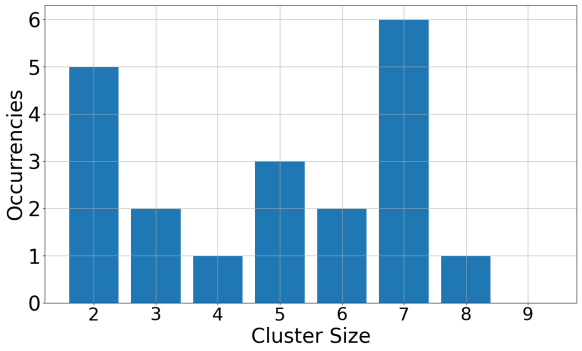
6.5. Selected pairs Analysis



(a) Period 0.



(b) Period 11.



(c) Period 19.

Figure 11: Distribution of clusters where selected pairs belong to, for different periods.

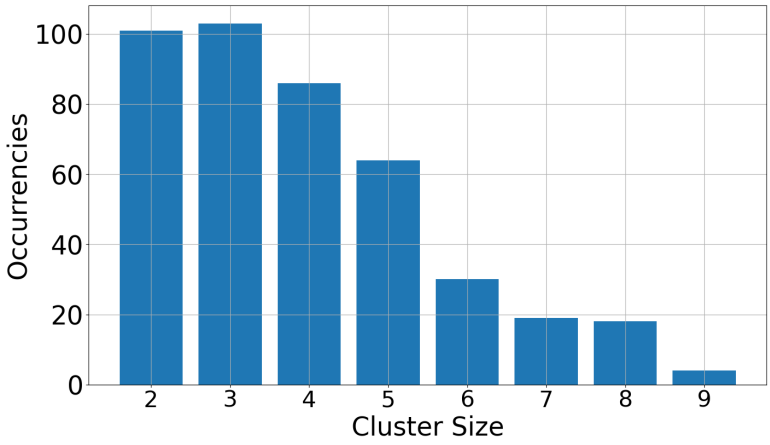
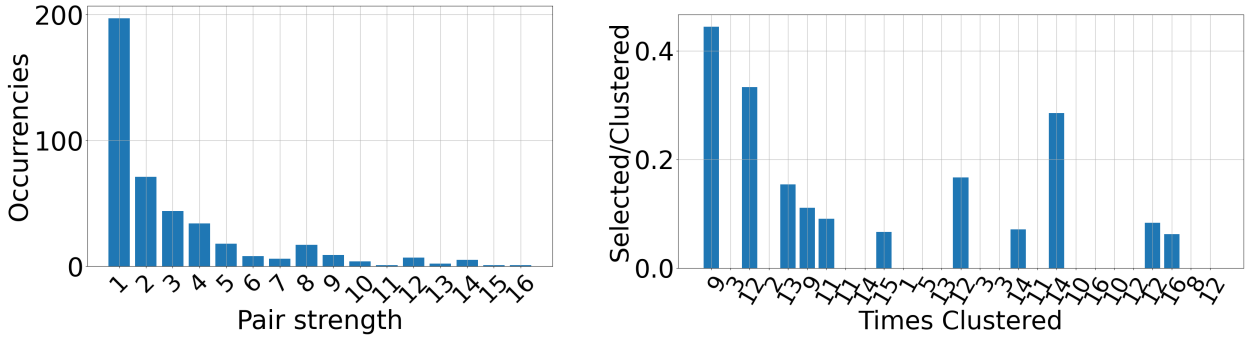


Figure 12: Distribution of cluster sizes of selected pairs.

In the previous Section, we provided an analysis regarding the performance of the clustering algorithm: a similar investigation is here assessed regarding the selected pairs only. Thus, in Figure 11 we can see three examples of cluster size frequencies where selected pairs came from. Although no common pattern can be identified, the general picture is of a similar distribution to the one related to the clusters size (which was shown in Figure 5a), as we can see from Figure 12. In fact, it peaks early and then decreases in a superlinear fashion. Notice that, however, the peak is at 3 and the decrease is quite smoother than the other distribution, causing the mean size to increase (3.9 vs 2.7): this behaviour suggests that, despite being one of the main sources of pairs, small clusters may introduce some noise too. Such a result is coherent with what was found in Section 3.1, where low values for the MinPts parameter were not always the optimal choice.



(a) Selected pairs strength frequencies.

(b) Percentage of times a symmetric pair was selected over the times it was clustered (seen as the x axis tick).

Figure 13: Symmetric pairs are not a remarkable portion of the selected ones.

Finally, in Figure 13a the pairs strength frequencies for selected pairs is shown: again, similar distribution to the one shown in Figure 10a, although we miss the anomalous region after 8.
The reason is that symmetric pairs are almost never selected, as can be seen from Figure 13b, where 24 over 27 pairs were selected less than 20% of the times they were in the same cluster. Therefore, it seems that clustering helps in reducing the computational effort required to search for promising pairs, indeed, but the cointegration test, along with the other rules, are still crucial to obtain robust mean-reverting spread series.

7. Experimental Results

Here we present the results of the experiments mentioned at the end of Section 4.3: we start by applying OGDM with the best parameters found on the validation set (first quarter of the dataset) on the remaining period of time. The results of both validation and test set are reported in Section 7.1, while an analysis of the underlying budget distribution among experts is the object of Section 7.2. Lastly, we show the findings of the experiments with different level of transaction costs in Section 7.3.

7.1. Transaction Costs-Free Experiments

In the following Section we will present the results of the experiments that assumed a negligible cost for transactions. The cumulative wealth obtained by fine-tuned OGDM is compared with the baseline strategies, then, the resulting budget distribution among the experts is investigated.
Starting from Figure 14, we report the cumulative wealth obtained by OGDM on the test set, with the parameters selected on the validation dataset, along with two baselines: the first one is the sample mean obtained from the *independent* experts from the grid, while the second one are the two available experts that are closest to the trading scenario proposed in [Sarmento and Horta, 2020]. For what concerns the *independent setting*, it is the average of the performance of the experts without forcing an equal distribution of the budget at each rebalancing step (as would happen in the case of $\eta_t = \lambda_t = 0, \forall t$). Instead, if we recall [Sarmento and Horta, 2020] trading strategy, the suggested parametrization of [p-value,PCA features,MinPts,formation period] would be $[0.05, 4, 2, 520]$: in order to provide a fair comparison with respect to OGDM, we will consider expert 48 with $[0.03, 4, 2, 500]$ and expert 374 with $[0.05, 4, 4, 400]$ as approximations, since they are the most similar ones inside the set of experts considered for OGDM application. Let us observe that the algorithm was not initialized again once the test started, in order to stick to a realistic scenario.

