Dealing with Transaction Costs in Portfolio Optimization: Online Gradient Descent with Momentum

Edoardo Vittori[1,2], Martino Bernasconi de Luca[1], Francesco Trovò[1], Marcello Restelli[1]
{edoardo.vittori,martino.bernasconideluca,francesco1.trovo,marcello.restelli}@polimi.it
[1]Politecnico di Milano, [2]Intesa Sanpaolo

ABSTRACT

Outperforming the markets through active investment strategies is one of the main challenges in finance. The random movements of assets and the unpredictability of catalysts make it hard to perform better than the average market, therefore, in such a competitive environment, methods designed to keep low transaction costs have a significant impact on the obtained wealth. This paper focuses on investing techniques to beat market returns through online portfolio optimization while controlling transaction costs. Such a framework differs from classical approaches as it assumes that the market has an adversarial behavior, which requires frequent portfolio rebalancing.

This paper analyses critically the known online learning literature dealing with transaction costs and proposes a novel algorithm, namely Online Gradient Descent with Momentum (OGDM), to control (theoretically and empirically) the costs. The existing algorithms designed for this setting are either (i) not providing theoretical guarantees, (ii) providing a bound to the total regret, conditionally on unrealistic assumptions or (iii) computationally not efficient. In this paper, we prove that OGDM has nice theoretical, empirical, and computational performances. We show that it has regret, considering costs, of the order $O(\sqrt{T})$, T being the investment horizon, and has $\Theta(M)$ per-step computational complexity, M being the number of assets. Furthermore, we show that this algorithm provides competitive gains when compared empirically with state-of-the-art online learning algorithms on a real-world dataset.

CCS CONCEPTS

• **Computing methodologies → Online learning settings**; **Sequential decision making**.

KEYWORDS

Online Portfolio Optimization, Transaction Costs

ACM Reference Format:
Edoardo Vittori[1,2], Martino Bernasconi de Luca[1], Francesco Trovò[1], Marcello Restelli[1]. 2020. Dealing with Transaction Costs in Portfolio Optimization: Online Gradient Descent with Momentum. In *ACM International Conference on AI in Finance (ICAIF '20), October 15–16, 2020, New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3383455.3422531

1 INTRODUCTION

The amount of assets managed by funds and private investors is currently more than 85 trillion USD, a quantity comparable to the global GDP.[1] This capital is invested with a variety of techniques which, simplifying and generalizing, are usually designed for one of the two steps of the investment process: the asset selection, *i.e.*, the choice of the most promising subset of available financial assets to invest in, and asset allocation, *i.e.*, deciding how much to invest in each asset [20]. In this paper, we focus on the latter, addressed in scientific literature as the *portfolio optimization problem.* In this context, the investment strategy rebalances the portfolio by buying and selling assets frequently, and this generates the so-called *transaction costs.* The scientific literature has extensively analyzed the problem of increasing gains by trying to predict the movement of market instruments, yet very few works are focusing on keeping transaction costs under control [7, 14].

While private funds do not disclose the techniques used to manage their assets, there exists an extensive scientific literature on portfolio optimization based on quantitative methods, started by [34, 37], which is also known as Modern Portfolio Theory (MPT). In MPT an investor has to optimize an utility function of the portfolio which accounts for the trade-off between the mean and the variance of the returns; this line of thinking continues to be applied in modern techniques such as Risk-averse RL [6]. Even if works in this area have been considering transaction costs [18], they require that the price evolution of assets satisfies strong statistical assumptions, which are hardly met in the real-world. A different approach is provided by capital growth theory [21, 26, 33], which originates from information theory and has now developed, in the machine learning community, under the name of Online Portfolio Optimization (OPO). The underlying theory assumes that the assets are controlled by an adversary who knows your investment strategy: no stochastic characterization is given to the market data and, therefore, the strategies designed in this framework do not require strong assumptions about market behavior. To react to the adversary, it is necessary to rebalance the portfolio at every time step and, thus, it is crucial to optimize the generated transaction costs. In this paper, we propose the use of a novel algorithm: Online Gradient Descent with Momentum (OGDM) for the OPO problem and show that it provides strong theoretical guarantees about the combined gains and costs incurred during the investment process.

In the OPO literature, two metrics have been used to measure the performance of an algorithm: regret on the wealth and per-round computational complexity. More specifically, in this framework, the wealth provided by a strategy is compared to the one of the best Constant Rebalanced Portfolio (CRP) [12], *i.e.*, a clairvoyant

[1]https://www.opalesque.com/671554/Global_assets_under_management_rose_to155.html.

strategy that keeps the best constant proportion of assets during the entire investment horizon. The regret on the wealth for an algorithm is defined as the difference between the growth rate of the best CRP and the rate obtained by the algorithm and represents the loss suffered by the algorithm due to the lack of information. Keeping this quantity limited is of paramount importance; indeed, the algorithms that show sub-linear dependence of the regret w.r.t. the investment horizon T are called universal [2]. Conversely, the complexity of an algorithm is evaluated by the number of operations needed to compute the strategy used to rebalance the portfolio. Generally, there exists a trade-off between the regret incurred by an algorithm and its computational complexity. For instance, [12] proposed the Universal Portfolio (UP) algorithm, which achieves a regret on the wealth of $O(\log T)$ and requires a per-step computational cost of $\Theta(T^M)$, where M is the number of assets used in the portfolio, while [23] proposed the Exponential Gradient (EG) algorithm which has a regret on the wealth of $O(\sqrt{T})$, with a per-step computational cost of $\Theta(M)$.

The regret analyzed by these algorithms assumes that rebalancing the portfolio does not generate any transaction costs and, thus, they do not guarantee sub-linear regret in the realistic scenario in which costs are taken into account. There exist a wide variety of heuristic methods that tried to overcome this problem *e.g.*, [29, 40], but they do not provide a theoretical assurance on total regret, *i.e.*, the regret that includes transaction costs. To the best of our knowledge, there are only two studies that analyze total regret: U_CP [7], and Online Lazy Updates (OLU) [14]. Both works provide theoretical guarantees in settings in which the costs are proportional to the reallocated assets. More specifically, the former modifies the UP algorithm to include costs and has a total regret of $O(\log T)$, but with a prohibitive computational cost, similarly to UP, and the latter provides a $O(\sqrt{T})$ total regret but requires the unrealistic assumption that the costs decreases during the investment horizon. In this paper, for the first time, we propose: (i) the use of a novel online convex optimization algorithm: the OGDM algorithm, a modified version of Online Gradient Descent (OGD) [41], to deal with transaction costs in the OPO framework; (ii) a theoretical analysis of the total regret of OGDM in the presence of costs, assuming that they are proportional to the reallocated assets, obtaining a total regret order of $O(\sqrt{T})$; (iii) an in-depth analysis of the algorithms dealing with transaction costs, *i.e.*, OLU and U_CP, shedding light on their limits; (iv) an empirical comparison between the performance of OGDM and state-of-the-art OPO algorithms, to guide the choice of the best method to use on the OPO problem depending on the magnitude of the costs.

2 RELATED WORKS

Many algorithms from the online learning literature have been applied to the OPO framework since they provide both strong theoretical guarantees in the adversarial setting and good empirical results. The most interesting ones have been described in detail by Li and Hoi [27] and by Dochow [15]. U_CP and OLU are the algorithms closest to our work, and will be discussed in depth in Sections 4.3 and 4.4.

Most notably, the Online Newton Step (ONS) [2, 22] algorithm has been shown to provide good performance in terms of regret on the wealth when empirically tested, as well as feasible computational complexity. There are also heuristic algorithms designed to solve the OPO problem, *e.g.*, Anticor [8], PAMR [30], OLMAR [28], and MRTC [40], which outperform the algorithms described above in terms of empirical performance. Remarkably, none of the above algorithms provide guarantees on the total regret.

In addition, Li et al. in [29] extend both traditional and heuristic algorithms to include an additional term to the optimization function to handle transaction costs, but only provide an empirical analysis and no type of regret guarantees. Notably, what is presented by Ito et al. [24] is related to the objective of controlling transaction costs. Indeed, the assumption that the portfolio is composed of a small set of assets indirectly addresses such a problem. However, the authors do not present theoretical guarantees on the potential costs incurred by such an algorithm.

The problem of dealing with transaction costs has also been tackled in sequential decision-making settings similar to the OPO one, *i.e.*, in the expert and bandit learning field [10, 39] and the Metrical Task Systems (MTS) literature [19, 31, 32]. In the expert and bandit literature the problem of learning with costs has been analyzed either purely theoretically by Cesa-Bianchi et al. in [10] or under the bandit feedback by Trovò et al. in [39], which is not a realistic feedback in our financial application. In the MTS literature [19, 31, 32], the notion of regret has been extended to include the cost of changing the prediction of the algorithm over time, but the framework allows the learner to know the future realizations of the environment, *i.e.*, the price of the assets for the next day, which is unreasonable in our application.

Finally, the algorithm we propose to deal with online optimization is inspired by the effectiveness of the momentum technique by Polyak in [35]. The momentum term smooths the estimation of the gradient and it has been used successfully in the optimization of complex non-linear functions, such as Neural Networks [36, 38].

3 PROBLEM FORMULATION

This section provides a formal description of the OPO problem and defines the transaction costs. The framework consists of a sequential decision problem in which, at each (discrete) round $t \in \{1, \dots, T\}$ over an investment horizon $T \in \mathbb{N}$, an investor makes a portfolio allocation over a set of $M \in \mathbb{N}$ different assets by choosing a vector $\mathbf{x}_t := (x_{1,t}, \dots, x_{M,t})$, with $\mathbf{x}_t \in \Delta_{M-1}$, Δ_{M-1} being the $(M-1)$-simplex in $\mathbb{R}^M$. Each element $x_{j,t}$ of $\mathbf{x}_t$ is the proportion of asset j contained in the portfolio at round t. The sequence $\mathbf{x}_{1:T} := (\mathbf{x}_1, \dots, \mathbf{x}_T)$ represents the investment strategy over T rounds.[2] Let us define the price relatives, $\mathbf{r}_t := (r_{1,t}, \dots, r_{M,t})$, *i.e.*, $r_{j,t} = \frac{p_{j,t+1}}{p_{j,t}}$, where $p_{j,t}$ is the price of asset j at round t, and the price relatives sequence as $\mathbf{r}_{1:T} := (\mathbf{r}_1, \dots, \mathbf{r}_T)$. As is commonly done in the portfolio allocation literature [2], we assume that the price of the assets does not change too much during two consecutive rounds, or, formally:

Assumption 1. *There exist two finite constants $\epsilon_l, \epsilon_u \in \mathbb{R}^+$ s.t. the price relatives $r_{j,t} \in [\epsilon_l, \epsilon_u]$, with $0 < \epsilon_l \leq \epsilon_u < +\infty$, for each round $t \in \{1, \dots, T\}$ and each asset $j \in \{1, \dots, M\}$.*

[2]The time duration of a step is discretionary and may be defined as a few hours, days, weeks, or months. Commonly, investors choose daily discretization when using OPO techniques.

Regret on the Wealth. The cumulative wealth $W_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T})$ at round T, for an investment strategy $\mathbf{x}_{1:T}$ and a sequence of price relatives $\mathbf{r}_{1:T}$, is defined as:

$$W_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T}) := \prod_{t=1}^{T} \langle \mathbf{x}_t, \mathbf{r}_t \rangle, \tag{1}$$

where we denote with $\langle \cdot, \cdot \rangle$ the scalar product. We define the loss incurred at round t for choosing a portfolio $\mathbf{x}_t$ as:

$$f_t(\mathbf{x}_t) := -\log(\langle \mathbf{x}_t, \mathbf{r}_t \rangle).$$

For an algorithm $\mathfrak{U}$ which generates an investment strategy $\mathbf{x}_{1:T}$, the *regret on the wealth* $R_T(\mathfrak{U})$ at round T is the difference between the cumulative losses of the best CRP and those of the algorithm $\mathfrak{U}$, formally:

$$\begin{aligned} R_T(\mathfrak{U}) &= \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^*) \\ &= \log(W_T(\mathbf{x}^*_{1:T}, \mathbf{r}_{1:T})) - \log(W_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T})), \end{aligned}$$

where $\mathbf{x}^*_{1:T}$ is a constant investment strategy using the best CRP computed on the sequence $\mathbf{r}_{1:T}$, *i.e.*, for all $t \in \{1, \dots, T\}$ the investment strategy is $\mathbf{x}^* = \mathbf{x}^*_t = \arg\sup_{\mathbf{x} \in \Delta_{M-1}} \prod_{i=1}^{T} \langle \mathbf{x}, \mathbf{r}_i \rangle$.

Regret on the Costs. In real-world scenarios, transaction costs may vary depending on multiple aspects [5], *e.g.*, the liquidity of the financial instrument, or the market impact of the trade. In this work, following the approach previously used in the OPO literature [7], we use an approximation of the real transaction costs, considering them proportional to the difference in portfolio allocation over two consecutive rounds. Formally, the transaction costs at round t are implicitly determined by the solution of the following equation (known in finance as turnover):

$$\alpha_t = 1 - \gamma ||\mathbf{x}'_{t-1} - \mathbf{x}_t \alpha_t||_1, \tag{2}$$

where α_t is the proportion of residual wealth after the transaction fees, γ is the transaction rate, which is equal for buying and selling and fixed throughout the investment horizon, and $\mathbf{x'}_{t-1} = \frac{\mathbf{x}_{t-1} \otimes \mathbf{r}_{t-1}}{\langle \mathbf{x}_{t-1}, \mathbf{r}_{t-1} \rangle}$ is the portfolio composition after the market movement $\mathbf{r}_{t-1}$.[3] The wealth considering transaction costs becomes:

$$\tilde{W}_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T}) = \prod_{t=1}^{T} \langle \mathbf{x}_t, \mathbf{r}_t \alpha_t \rangle, \tag{3}$$

where α_t is the solution of Equation (2). If we assume that the price relatives $\mathbf{r}_t$ are small, we have that $\mathbf{x}'_{t-1} \approx \mathbf{x}_{t-1}$ and $\alpha_t \mathbf{x}_t \approx \mathbf{x}_t$, and, therefore, the proportion of remaining wealth becomes $\alpha_t = 1 - \gamma ||\mathbf{x}_{t-1} - \mathbf{x}_t||_1$. Using the above approximations, the wealth $\tilde{W}_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T})$ can be transformed as follows:

$$\log(\tilde{W}_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T})) = \log\left(\prod_{t=1}^{T} \langle \mathbf{x}_t, \mathbf{r}_t \alpha_t \rangle\right) \tag{4}$$

$$\approx \log(W_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T})) + \log\left(\prod_{t=1}^{T} \alpha_t\right) \tag{5}$$

$$\approx \log(W_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T})) - \sum_{t=1}^{T} \gamma ||\mathbf{x}_t - \mathbf{x}_{t-1}||_1, \tag{6}$$