Notably, as depicted in Figure 14, performance varies significantly on the rebalancing interval basis: in fact, while good optimization was achievable for 7-14-63 days, 21 and 42 yielded almost no improvement over the independent approach. One of the reasons is that, along with the rebalancing interval, the underlying problem changes too: indeed, allowing pairs to be traded for a longer period of time may allow for greater gains/losses, while diminishing the frequency of the OGDM updates may result in missing trading opportunities. These two facts are in a trade-off, thus exploring the rebalancing interval impact was one of the purposes of this research.

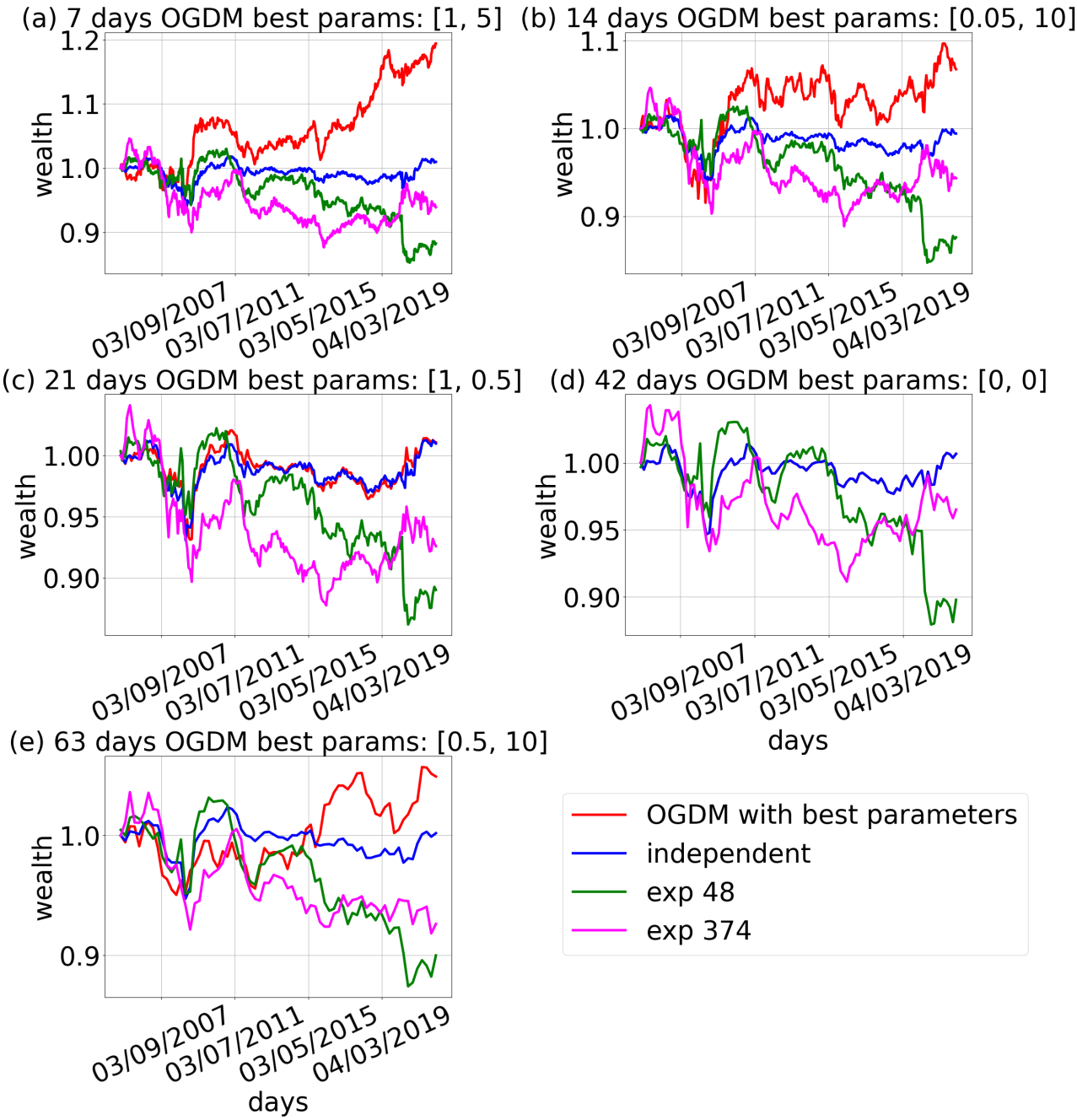


Figure 14: Total wealth obtained in the test set by OGDM with best parameters, divided by rebalancing interval and compared w.r.t. the average of independent experts and the two experts that were similar the most to the suggested parameters of [Sarmento and Horta, 2020].

For what concerns the values of H and Λ, the optimal combination seems to be ad-hoc for each rebalancing interval length, with the parameters being optimized at different values over the four intervals where OGDM managed to outperform the $[0, 0]$ choice. This result is in line with how the problem itself changes with the rebalancing interval length: different signal-to-noise ratio and overall dynamics of the system yield different best parameters.

By looking at the graphs individually, we can notice that the goodness of the optimization changes over time: for example, in Figure 14e, we can see an initial long period of difficulties in the optimization, followed by a strong increasing trend towards the start of 2016. However, considering the long period of time analysed, these kind of variations were likely to be expected: market conditions may change often during a 15 years time span, thus the algorithm may need some time to adapt. Nevertheless, the parameters suggested in [Sarmento and Horta, 2020] proved to be quite suboptimal for the vast majority of time, no matter the rebalancing interval.