[3]With $\mathbf{a} \otimes \mathbf{b}$ we denote the element-wise product between the two vectors $\mathbf{a}$ and $\mathbf{b}$.

Algorithm 1 OGDM in OPO with Transaction Costs

```
Require: learning rate sequence $\{\eta_1, \dots, \eta_T\}$, momentum parameter sequence $\{\lambda_1, \dots, \lambda_T\}$
1: Set $\mathbf{x}_1 \leftarrow \frac{1}{M}\mathbf{1}$
2: for $t \in \{1, \dots, T\}$ do
3:     Select $\mathbf{x}_{t+1} \leftarrow \Pi_{\Delta_{M-1}}\left(\mathbf{x}_t + \eta_t \frac{\mathbf{r}_t}{\langle \mathbf{r}_t, \mathbf{x}_t \rangle} - \frac{\lambda_t}{2}(\mathbf{x}_t - \mathbf{x}_{t-1})\right)$
4:     Observe $\mathbf{r}_{t+1}$ from the market
5:     Get wealth $\log(\langle \mathbf{r}_{t+1}, \mathbf{x}_{t+1} \rangle) - \gamma ||\mathbf{x}_{t+1} - \mathbf{x}_t||_1$
6: end for
```

where we used a first term expansion $\log(1 - y) \sim -y$ to get Equation (6), given that the transaction rate is $\gamma \ll 1$. Using the second term of Equation (6) we define the proportional costs $C_T(\mathfrak{U})$ incurred by an algorithm $\mathfrak{U}$ during the investment horizon of T, as is done in [14]:

$$C_T(\mathfrak{U}) := \gamma \sum_{t=1}^{T-1} ||\mathbf{x}_{t+1} - \mathbf{x}_t||_1. \tag{7}$$

The *total regret* $R^C_T(\mathfrak{U})$, *i.e.*, the regret computed considering the transaction costs, is defined as:

$$R^C_T(\mathfrak{U}) = R_T(\mathfrak{U}) + C_T(\mathfrak{U}).$$

The financial interpretation is that *regret on the costs* consists in the (approximated) turnover gap and it decreases the final wealth of the investor. While *total regret* is the combination of the regret coming from the suboptimal choice of the portfolio and the one from the turnover, *i.e.*, the wealth gap considering transaction costs. Notice that the best CRP investment strategy $\mathbf{x}^*_{1:T}$ generates no costs under this model, therefore the costs $C_T(\mathfrak{U})$ also represent the *regret on the costs* of the algorithm $\mathfrak{U}$ due to the transaction fees paid over the investment horizon T.

4 OGDM FOR PORTFOLIO OPTIMIZATION

This section describes the new OGDM algorithm and how it can be tailored to the OPO framework. We also provide a theoretical analysis on the OGDM regret in the presence of transaction costs and compare it with the state of the art algorithms for this problem.

4.1 OGD with Momentum

The definition of the OGDM update rule for a generic convex loss function $f_t(\mathbf{x}_t)$ over a generic convex set X is the following:

$$\mathbf{x}_{t+1} = \Pi_X\left(\mathbf{x}_t - \eta_t \nabla f_t(\mathbf{x}_t) - \frac{\lambda_t}{2}(\mathbf{x}_t - \mathbf{x}_{t-1})\right), \tag{8}$$

where $\Pi_X(y) := \arg\inf_{x \in X} ||y - x||^2_2$ is the standard projection of the vector y onto X, $\eta_t > 0$ is the learning rate at round t, λ_t is a parameter controlling the momentum influence at round t, and $\nabla(\cdot)$ denotes the gradient operator. Recalling that in the OPO framework the function to be minimized is the loss $f_t(\mathbf{x}_t) = -\log(\langle \mathbf{x}_t, \mathbf{r}_t \rangle)$, the portfolio update rule becomes:

$$\mathbf{x}_{t+1} = \Pi_{\Delta_{M-1}}\left(\mathbf{x}_t + \eta_t \frac{\mathbf{r}_t}{\langle \mathbf{x}_t, \mathbf{r}_t \rangle} - \frac{\lambda_t}{2}(\mathbf{x}_t - \mathbf{x}_{t-1})\right). \tag{9}$$

The pseudo-code corresponding to the OGDM algorithm in the OPO framework, including transaction costs, is presented in Algorithm 1.

The algorithm starts with a portfolio $\mathbf{x}_1$ of weights equally allocated among the M available assets (Line 1). Then, for each round $t \in \{1, \dots, T\}$ it rebalances the assets according to Equation (9) (Line 3), it observes the market outcomes $\mathbf{r}_{t+1}$ (Line 4), and gains a per-round wealth, including costs, of $\log(\langle \mathbf{r}_{t+1}, \mathbf{x}_{t+1}\rangle) - \gamma||\mathbf{x}_{t+1} - \mathbf{x}_t||_1$ (Line 5).

4.2 Regret Analysis

We now present the main result on the total regret of the OGDM algorithm.

Theorem 1. *The OGDM algorithm with* $\eta_t = \frac{K_\eta}{\sqrt{t}}$*, and* $\lambda_t = \frac{K_\lambda}{t}$*, for each value of* $K_\eta, K_\lambda \in \mathbb{R}^+$*, has a total regret of:*

$$R_T^C \le \left[\frac{D^2}{K_\eta}\left(\frac{1}{2} + K_\lambda\right) + K_\eta \tilde{G}\left(2\gamma\sqrt{M} + \tilde{G}\right)\right]\sqrt{T}, \quad (10)$$

where $D = \sup_{\mathbf{x},\mathbf{y}\in X} ||\mathbf{x} - \mathbf{y}||_2$*, and* $\tilde{G} = \sup_{\mathbf{x}\in X} ||\nabla f_t(\mathbf{x})||_2 + \frac{DK_\lambda}{2K_\eta}$.

Proof. Recall that for a generic loss function $f_t : X \to \mathbb{R}$ and a convex set X the OGDM algorithm has the update rule in Equation (8). This formulation can be rewritten as:

$$\mathbf{x}_{t+1} = \Pi_X\left(\mathbf{x}_t - \eta_t \nabla \tilde{f}_t(\mathbf{x}_t)\right), \quad (11)$$

by defining: $\tilde{f}_t(\mathbf{x}) = f_t(\mathbf{x}) + \frac{\beta_t}{2}||\mathbf{x} - \mathbf{x}_{t-1}||_2^2$, with $\beta_t = \frac{\lambda_t}{\eta_t}$. Note that, by the triangle inequality $||\nabla \tilde{f}(\mathbf{x}_t)||_2 \le \tilde{G}$.

Before presenting the main result, we recall that, from [41], given the update in Equation (11) we have:

$$\begin{aligned}||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2 &= ||\Pi_X(\mathbf{x}_t - \eta_t\nabla\tilde{f}_t(\mathbf{x}_t)) - \mathbf{x}^*||_2^2\\ &\le ||\mathbf{x}_t - \mathbf{x}^*||_2^2 - 2\eta_t\langle \mathbf{x}_t - \mathbf{x}^*, \nabla\tilde{f}_t(\mathbf{x}_t)\rangle + \eta_t^2||\nabla\tilde{f}_t(\mathbf{x}_t)||_2^2,\end{aligned}$$

where we used the fact that the projection operator $\Pi_{\Delta_X}(\cdot)$ is non-expansive. Rearranging the terms, we have:

$$\langle \mathbf{x}_t - \mathbf{x}^*, \nabla\tilde{f}_t(\mathbf{x}_t)\rangle \le \frac{1}{2\eta_t}\left(||\mathbf{x}_t - \mathbf{x}^*||_2^2 - ||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2\right) + \frac{\eta_t}{2}\tilde{G}^2. \quad (12)$$

Using the above inequality, the total regret $R_T^C(OGDM)$ of the OGDM algorithm is bounded as follows:

$$\begin{aligned}
R_T^C(OGDM) &= \sum_{t=1}^T f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*) + \gamma\sum_{t=1}^T ||\mathbf{x}_t - \mathbf{x}_{t-1}||_1 \quad (13)\\
&= \sum_{t=1}^T \tilde{f}_t(\mathbf{x}_t) - \tilde{f}_t(\mathbf{x}^*) - \sum_{t=2}^T \frac{\beta_t}{2}\left(||\mathbf{x}_t - \mathbf{x}_{t-1}||_2^2 - ||\mathbf{x}^* - \mathbf{x}_{t-1}||_2^2\right)\\
&\quad + \gamma\sum_{t=1}^T ||\mathbf{x}_t - \mathbf{x}_{t-1}||_1 \quad (14)\\
&\le \sum_{t=1}^T \langle \mathbf{x}_t - \mathbf{x}^*, \nabla\tilde{f}_t(\mathbf{x}_t)\rangle + \sum_{t=1}^T \frac{\beta_t}{2}||\mathbf{x}^* - \mathbf{x}_{t-1}||_2^2\\
&\quad + \gamma\sum_{t=1}^T \sqrt{M}\eta_t||\nabla\tilde{f}_t(\mathbf{x}_t)||_2 \quad (15)\\
&\le \sum_{t=1}^T \frac{1}{2\eta_t}\left(||\mathbf{x}_t - \mathbf{x}^*||_2^2 - ||\mathbf{x}_{t+1} - \mathbf{x}^*||_2^2\right) + \sum_{t=1}^T \frac{\eta_t}{2}\tilde{G}^2\\
&\quad + \sum_{t=1}^T \frac{\beta_t}{2}||\mathbf{x}^* - \mathbf{x}_{t-1}||_2^2 + \gamma\sum_{t=1}^T \sqrt{M}\eta_t||\nabla\tilde{f}_t(\mathbf{x}_t)||_2 \quad (16)\\
&\le \frac{D^2}{2\eta_1} + \frac{D^2}{2}\sum_{t=2}^T\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right) + \sum_{t=1}^T \frac{\eta_t}{2}\tilde{G}^2\\
&\quad + \sum_{t=1}^T \frac{\beta_t}{2}||\mathbf{x}^* - \mathbf{x}_{t-1}||_2^2 + \gamma\sum_{t=1}^T \sqrt{M}\eta_t||\nabla\tilde{f}_t(\mathbf{x}_t)||_2 \quad (17)\\
&\le \frac{D^2}{2\eta_T} + \sum_{t=1}^T \frac{\eta_t}{2}\tilde{G}^2 + \sum_{t=1}^T \frac{\beta_t}{2}D^2 + \gamma\sqrt{M}\tilde{G}\sum_{t=1}^T \eta_t, \quad (18)
\end{aligned}$$

where we dropped the negative term and used the convexity of $\tilde{f}(\cdot)$ to derive Equation (15), and used the result in Equation (12) to derive Equation (16).

Finally, substituting $\beta_t = \frac{\lambda_t}{\eta_t}$, $\lambda_t = \frac{K_\lambda}{t}$, $\eta_t = \frac{K_\eta}{\sqrt{t}}$ in Equation (18), and using the fact that $\sum_{t=1}^T \frac{1}{\sqrt{t}} \le 2\sqrt{T}$ concludes the proof.

□

If we assume that the price of the assets does not change too much during two consecutive rounds, as asserted by Assumption 1, we have:

Corollary 1. *If Assumption 1 holds, the OGDM algorithm* $\eta_t = \frac{K_\eta}{\sqrt{t}}$*, and* $\lambda_t = \frac{K_\lambda}{t}$*, for each* $K_\eta > 0$ *and* $K_\lambda > 0$*, has total regret of:*

$$\begin{aligned}R_T^C(OGDM) \le \sqrt{T}\Bigg[&\frac{K_\lambda^2 + 4K_\lambda + 2}{2K_\eta} + K_\eta M\frac{\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)\\ &+ \sqrt{2}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)K_\lambda\sqrt{M}\Bigg].\end{aligned}$$

Using the previous bound we can optimize the regret bound w.r.t. the parameters K_η and K_λ as follows:

Corollary 2. *If Assumption 1 holds, the OGDM algorithm with* $\eta_t = \frac{1}{\sqrt{t}}\left[\frac{M\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)\right]^{-1/2}$ *and* $\lambda_t = 0$ *has a total regret of:*

$$R_T^C(OGDM) \le 2\sqrt{\frac{M\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{\epsilon_l} + 2\gamma\right)T}. \quad (19)$$

Notice that the OGDM with the previous choice of the bound corresponds to the OGD algorithm with a learning rate of $\eta_t = K_\eta/\sqrt{t}$. Indeed, this is consistent with the fact that $R_T(OGD) = O(\sqrt{T})$ for a generic convex function $f_t(x)$, as shown by Belmega et al. in [4]. Even if this is the choice that minimizes the upper bound on the total regret, it might be suboptimal in practice. In the experimental section we will analyze the empirical performance of choices of $K_\lambda \ne 0$.