Even though [Sarmento and Horta, 2020] parameters performed poorly during the test set, they would have been a strong choice during the validation period: as we can see from Figure 15, expert 48 yields the best results,

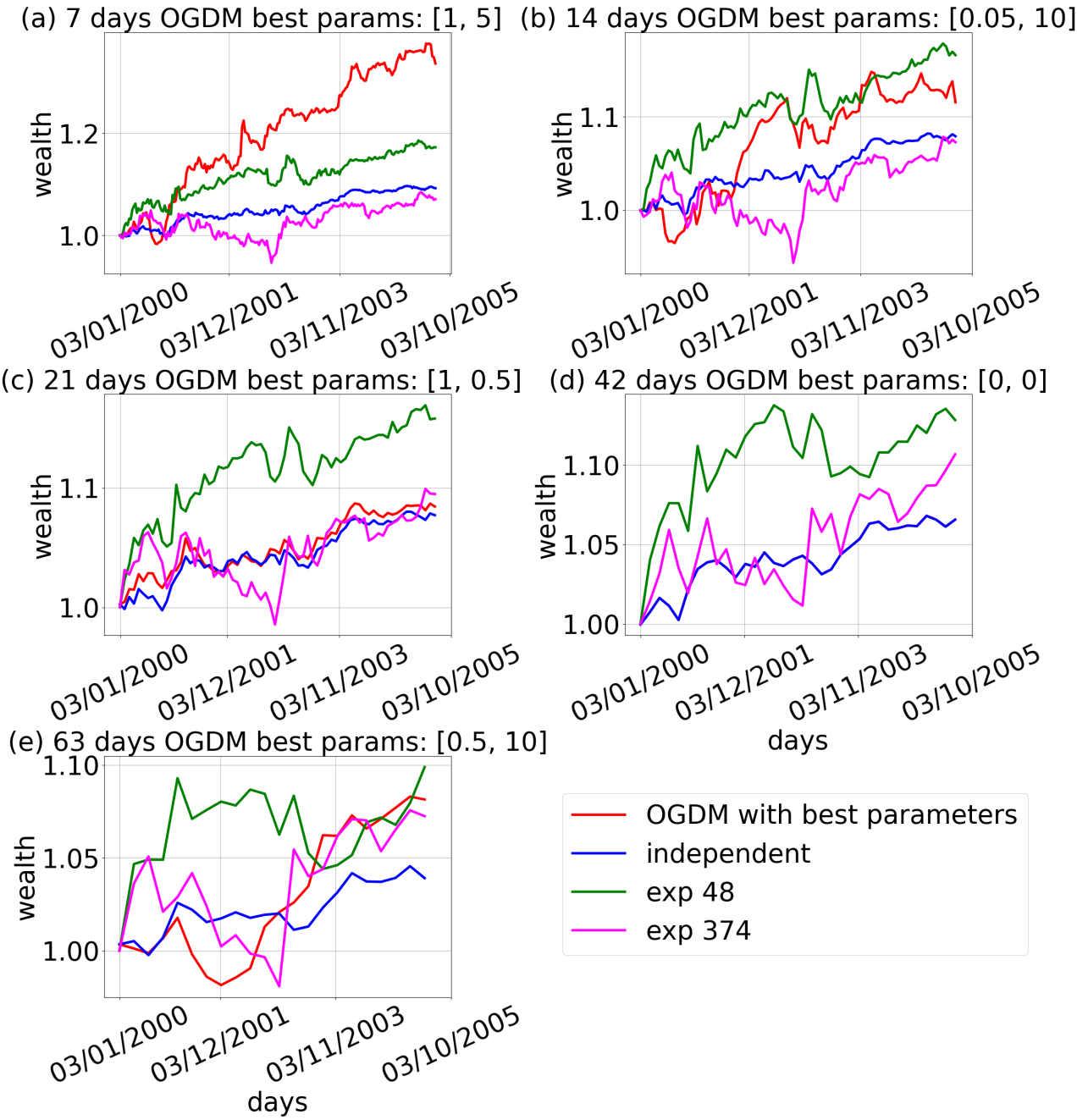
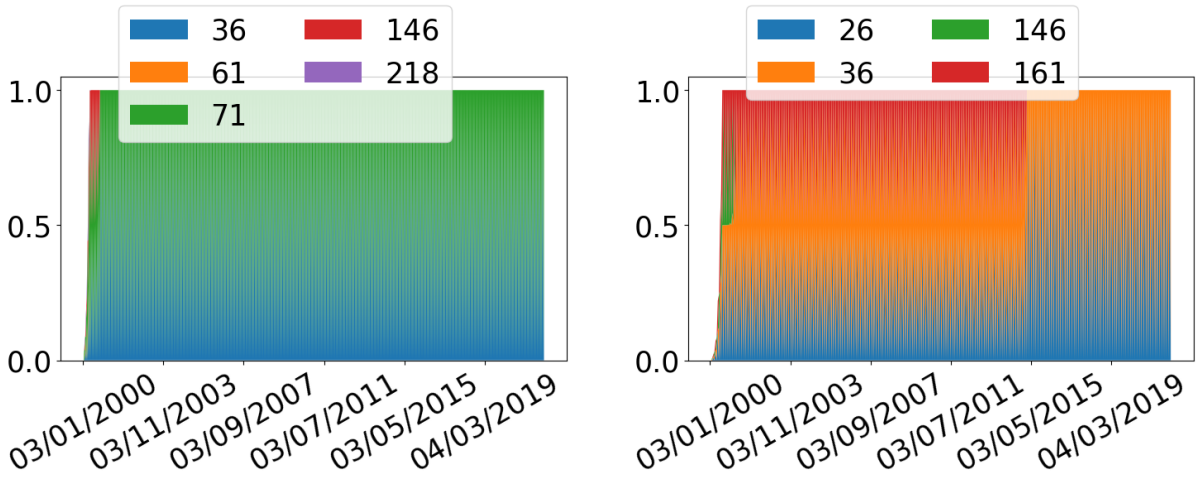


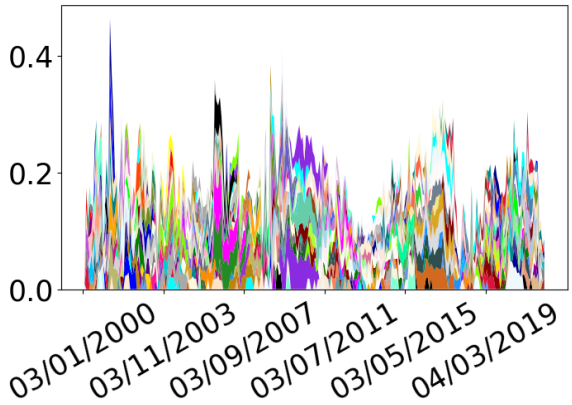
Figure 15: Total wealth obtained in the validation set by OGDM with best parameters, divided by rebalancing interval and compared w.r.t. the average of independent experts and the two experts that were similar the most to the suggested parameters of [Sarmento and Horta, 2020].

except for (a) where it is still the best baseline. This fact remarks the importance of a dynamic strategy: the best pairs trading parametrization changes over time, as already reported in Section 3.1, thus trying to adapt as information is available yields better long-term gains. Moreover, the parameters sensitivity of the strategy is well visualized here: expert 374, numerically similar to 48, has mixed results, but generally worse ones. Thus, switching to an ensemble of strategies, as with OGDM, helps in smoothing such sensitivity by reducing the variance and adapting to the ever-changing market.

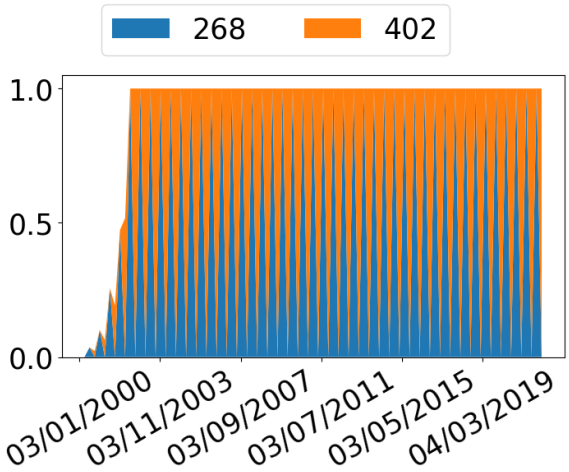


(a) 7 days interval.

(b) 14 days interval.



(c) 21 days interval.



(d) 63 days interval.

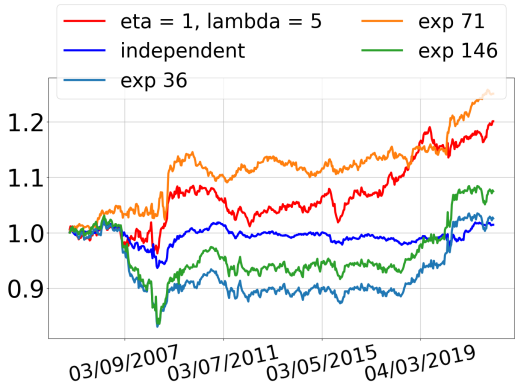
Figure 16: Budget distribution for the transaction costs-free experiments (top 10 heaviest experts). Notice how (a), (b) and (d) oscillate between two experts after the first time steps: this behaviour is most likely related to the high value of the momentum parameter selected during the validation step.

7.2. Budget Distribution Analysis

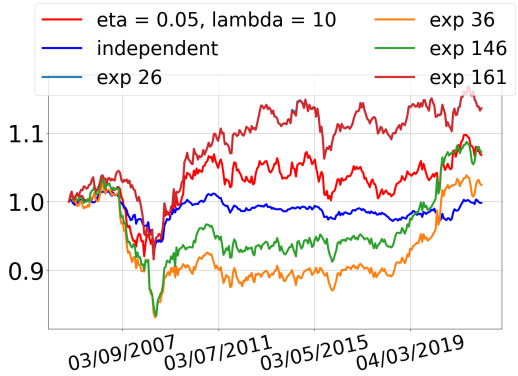
Another interesting quantity that can be extracted form OGDM results is the budget distribution, that is, how money is distributed among experts during the considered time span. In order to provide a useful visualization, since plotting the budget line of each expert would be unfeasible, we decided to show only the 10 experts with the largest weights (approximately 2.5% of the total) at each time step; moreover, we exclude from the representation experts that would be printed less than five times and present the results with a stacked plot. As we can see from Figure 16, we have two kinds of behaviour:

- the expected, chaotic distribution shown in Figure 16c, where we chose not to report the legend due to its dimension: in fact, a lot of different experts enter and exit the top 10, suggesting that the algorithm is rapidly adapting to the environment feedback; even though no expert is particularly relevant, we can see that about 20% of the budget is allocated to the top 2.5% experts: since the distribution is not particularly different from the uniform one, results are similar to the independent baseline;
- the other Figures tell a different story: OGDM is able to converge to a specific set of well-performing experts. However, the budget allocation degenerates to an oscillatory behaviour, resulting in moving the whole budget alternatively from an expert to another one;
- as a final note, the 42 days interval graph has been omitted since no useful $[H, \Lambda]$ were found, resulting in a constant uniform budget distribution.

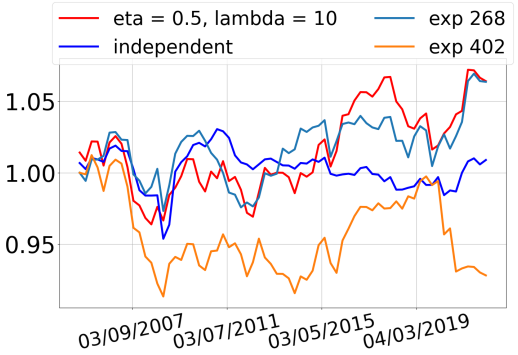
In order to better understand such a swinging behaviour, in Figure 17 we report the cumulative wealth of OGDM



(a) 7 days interval.



(b) 14 days interval.



(c) 63 days interval.

Figure 17: Total test wealth obtained by OGDM with best parameters, divided by rebalancing interval and compared w.r.t. the average of independent experts and the main experts involved.

alongside the main experts where the budget was allocated. Then, we can understand that the budget was generally split among two experts: one that performed way better than the average and one that is subject to less variance but is actually below average.

All things considered, we can conclude that we are facing some kind of overfitting: in fact, the three configurations share a high value of the momentum parameter, resulting in weighing the first updates way more than the following ones. Despite this undesired phenomenon, the performance on unseen data were good, with all three parametrizations performing quite better than both the average and the [Sarmento and Horta, 2020] suggested parameters.

7.3. Experiments with Transaction Costs

We will now address the results obtained in a realistic scenario, where fees have to be payed in order to execute transactions. In the following tests, parameter γ took the values $[0.001, 0.005, 0.01]$, which approximately translate to low, medium and high transaction costs environments.

The cumulative wealths from the experiments with transaction costs are reported in Figure 18: as we can see, both the naive independent implementation and the OGDM implementation would suffer in a real world scenario. Due to computational issues, it was not possible to obtain the curves of the experts similar to [Sarmento and Horta, 2020]: still, we can safely assume that introducing transaction costs would not improve the performance with respect to the independent average.