Finally, knowing the time horizon T in advance from the starting of the investment period we have:

Corollary 3. *If Assumption 1 holds, the OGDM algorithm with* $\eta_t = \frac{1}{\sqrt{T}}\left[\frac{M\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{2\epsilon_l} + \gamma\right)\right]^{-1/2}$ *and* $\lambda_t = 0$ *has a total regret of:*

$$R_T^C(OGDM) \le 2\sqrt{\frac{M\epsilon_u}{\epsilon_l}\left(\frac{\epsilon_u}{2\epsilon_l} + \gamma\right)T}.$$

Table 1: Theoretical results, in terms of regret and computational complexity, for the analysed algorithms.

	OGDM	U_CP	OLU	ONS
R_T	$O(\sqrt{T})$	$O(\log T)$	$O(\sqrt{T})$	$O(\log T)$
R_T^C	$O(\sqrt{T})$	$O(\log T)$	$O(T)$	-
Complexity	$\Theta(M)$	$\Theta(T^M)$	$\Theta(M)$	$\Theta(M^2)$

This result provides a slightly improved constant in the bound over the *any-time* bound given by Corollary 2. In the next sections, we compare the theoretical guarantees of OGDM in terms of computational complexity and total regret with OLU and U_CP, the only algorithms that provide upper bounds to total regret.

4.3 Discussion on the Per-round Computational Complexity

In this section we explore the theoretical results of OGDM in comparison to the existing literature w.r.t. regret bounds and computational complexity, summarized in Table 1.[4]

Regarding the computational complexity of the OGDM algorithm, at each round t, it evaluates a scalar product, a division and two vector subtractions (see Line 3, Algorithm 1), which require a number of operations linearly proportional to the number of assets M. It also performs a projection onto the simplex, which can be computed in linear time with the number of assets M (see Duchi et al. [16] for details). Therefore, the total expected computational cost per round is $\Theta(M)$. Note that, since the learning rate η_t is decreasing over time, the projection operation is less likely to be required as we proceed with the investment process, decreasing the per-step computational effort. Conversely, the technique used in literature to implement the U_CP strategy requires a number of operations per round of $\Theta(T^M)$ [25], which does not scale well for large horizons T, or settings where the number of assets M is large.

In [14], Das et al. propose to use the Alternating Direction Method of Multipliers (ADMM) [9] to implement the update rule of OLU. Although in terms of computational complexity it has the same properties of OGDM, the OLU algorithm is more computationally costly (in terms of constants) than the OGDM update, since it consists of solving a problem with linear complexity in M multiple times until ADMM converges, but still provides a feasible solution in terms of computational effort. To conclude, U_CP is a solution suitable only for problems with a small number of assets M and a short investment horizon T; conversely, OGDM and OLU can handle data streams that come at higher frequencies, *e.g.*, the ones required by some specific financial applications [1].

4.4 Discussion on the Total Regret Bounds

As discussed above, the OLU algorithm is the only algorithm competing with OGDM, in terms of per-round computational complexity. OLU can be interpreted as an instance of Composite Objective Mirror Descent (COMID) [17], whose update is the following:

$$\mathbf{x}_{t+1} = \arg\inf_{\mathbf{x}\in\Delta_{M-1}} \left\{ \eta \langle \nabla f_t(\mathbf{x}_t), \mathbf{x} \rangle + \eta\, r(\mathbf{x}) + d_\psi(\mathbf{x}, \mathbf{x}_t) \right\},$$

where $r(\mathbf{x})$ is a regularization term of the loss function $f_t(\mathbf{x})$, and $d_\psi(\mathbf{x}, \mathbf{y})$ is a Bregman divergence [3] generated by the convex function $\psi(\mathbf{x})$. More specifically, the OLU algorithm uses as regularizer $r(\mathbf{x}) := ||\mathbf{x} - \mathbf{x}_t||_1$ and divergence $d_\psi(\mathbf{x}, \mathbf{y}) := \frac{1}{2}||\mathbf{x} - \mathbf{y}||_2^2$.

Assuming to know *a priori* the time horizon T and under Assumption 1, the authors of OLU provide the following guarantee:

Theorem 2 (Total Regret of OLU [13]). *If Assumption 1 holds, the OLU algorithm with $\eta = \frac{K}{\sqrt{T}}$, $\forall K \in \mathbb{R}^+$ has a total regret of:*

$$R_T^C(OLU) \leq \left(\frac{1}{K} + \frac{MK\epsilon_u^2}{2\epsilon_l^2} \right) \sqrt{T} + 2\gamma T. \tag{20}$$

Notice that the OLU algorithm achieves a regret of $O(\sqrt{T})$ only if the transaction rate $\gamma \propto \frac{1}{\sqrt{T}}$, *i.e.*, if the transaction rate decreases over time. We can observe that the first term of the r.h.s. of Equation (20) corresponds to the regret on the wealth. Instead, if we focus on the second term of the r.h.s. of Equation (20) and we assume that γ is constant over the investment horizon T, we would have a total regret of the order of $O(T)$ for the OLU algorithm. This does not happen to OGDM, which, under these assumptions, provides a total regret of the order of $O(\sqrt{T})$. Conversely, if we assume $\gamma \propto \frac{1}{\sqrt{T}}$ as in [14], the last term in Equation (19) would have constant regret on the costs, *i.e.*, $C_T(OGDM) \leq \frac{2\epsilon_u M}{\epsilon_l} = O(1)$, compared to an order of $O(\sqrt{T})$ obtained by OLU, which makes OGDM strictly better than OLU in terms of total regret bound.

5 EXPERIMENTS

In this section we analyze the empirical performance of the OGDM algorithm and the two algorithms from the OPO literature that provide guarantees on total regret: U_CP [7], and OLU [14].[5] Furthermore, we compare OGDM with OGD, to evaluate the empirical improvement provided by the momentum, and with ONS [2], which has theoretical guarantees and is known to provide the best empirical results for the regret on the wealth $R_T(\mathfrak{U})$.

To compare the algorithms, we used four different datasets, summarized in Table 2. More specifically, we ran experiments on the NYSE(O), SP500, and TSE datasets, which are well-known benchmarks used in several research papers on portfolio optimization providing the daily prices for a fixed set of asset.[6] The experiments on each of the above datasets consist in the execution of the analyzed algorithms on portfolios selected by randomly drawing 5 different assets from the original datasets. The fourth one, namely the *Corona dataset*, was designed using data coming from the recent *CoVid-19 crisis* period, to explicitly analyze the behavior of the OPO algorithms in times of high volatility. Table 3 describes the assets and time period included in the Corona dataset.

For comparison purposes, we set the same η_t for OGD and OGDM as prescribed by Corollary 2, with $\epsilon_l = 0.8$ and $\epsilon_u = 1.2$, for which Assumption 1 holds for all the datasets. Instead, to tune the K_λ for the sequence λ_t in Corollary 1 for OGDM and the parameters required by the other algorithms (OLU, ONS, and U_CP),

[4]We report the regret bounds and complexity also for the ONS algorithm for sake of completeness.

[5]We used a naïve version of U_CP since the classic implementation would have taken an unfeasible amount of time for the experiments. More specifically, we discretized the simplex with 10^4 points and used the corresponding CRPs to approximate the integrals used by U_CP.

[6]These datasets are available at http://www.cs.technion.ac.il/~rani/portfolios/.

Table 2: Datasets used in the experimental campaign.

Datasets				
Name	Market	Year Span	Days	Assets
NYSE(O)	New York Stock Exchange	1962 - 1984	5651	36
TSE	Toronto Stock Exchange	1994 - 1998	1258	88
SP500	Standard Poor's 500	1998 - 2003	1276	25
Corona	Global	2019 - 2020	280	4

Table 3: Assets included in the Corona Dataset.

Corona Dataset (03/29/2019 - 05/08/2020)		
Ticker	Description	Market Category
SPY	SPDR S&P 500 ETF Trust	Equity
BNDX	Vanguard Bond Index Fund ETF	Fixed Income
DAX	Global X DAX Germany ETF	Equity
VIX	CBOE Volatility Index	Derivatives

we divided the datasets into a validation and testing set of equal size, we optimized the parameters on the former one and evaluated the performance of the algorithms on the latter one. All algorithms have been initialized with $\mathbf{x}_1 = \frac{1}{M}\mathbf{1}$.[7]

As performance indexes to compare the algorithms we used:

- the *wealth with costs*
$$W_T^C(\mathbf{x}_{1:T}, \mathbf{r}_{1:T}) = W_T(\mathbf{x}_{1:T}, \mathbf{r}_{1:T}) - \gamma \sum_{t=1}^{T} ||\mathbf{x}_t - \mathbf{x}_{t-1}||_1;$$
- the *Annual Percentage Yield* (APY), assuming 250 working days per year and one update per day
$$A(T) = W_T^C(\mathbf{x}_{1:T}, \mathbf{r}_{1:T})^{250/T} - 1;$$
- the *average variation of the portfolio per-round*:
$$V_t(\mathfrak{U}) := \frac{C_t(\mathfrak{U})}{\gamma t}.$$

Notice that above quantity $V_t(\mathfrak{U})$ is defined so that it is independent from the parameter γ.

5.1 Experiments without transaction costs

In this first experiment, we test the performance of the OGDM algorithm in a situation where transaction costs are negligible ($\gamma = 0$). More specifically, we present two different experiments that allow us to draw conclusions on the behavior of the different algorithms in two radically different market scenarios.

Results. Figure 1a and 1b show the evolution of the total wealth $W_t^C(\mathfrak{U})$ of the different algorithms over the investment horizon in a specific run, on the Corona dataset and on the NYSE(O) dataset, respectively. In these experiments OGDM obtains a cumulative wealth larger than any other algorithm analyzed, suggesting that it can obtain the best performance even in the absence of costs.

Comparing the two figures we notice that, while in Figure 1b all tested algorithms have a generally positive trend in terms of wealth $W_t^C(\mathfrak{U})$ over time, in Figure 1a OGDM, OGD, and U$_C$P are able to provide significantly better performance in the last part of the time horizon ($220 \leq t \leq 250$). This is due to the presence of the VIX assets present in the Corona dataset, which had an impressive gain during the initial phased of the CoVid19 spread.

Instead, if we look at the periods of general market stability (the entire time horizon of Figure 1a and the period in $1 \leq t \leq 220$ of Figure 1a), there is no clear outstanding algorithm among the ones we analyzed. It is curious to notice how OGDM, OGD, and U$_C$P show similar behaviors, while OLU and ONS are different from the first three, but similar to each other. Overall, the OGDM algorithm is capable of obtaining a performance comparable to the ones present in the literature in both settings.

5.2 Experiments with transaction costs

In the second set of experiments, we run the algorithms on the NYSE(O), SP500, and TSE datasets and evaluate the performance of the algorithms in settings with different values of the transaction cost rate $\gamma \in \{0, 0.0005, 0.001, 0.003, 0.006, 0.01, 0.02, 0.04\}$. We evaluated the different algorithms in terms of $APY(T)$ and average variation of the portfolio per round $V_t(\mathfrak{U})$.[8] The 95% confidence intervals for the analyzed quantities have been computed with statistical bootstrapping and appear as semi-transparent areas.

Results. Figure 1c provides the results of a setting with $\gamma = 0.01$ for the same set of assets shown in Figure 1c. These examples show that OGDM is able to obtain essentially the same performance as the framework without costs even with large transaction costs, while all other algorithms degrade noticeably.

In Figure 2, we present the results for the average APY on three datasets: NYSE(O), SP500, and TSE. Without transaction costs ($\gamma = 0$), all the analyzed algorithms obtain consistently an APY between around 10% and 15%. In this setting, ONS is the algorithm with the largest average APY, but as we increase the transaction costs rate, it is the algorithm that suffers the most, along with OLU. The performance of U$_C$P deteriorates as the transaction cost rate increases, but its loss is limited compared to the ones of ONS and OLU, always providing a wealth $W_t^C(U_CP) > 0$. OGDM and OGD outperform the other algorithms when the transaction rate is $\gamma \geq 0.01$. While showing similar behavior for the SP500 dataset, in the other two experiments the OGDM is able to outperform OGD obtaining almost the same APY as in the setting with no costs, even when $\gamma = 0.04$, *i.e.*, transaction costs are 4%. Conversely, OGD, not using an explicit term for costs in its optimization procedure, has a slight decrease in terms of wealth as costs increase.

In Figure 3, we present the results in terms of average variation of the portfolio per round $V_t(\mathfrak{U})$ for three datasets: NYSE(O), SP500, and TSE. The worst performing algorithm is OLU since, as expected from the theory (see Section 4.4), its variation of the portfolio per round, which is proportional to the transaction costs, is approximately constant. The second worst performer is ONS which, even though starting with a larger variation than OLU and not having any theoretical guarantee, is likely to have a sub-linear variation per round that decreases over time, but at a slower pace than the remaining three algorithms. Comparing OGDM, OGD, and U$_C$P, we can see that they all have a variation of the portfolio per round that decreases over time. Out of these three, U$_C$P is the one with

[7]The final version of the paper will provide a link to the code implementing the described experimental setup, not provided here for anonymization purposes.

[8]We also run experiments with transaction costs on the Corona dataset, and the results are not presented here as they are in line with the ones presented for the other datasets.