When looking at the performance of OGDM, we immediately see that the shorter the rebalancing interval is, the steeper the performance decreases: this is clearly related to the OGDM updates, which may happen to move money from/to open positions, causing transactions to happen. However, notice how the naive strategy is highly suffering form transaction costs too: thus, it may suggest that most of the costs are due to opening and closing positions rather than the OGDM optimisation.

Despite the aforementioned difficulties when dealing with transaction costs, let us remark the good performance of OGDM in Figure 18e: despite the additional price of the optimisation (i.e. more transactions) there is a clear improvement in the profit when transaction costs are low.

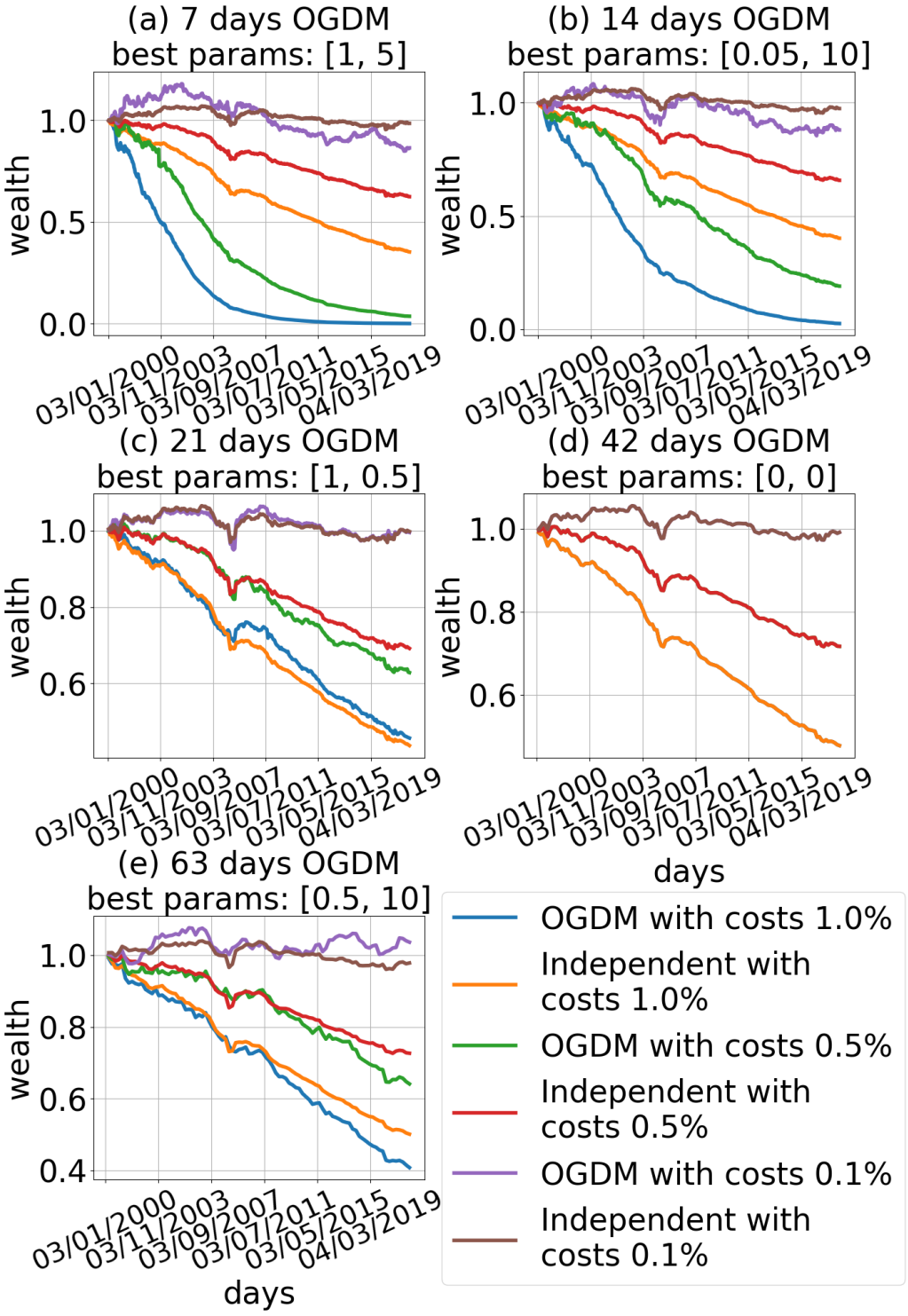


Figure 18: Total wealth obtained on the whole dataset by OGDM with best parameters, divided by rebalancing interval, for different costs of transaction and compared w.r.t. the average of independent experts.

8. Conclusions

We provided an extension to the work of [Sarmento and Horta, 2020] by analyzing which hyperparameters could be suitable for optimisation in a daily stock market context, namely, the S&P 500 one. We found out that the formation period length, the number of PCA features, OPTICS MinPts parameter and the p-value threshold were the most important, while the rest of the rules parameters (that is, Hurst exponent, mean-reversion half-life and average number of mean-crossings per year) were not crucial. Furthermore, we analysed the clusters obtained by the default [Sarmento and Horta, 2020] parameters, discovering that selected pairs were not consistent with the suggested clustering, being them often in different clusters, or identified as noise, with the exception of the times they were selected.

Then, we actually performed the optimisation of the most suitable parameters through the OPO framework and the OGDM algorithm: the suggested parameters from [Sarmento and Horta, 2020] showed to be suboptimal for each one of the rebalancing intervals tested, while OGDM managed to outperform both the suggested parameters and the average independent expert when transaction costs were not involved. Notice, however, that the suggested parameters would have been a good choice during the validation period, while they decline in profit during the testing one: these findings suggest that some sort of adaptive strategy is needed in order to optimise the wealth obtained. The budget distribution over the experts was analysed too: despite the witnessing of some sort of overfitting, the overall performance were good, even if they came from unseen data.

Moving on to the experiments that included transaction costs, we found that the whole strategy suffers: despite OGDM had the worst performance lines, possibly due to its increased transaction rate, the mean results obtained by adopting independent experts is far from generating profit too. The only exception was for the largest rebalancing interval and lowest transaction cost rate: in this case, the overhead introduced by OGDM was outclassed by the additional gains generated, achieving actual profit at the end of the trading period considered.