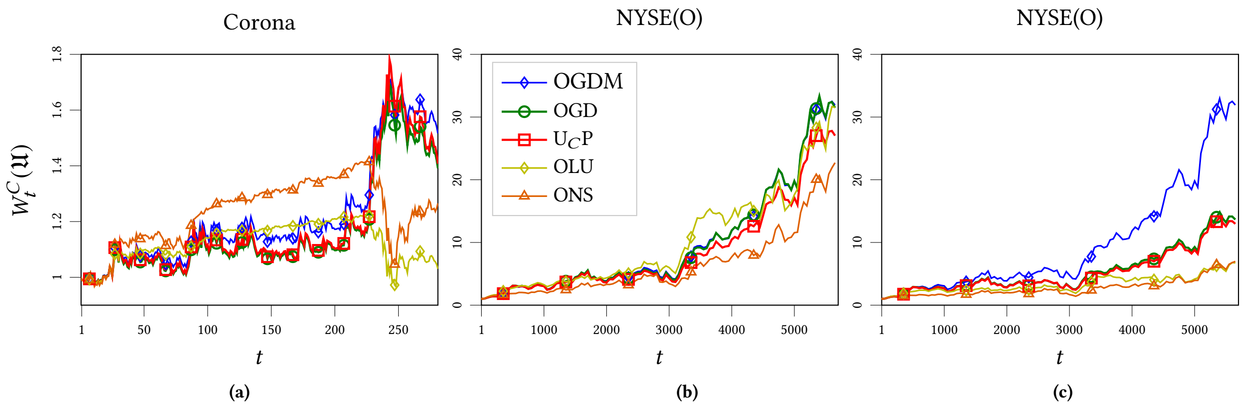

Figure 1: Wealth $W_T^C(\mathfrak{U})$ of a specific run, on the Corona dataset (a) and on 5 stocks of the NYSE(O) for $\gamma = 0$ (b), and $\gamma = 0.01$ (c).

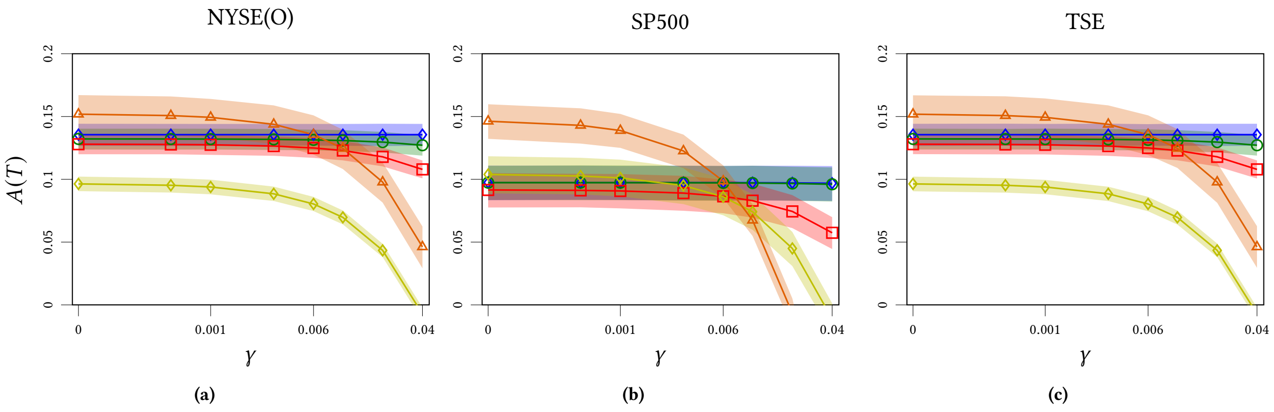

Figure 2: Average APY computed on the wealth $W_T^C(\mathbf{x}_{1:T}, \mathbf{r}_{1:T})$ assuming the costs given by $C_T(\mathfrak{U})$.

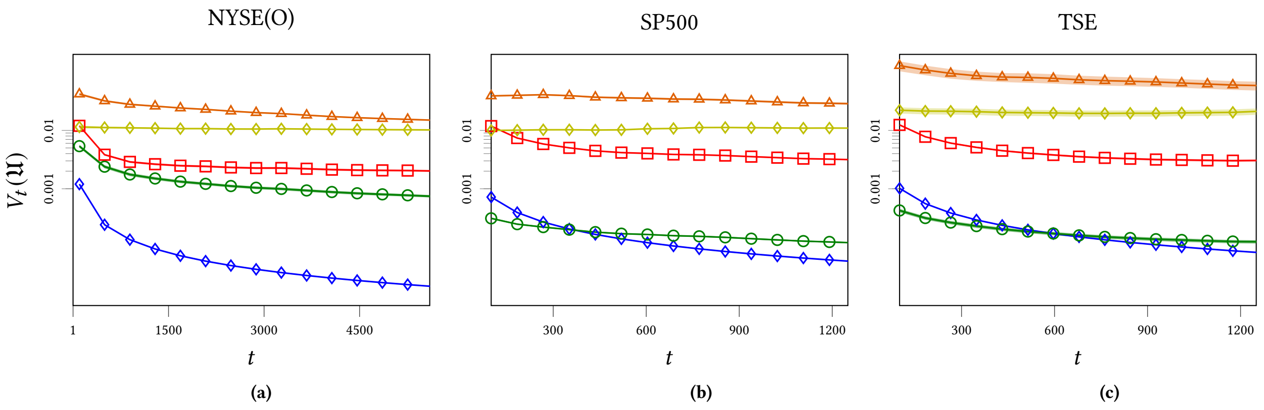

Figure 3: Average variation of the portfolio $V_t(\mathfrak{U})$ incurred on a varying time horizon t.

the worst performance. Finally, comparing OGDM and OGD, the respective behavior varies between the three datasets, and it seems that OGDM is slightly better at minimizing the costs as time increases. This is unexpected given the previous results, in which OGDM seems to provide a larger APY by keeping costs low. Thus, this result suggests that OGDM performs well not just because it is good at handling transaction costs, but also because it has a superior investment strategy.

To conclude, the experiments confirm the theoretical properties discussed in Section 4 and suggest that OGDM is the best algorithm to use in the presence of large ($\gamma > 0.01$) transaction costs.

6 CONCLUSIONS

The focus of this paper is to control transaction costs in the OPO problem. We achieved this result by introducing a novel algorithm: *Online Gradient Descent with Momentum*. Indeed, this paper critically analyses the existing online learning literature dealing with scenarios with transaction costs and proposes the use of OGDM to control (theoretically and empirically) the costs. Existing algorithms designed for this setting are either (i) not providing theoretical guarantees (*e.g.*, the algorithms in [29]), (ii) providing a bound to the total regret, conditionally on unrealistic assumptions (*e.g.*, OLU [14]) or (iii) computationally inefficient (*e.g.*, U$_C$P [7]). In this paper, we proved that OGDM has nice theoretical, empirical, and computational performance in the analyzed setting. Finally, we compared the empirical performance of OGDM with state-of-the-art algorithms on a real dataset and provided insights into the settings in which it is likely to provide larger cumulative wealth.

Future developments could be to extend the bound on the transaction costs to a wider class of algorithms, *e.g.*, the ones derived from Online Mirror Descent (OMD). Furthermore, an extension would be to explore environments that are not completely adversarial and not even completely stochastic. One approach would be to develop an algorithm that plays an equilibrium strategy against adversaries and adapts to stationary environments as proposed in [11]. Finally, it would be interesting to extend the transaction cost model to include liquidity constraints and market impact.

REFERENCES

[1] J. Abernethy and S. Kale. 2013. Adaptive market making via online learning. In *Neural Information Processing Systems*. NeurIPS, Stateline, Nevada, United States, 2058–2066.

[2] A. Agarwal, E. Hazan, S. Kale, and R.E. Schapire. 2006. Algorithms for portfolio management based on the Newton method. In *International Conference on Machine Learning*. ICML, Pittsburgh, Pennsylvania, United States, 9–16.

[3] A. Banerjee, S. Merugu, I.S. Dhillon, and J. Ghosh. 2005. Clustering with Bregman divergences. *J MACH LEARN RES* 6 (2005), 1705–1749.

[4] E.V. Belmega, P. Mertikopoulos, R. Negrel, and L. Sanguinetti. 2018. Online convex optimization and no-regret learning: Algorithms, guarantees and applications. *arXiv:1804.04529* 12 April (2018), 1–34.

[5] H. Bessembinder and H.M. Kaufman. 1997. A comparison of trade execution costs for NYSE and NASDAQ-listed stocks. *J FINANC QUANT ANAL* 32, 3 (1997), 287–310.

[6] Lorenzo Bisi, Luca Sabbioni, Edoardo Vittori, Matteo Papini, and Marcello Restelli. 2020. Risk-Averse Trust Region Optimization for Reward-Volatility Reduction. In *International Joint Conference on Artificial Intelligence*. IJCAI, Held Online, 4583–4589.

[7] A. Blum and A. Kalai. 1999. Universal portfolios with and without transaction costs. *MACH LEARN* 35, 3 (1999), 193–205.

[8] A. Borodin et al. 2004. Can we learn to beat the best stock. In *Neural Information Processing Systems*. NeurIPS, Vancouver, Canada, 345–352.

[9] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *FOUND TRENDS MACH LEARN* 3, 1 (2011), 1–122.

[10] N. Cesa-Bianchi, O. Dekel, and O. Shamir. 2013. Online learning with switching costs and other adaptive adversaries. In *Neural Information Processing Systems*. NeurIPS, Stateline, Nevada, United States, 1160–1168.

[11] Vincent Conitzer and Tuomas Sandholm. 2007. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning* 67, 1-2 (2007), 23–43.

[12] T.M. Cover. 1991. *The Kelly Capital Growth Investment Criterion*. World Scientific, Singapore. 181–209 pages.

[13] P. Das. 2014. *Online convex optimization and its application to online portfolio selection*. Ph.D. Dissertation. The University of Minnesota.

[14] P. Das, N. Johnson, and A. Banerjee. 2013. Online Lazy Updates for Portfolio Selection with Transaction Costs. In *Conference on Artificial Intelligence*. AAAI, Bellevue, Washington, United States, 202–208.

[15] Robert Dochow. 2016. *Online algorithms for the portfolio selection problem*. Springer, Berlin, Germany.

[16] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. 2008. Efficient projections onto the l1-ball for learning in high dimensions. In *International Conference on Machine Learning*. ICML, Helsinki, Finland, 272–279.

[17] J.C. Duchi, S. Shalev-Shwartz, Y. Singer, and A. Tewari. 2010. Composite Objective Mirror Descent. In *Conference on Learning Theory*. COLT, Haifa, Israel, 14–26.

[18] Y. Fang, K.K. Lai, and S. Wang. 2006. Portfolio rebalancing model with transaction costs based on fuzzy decision theory. *EUR J OPER RES* 175, 2 (2006), 879–893.

[19] Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. 2019. Beyond Online Balanced Descent: An Optimal Algorithm for Smoothed Online Optimization. In *Neural Information Processing Systems*. NeurIPS, Vancouver, Canada, 1873–1883.

[20] R.C Grinold and R.N. Kahn. 2000. *Active portfolio management*. McGraw Hill, New York, New York, United States.

[21] N.H. Hakansson and W.T. Ziemba. 1995. Capital growth theory. *Handbooks in operations research and management science* 9 (1995), 65–86.

[22] E. Hazan, A. Agarwal, and S. Kale. 2007. Logarithmic Regret Algorithms for Online Convex Optimization. *MACH LEARN* 69 (2007), 169–192.

[23] D.P. Helmbold, R. E Schapire, Y. Singer, and M.K. Warmuth. 1998. On-Line Portfolio Selection Using Multiplicative Updates. *MATH FINANC* 8, 4 (1998), 325–347.

[24] S. Ito, , D. Hatano, H. Sumita, A. Yabe, T. Fukunaga, N. Kakimura, and K. Kawarabayashi. 2018. Regret Bounds for Online Portfolio Selection with a Cardinality Constraint. In *Neural Information Processing Systems*. NeurIPS, Montréal, Canada, 1–10.

[25] A. Kalai and S. Vempala. 2002. Efficient algorithms for universal portfolios. *J MACH LEARN RES* 3, Nov (2002), 423–440.

[26] J.L. Kelly. 1956. A New Interpretation of Information Rate. *Bell Systems* 21 March (1956), 25–34.

[27] B. Li and S. Hoi. 2014. Online portfolio selection: A survey. *ACM COMPUT SURV* 46, 3 (2014), 35.

[28] B. Li, S. Hoi, D. Sahoo, and Z.Y. Liu. 2015. Moving average reversion strategy for on-line portfolio selection. *ARTIF INTELL* 222 (2015), 104–123.

[29] B. Li, J. Wang, D. Huang, and S. Hoi. 2018. Transaction cost optimization for online portfolio selection. *QUANT FINANC* 18, 8 (2018), 1411–1424.

[30] B. Li, P. Zhao, S. Hoi, and V. Gopalkrishnan. 2012. PAMR: Passive aggressive mean reversion strategy for portfolio selection. *MACH LEARN* 87, 2 (2012), 221–258.

[31] Y. Li, G. Qu, and N. Li. 2018. Online optimization with predictions and switching costs: Fast algorithms and the fundamental limit. *arXiv:1801.07780* 7 March (2018), 1–24.

[32] M. Lin, A. Wierman, A. Roytman, A. Meyerson, and L.L.H. Andrew. 2012. Online optimization with switching cost. *PERF E R SI* 40, 3 (2012), 98–100.

[33] L.C. MacLean, E.O. Thorp, and W.T. Ziemba. 2011. *The Kelly capital growth investment criterion: Theory and practice*. Vol. 3. World Scientific, Singapore.

[34] H. Markowitz. 1952. Portfolio selection. *The journal of finance* 7, 1 (1952), 77–91.

[35] Boris T Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *U. S. S. R. Comput. Math. and Math. Phys.* 4, 5 (1964), 1–17.

[36] Ning Qian. 1999. On the momentum term in gradient descent learning algorithms. *Neural networks* 12, 1 (1999), 145–151.

[37] W.F. Sharpe. 1963. A simplified model for portfolio analysis. *MANAGE SCI* 9, 2 (1963), 277–293.

[38] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*. ICML, Atlanta, Georgia, United States, 1139–1147.

[39] F. Trovò, S. Paladino, M. Restelli, and N. Gatti. 2016. Budgeted multi-armed bandit in continuous action space. In *European Conference on Artificial Intelligence*. ECAI, The Hague, Netherlands, 560–568.

[40] X. Yang, H. Li, Y. Zhang, and J. He. 2018. Reversion strategy for online portfolio selection with transaction costs. *INT J APP DECIS SCI* 11, 1 (2018), 79–99.

[41] M. Zinkevich. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning*. ICML, Washington D.C., United States, 928–936.