Many different directions could constitute an extension to the presented work: starting from the expert generation, some metric of similarity could be employed to apply OGDM on a restricted selection of parametrizations, possibly discarding similar ones in terms of both performance and hyperparameters used. Particularly, clustering methods could perform such filtering, achieving many improvements, such as increasing the signal-to-noise ratio from the environment and an easier visualization of the budget distribution. The underlying data play a leading role too: allowing pairs to be more complex than two assets (e.g. an asset vs the other components of its cluster) may identify more stable relationships. A similar idea was already proposed in [Galenko et al., 2012], thus applying our framework on top of their procedure may yield good results. Lastly, an in-depth analysis of the resulting portfolio in terms of distribution over pairs or assets, rather than experts, may suggest the removal of the experts layer. As a result, OGDM would be applied to a portfolio of spreads, simplifying the overall structure and allowing for an easier economic interpretation.

References

[Ankerst et al., 1999] Ankerst, M., Breunig, M. M., Kriegel, H.-P., and Sander, J. (1999). Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60.

[Barabási and Vicsek, 1991] Barabási, A.-L. and Vicsek, T. (1991). Multifractality of self-affine fractals. *Physical review A*, 44(4):2730.

[Barunik and Kristoufek, 2010] Barunik, J. and Kristoufek, L. (2010). On hurst exponent estimation under heavy-tailed distributions. *Physica A: Statistical Mechanics and its Applications*, 389(18):3844–3855.

[Berkhin, 2006] Berkhin, P. (2006). A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer.

[Caldeira et al., 2013] Caldeira, J. F., Moura, G. V., et al. (2013). Selection of a portfolio of pairs based on cointegration: A statistical arbitrage strategy. *Brazilian Review of Finance*, 11(1):49–80.

[Cesa-Bianchi and Lugosi, 2006] Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge university press.

[Chen et al., 2012] Chen, H. J., Chen, S. J., and Li, F. (2012). Empirical investigation of an equity pairs trading strategy.

[Clegg and Krauss, 2018] Clegg, M. and Krauss, C. (2018). Pairs trading with partial cointegration. *Quantitative Finance*, 18(1):121–138.

[Di Matteo et al., 2003] Di Matteo, T., Aste, T., and Dacorogna, M. M. (2003). Scaling behaviors in differently developed markets. *Physica A: Statistical Mechanics and its Applications*, 324(1-2):183–188.

[Duchi et al., 2008] Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279.

[Engle and Granger, 1987] Engle, R. F. and Granger, C. W. (1987). Co-integration and error correction: representation, estimation, and testing. *Econometrica: journal of the Econometric Society*, pages 251–276.

[Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

[Fallahpour et al., 2016] Fallahpour, S., Hakimian, H., Taheri, K., and Ramezanifar, E. (2016). Pairs trading strategy optimization using the reinforcement learning method: a cointegration approach. *Soft Computing*, 20(12):5051–5066.

[Galenko et al., 2012] Galenko, A., Popova, E., and Popova, I. (2012). Trading in the presence of cointegration. *The Journal of Alternative Investments*, 15(1):85–97.

[Gatev et al., 2006] Gatev, E., Goetzmann, W. N., and Rouwenhorst, K. G. (2006). Pairs trading: Performance of a relative-value arbitrage rule. *The Review of Financial Studies*, 19(3):797–827.

[Han et al., 2021] Han, C., He, Z., and Toh, A. J. W. (2021). Pairs trading via unsupervised learning. *Available at SSRN 3835692*.

[Huck, 2009] Huck, N. (2009). Pairs selection and outranking: An application to the s&p 100 index. *European Journal of Operational Research*, 196(2):819–825.

[Huck, 2010] Huck, N. (2010). Pairs trading and outranking: The multi-step-ahead forecasting case. *European Journal of Operational Research*, 207(3):1702–1716.

[Huck, 2019] Huck, N. (2019). Large data sets and machine learning: Applications to statistical arbitrage. *European Journal of Operational Research*, 278(1):330–342.

[Huck and Afawubo, 2015] Huck, N. and Afawubo, K. (2015). Pairs trading and selection methods: is cointegration superior? *Applied Economics*, 47(6):599–613.

[Hurst, 1951] Hurst, H. E. (1951). Long-term storage capacity of reservoirs. *Transactions of the American society of civil engineers*, 116(1):770–799.

[Kim et al., 2022] Kim, S.-H., Park, D.-Y., and Lee, K.-H. (2022). Hybrid deep reinforcement learning for pairs trading. *Applied Sciences*, 12(3):944.

[Kim and Kim, 2019] Kim, T. and Kim, H. Y. (2019). Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries. *Complexity*, 2019.

[Krauss et al., 2017] Krauss, C., Do, X. A., and Huck, N. (2017). Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the s&p 500. *European Journal of Operational Research*, 259(2):689–702.

[Krauss et al., 2015] Krauss, C. et al. (2015). Statistical arbitrage pairs trading strategies: Review and outlook. *FAU Discussion Papers in Economics*, (09/2015).

[Ramos-Requena et al., 2021] Ramos-Requena, J., López-García, M., Sánchez-Granero, M., and Trinidad-Segovia, J. (2021). A cooperative dynamic approach to pairs trading. *Complexity*, 2021.

[Sarmento and Horta, 2020] Sarmento, S. M. and Horta, N. (2020). Enhancing a pairs trading strategy with the application of machine learning. *Expert Systems with Applications*, 158:113490.

[Vittori et al., 2020] Vittori, E., de Luca, M. B., Trovò, F., and Restelli, M. (2020). Dealing with transaction costs in portfolio optimization: online gradient descent with momentum. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8.

[Wang et al., 2021] Wang, C., Sandås, P., and Beling, P. (2021). Improving pairs trading strategies via reinforcement learning. In *2021 International Conference on Applied Artificial Intelligence (ICAPAI)*, pages 1–7. IEEE.

A. Appendix A

From Table 4 to 9 we report the full list of hyperparameters configurations (i.e. experts) among which OGDM divided its budget: for space reasons, every line has two experts, each one identified by a [p-Value,PCA,MinPts,Formation] tuple. See Section 3.1 for a complete explanation of the parameters meaning.

p-Value	PCA	MinPts	Formation	p-Value	PCA	MinPts	Formation
0,03	1	2	360	0,03	1	10	800
0,03	1	2	400	0,03	1	10	1080
0,03	1	2	500	0,03	1	12	360
0,03	1	2	600	0,03	1	12	400
0,03	1	2	700	0,03	1	12	500
0,03	1	2	800	0,03	1	12	600
0,03	1	2	1080	0,03	1	12	700
0,03	1	4	360	0,03	1	12	800
0,03	1	4	400	0,03	1	12	1080
0,03	1	4	500	0,03	1	13	360
0,03	1	4	600	0,03	1	13	400
0,03	1	4	700	0,03	1	13	500
0,03	1	4	800	0,03	1	13	600
0,03	1	4	1080	0,03	1	13	700
0,03	1	6	360	0,03	1	13	800
0,03	1	6	400	0,03	1	13	1080
0,03	1	6	500	0,03	1	20	360
0,03	1	6	600	0,03	1	20	400
0,03	1	6	700	0,03	1	20	500
0,03	1	6	800	0,03	1	20	600
0,03	1	6	1080	0,03	1	20	700
0,03	1	8	360	0,03	1	20	800
0,03	1	8	400	0,03	1	20	1080
0,03	1	8	500	0,03	2	2	360
0,03	1	8	600	0,03	2	2	400
0,03	1	8	700	0,03	2	2	500
0,03	1	8	800	0,03	2	2	600
0,03	1	8	1080	0,03	2	2	700
0,03	1	10	360	0,03	2	2	800
0,03	1	10	400	0,03	2	2	1080
0,03	1	10	500	0,03	2	4	360
0,03	1	10	600	0,03	2	4	400
0,03	1	10	700	0,03	2	4	500

Table 4: OGDM hyperparameters optimisation grid.

p-Value	PCA	MinPts	Formation	p-Value	PCA	MinPts	Formation
0,03	2	4	600	0,03	2	13	1080
0,03	2	4	700	0,03	2	20	360
0,03	2	4	800	0,03	2	20	400
0,03	2	4	1080	0,03	2	20	500
0,03	2	6	360	0,03	2	20	600
0,03	2	6	400	0,03	2	20	700
0,03	2	6	500	0,03	4	2	360
0,03	2	6	600	0,03	4	2	400
0,03	2	6	700	0,03	4	2	500
0,03	2	6	800	0,03	4	2	600
0,03	2	6	1080	0,03	4	2	700
0,03	2	8	360	0,03	4	2	800
0,03	2	8	400	0,03	4	2	1080
0,03	2	8	500	0,03	4	4	360
0,03	2	8	600	0,03	4	4	400
0,03	2	8	700	0,03	4	4	500
0,03	2	8	800	0,03	4	4	600
0,03	2	8	1080	0,03	4	4	700
0,03	2	10	360	0,03	4	4	800
0,03	2	10	400	0,03	4	4	1080
0,03	2	10	500	0,03	4	6	360
0,03	2	10	600	0,03	4	6	400
0,03	2	10	700	0,03	4	6	500
0,03	2	10	800	0,03	4	6	600
0,03	2	10	1080	0,03	4	6	700
0,03	2	12	360	0,03	4	6	800
0,03	2	12	400	0,03	4	6	1080
0,03	2	12	500	0,03	4	8	360
0,03	2	12	600	0,03	4	8	400
0,03	2	12	700	0,03	4	8	500
0,03	2	12	800	0,03	4	8	600
0,03	2	12	1080	0,03	4	8	700
0,03	2	13	360	0,03	4	8	800
0,03	2	13	400	0,03	4	8	1080
0,03	2	13	500	0,03	4	10	360
0,03	2	13	600	0,03	4	10	400
0,03	2	13	700	0,03	4	10	500
0,03	2	13	800	0,03	4	10	600

Table 5: OGDM hyperparameters optimisation grid.

p-Value	PCA	MinPts	Formation	p-Value	PCA	MinPts	Formation
0,03	4	10	700	0,03	10	8	360
0,03	4	10	800	0,03	10	8	400
0,03	4	10	1080	0,03	10	8	500
0,03	4	12	360	0,03	10	8	600
0,03	4	12	400	0,03	10	8	700
0,03	4	12	500	0,03	10	8	800
0,03	4	12	600	0,03	10	8	1080
0,03	4	12	700	0,03	10	10	360
0,03	4	12	800	0,03	10	10	400
0,03	4	12	1080	0,03	10	10	500
0,03	4	13	360	0,03	10	10	600
0,03	4	13	400	0,03	10	10	700
0,03	4	13	500	0,03	10	10	800
0,03	4	13	600	0,03	10	10	1080
0,03	4	13	700	0,03	10	12	360
0,03	4	13	800	0,03	10	12	400
0,03	4	13	1080	0,03	10	12	500
0,03	10	2	360	0,03	10	12	600
0,03	10	2	400	0,03	10	12	700
0,03	10	2	500	0,03	10	12	800
0,03	10	2	600	0,03	10	12	1080
0,03	10	2	700	0,03	10	13	360
0,03	10	2	800	0,03	10	13	400
0,03	10	2	1080	0,03	10	13	500
0,03	10	4	360	0,03	10	13	600
0,03	10	4	400	0,03	10	13	700
0,03	10	4	500	0,03	10	13	800
0,03	10	4	600	0,03	10	13	1080
0,03	10	4	700	0,03	10	20	400
0,03	10	4	800	0,03	10	20	500
0,03	10	4	1080	0,03	10	20	600
0,03	10	6	360	0,03	10	20	700
0,03	10	6	400	0,03	10	20	800
0,03	10	6	500	0,03	10	20	1080
0,03	10	6	600	0,03	14	2	360
0,03	10	6	700	0,03	14	2	400
0,03	10	6	800	0,03	14	2	500
0,03	10	6	1080	0,03	14	2	600

Table 6: OGDM hyperparameters optimisation grid.

p-Value	PCA	MinPts	Formation	p-Value	PCA	MinPts	Formation
0,03	14	2	700	0,03	14	13	360
0,03	14	2	800	0,03	14	13	400
0,03	14	2	1080	0,03	14	13	500
0,03	14	4	360	0,03	14	13	600
0,03	14	4	400	0,03	14	13	700
0,03	14	4	500	0,03	14	13	800
0,03	14	4	600	0,03	14	13	1080
0,03	14	4	700	0,03	14	20	400
0,03	14	4	800	0,03	14	20	500
0,03	14	4	1080	0,03	14	20	600
0,03	14	6	360	0,03	14	20	700
0,03	14	6	400	0,03	14	20	800
0,03	14	6	500	0,03	14	20	1080
0,03	14	6	600	0,03	20	2	360
0,03	14	6	700	0,03	20	2	400
0,03	14	6	800	0,03	20	2	500
0,03	14	6	1080	0,03	20	2	600
0,03	14	8	360	0,03	20	2	700
0,03	14	8	400	0,03	20	2	800
0,03	14	8	500	0,03	20	2	1080
0,03	14	8	600	0,03	20	4	360
0,03	14	8	700	0,03	20	4	400
0,03	14	8	800	0,03	20	4	500
0,03	14	8	1080	0,03	20	4	600
0,03	14	10	360	0,03	20	4	700
0,03	14	10	400	0,03	20	4	800
0,03	14	10	500	0,03	20	4	1080
0,03	14	10	600	0,03	20	6	360
0,03	14	10	700	0,03	20	6	400
0,03	14	10	800	0,03	20	6	500
0,03	14	10	1080	0,03	20	6	600
0,03	14	12	360	0,03	20	6	700
0,03	14	12	400	0,03	20	6	800
0,03	14	12	500	0,03	20	6	1080
0,03	14	12	600	0,03	20	8	360
0,03	14	12	700	0,03	20	8	400
0,03	14	12	800	0,03	20	8	500
0,03	14	12	1080	0,03	20	8	600

Table 7: OGDM hyperparameters optimisation grid.

p-Value	PCA	MinPts	Formation	p-Value	PCA	MinPts	Formation
0,03	20	8	700	0,05	2	4	600
0,03	20	8	800	0,05	2	4	1080
0,03	20	8	1080	0,05	2	6	800
0,03	20	10	360	0,05	2	8	700
0,03	20	10	400	0,05	2	8	800
0,03	20	10	500	0,05	2	10	360
0,03	20	10	600	0,05	2	12	700
0,03	20	10	700	0,05	2	20	1080
0,03	20	10	800	0,05	4	4	400
0,03	20	10	1080	0,05	4	4	700
0,03	20	12	360	0,05	4	4	1080
0,03	20	12	400	0,05	4	8	1080
0,03	20	12	500	0,05	4	10	800
0,03	20	12	600	0,05	4	12	400
0,03	20	12	700	0,05	4	20	500
0,03	20	12	800	0,05	10	2	1080
0,03	20	12	1080	0,05	10	4	360
0,03	20	13	500	0,05	10	4	700
0,03	20	13	600	0,05	10	8	700
0,03	20	13	700	0,05	10	12	360
0,03	20	13	800	0,05	10	12	1080
0,03	20	13	1080	0,05	10	13	500
0,03	20	20	800	0,05	10	13	700
0,05	1	2	360	0,05	10	20	400
0,05	1	2	400	0,05	14	2	500
0,05	1	2	500	0,05	14	4	600
0,05	1	2	600	0,05	14	4	1080
0,05	1	2	700	0,05	14	6	500
0,05	1	2	800	0,05	14	6	700
0,05	1	4	360	0,05	14	8	360
0,05	1	4	400	0,05	14	8	700
0,05	1	4	500	0,05	14	10	600
0,05	1	4	600	0,05	14	12	700
0,05	1	6	700	0,05	14	12	1080
0,05	1	8	600	0,05	14	13	1080
0,05	1	10	500	0,05	14	20	400
0,05	1	12	800	0,05	20	2	500
0,05	1	20	400	0,05	20	4	500

Table 8: OGDM hyperparameters optimisation grid.

p-Value	PCA	MinPts	Formation	p-Value	PCA	MinPts	Formation
0,05	20	6	700	0,08	4	13	1080
0,05	20	8	700	0,08	10	4	700
0,05	20	10	1080	0,08	10	6	400
0,05	20	12	800	0,08	10	6	500
0,05	20	13	1080	0,08	10	8	360
0,08	1	2	360	0,08	10	10	600
0,08	1	4	800	0,08	10	10	800
0,08	1	4	1080	0,08	10	12	400
0,08	1	6	360	0,08	10	12	500
0,08	1	6	700	0,08	10	13	500
0,08	1	10	360	0,08	10	20	500
0,08	1	10	700	0,08	10	20	1080
0,08	1	12	600	0,08	14	2	500
0,08	1	20	400	0,08	14	6	600
0,08	1	20	600	0,08	14	6	700
0,08	2	2	700	0,08	14	8	600
0,08	2	12	700	0,08	14	12	360
0,08	2	12	800	0,08	14	13	600
0,08	2	13	500	0,08	14	13	700
0,08	4	6	500	0,08	14	20	400
0,08	4	6	600	0,08	14	20	1080
0,08	4	6	700	0,08	20	2	400
0,08	4	6	800	0,08	20	6	700
0,08	4	8	400	0,08	20	8	400
0,08	4	10	500	0,08	20	8	1080
0,08	4	10	600	0,08	20	10	500
0,08	4	12	500	0,08	20	13	400
0,08	4	12	1080	0,08	20	13	600
0,08	4	13	360	0,08	20	20	360

Table 9: OGDM hyperparameters optimisation grid.

Abstract in lingua italiana

In finanza, il trading di coppia è una strategia che guadagna da due cespiti scommettendo sul ritorno alla media della serie numerica data dalla differenza dei loro valori. Le nuove tecnologie permettono al trading di coppia di cercare coppie redditizie attraverso test statistici al posto di somiglianze industriali. La rimozione di tali limitazioni, tuttavia, comporta un aumento esponenziale del numero di potenziali candidati: per superare tale problema, tecniche di apprendimento automatico, come PCA o l'analisi dei gruppi, vengono adottate per ridurre la ricerca delle coppie all'interno di sottoinsiemi di cespiti. In questa tesi, cominciamo con l'applicare, al contesto dei dati giornalieri di S&P 500, una strategia allo stato dell'arte di selezione delle coppie. Testiamo la procedure con gli iperparametri consigliati e valutiamo la correlazione tra una misura di coerenza dei gruppi da noi introdotta e le coppie selezionate. Successivamente, esaminiamo la sensibilità agli iperparametri del metodo ed ne identifichiamo un sottoinsieme di influenti. Infine, li ottimizziamo attraverso l'applicazione di Online Gradient Descent with Momentum: in un ambiente senza costi di transazione, il metodo ottiene buone prestazioni sia rispetto ai parametri statici suggeriti che alla media delle parametrizzazioni testate, mentre i guadagni diminuiscono quando i costi vengono considerati.

Parole chiave: ottimizzazione di iperparametri, trading di coppia, apprendimento automatico continuo, OGDM