Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Deep reinforcement learning for the optimal placement of cryptocurrency limit orders

Matthias Schnaubelt[1]

School of Business, Economics and Society, FAU Erlangen-Nürnberg, Lange Gasse 20, 90403 Nürnberg, Germany

ARTICLE INFO

Article history:
Received 21 June 2020
Accepted 26 April 2021
Available online xxx

Keywords:
Finance
Optimal execution
Limit order books
Machine learning
Deep reinforcement learning

ABSTRACT

This paper presents the first large-scale application of deep reinforcement learning to optimize execution at cryptocurrency exchanges by learning optimal limit order placement strategies. Execution optimization is highly relevant for both professional asset managers and private investors as execution quality affects portfolio performance at economically significant levels and is the target of regulatory supervision. To optimize execution with deep reinforcement learning, we design a problem-specific training environment that introduces a purpose-built reward function, hand-crafted market state features and a virtual limit order exchange. We empirically compare state-of-the-art deep reinforcement learning algorithms to several benchmarks with market data from major cryptocurrency exchanges, which represent an ideal test bed for our study as liquidity costs are relatively high. In total, we leverage 18 months of high-frequency data for several currency pairs with 300 million trades and more than 3.5 million order book states. We find proximal policy optimization to reliably learn superior order placement strategies. By interacting with our simulated limit order exchange, it learns cryptocurrency execution strategies that are empirically known from established markets. Order placement becomes more aggressive in anticipation of lower execution probabilities, which is indicated by trade and order imbalances.

1. Introduction

Optimal or *best execution*, as defined by Bertsimas and Lo (1998), is the execution strategy that minimizes expected execution costs when trading a given number of securities over a fixed time horizon. With optimized execution strategies, investors seek to minimize the implementation shortfall of their portfolio, i.e., the difference between observed market prices at the time of the portfolio allocation decision and the actually executed price (Perold, 1988). This paper presents the first large-scale application of deep reinforcement learning to optimize execution by learning optimal limit order placement strategies.

Execution decisions are frequently made by market participants and their quality severely affects overall transaction costs, which can compromise investment performance at economically significant levels (Anand, Irvine, Puckett, & Venkataraman, 2012; Bertsimas & Lo, 1998; Cont & Kukanov, 2017). For example, Edelen, Evans, and Kadlec (2013) analyze the trading costs of actively managed funds and find that of the 80 bp per-unit trading costs, as much as 14 bp and 53 bp can be attributed to exchange commissions and price impact, respectively. Consequently, execution quality can be a relevant decision factor for institutional clients when allocating trading volume to asset management companies (Ben-Rephael & Israelsen, 2018), and it is not surprising that empirical evidence suggests that strategic order posting is common practice to reduce execution costs (Battalio, Corwin, & Jennings, 2016; Tripathi Abhinava & Dixit, 2020). Poor order placement, on the other hand, can come at high costs, such as opportunity costs when failing to execute an order, a premium due to badly-chosen limit prices, or elevated fees for taking liquidity from the order book.[2] Regulation further forces brokers to optimize execution also for their retail client's orders: For instance, the Markets in Financial Instruments Directive (MiFID II) requires investment firms in the European Union to take all sufficient measures to execute client orders in the best possible way.[3] In the cryptocurrency domain,

E-mail address: matthias.schnaubelt@fau.de

[1] The author has benefited from many helpful discussions with Jonas Dovern, Thomas Fischer, Alexander Glas, Ingo Klein, Christopher Krauss, Markus Schmitz, Oleg Seifert and two anonymous referees.

[2] Consider the following example from our study: Selling 10 BTC as single market order at the Coinbase exchange incurs total execution costs of 32.79 bp on average. If, instead, an execution strategy optimized by deep reinforcement learning is employed, costs are reduced to 16.11 bp. The economic significance of this reduction is evident when comparing it to mean trade profits of 33.8 bp before costs for the trading strategy of Fischer, Krauss, and Deinert (2019).

[3] Compare directive 2014/65/EU, article 27. Similarly, the National Best Bid and Offer (NBBO) regulation of the U.S. Securities and Exchange Commission demands

https://doi.org/10.1016/j.ejor.2021.04.050
0377-2217/© 2021 Elsevier B.V. All rights reserved.

where such regulation does currently not exist, specialized firms offer to optimize trades for their clients (e.g., the "order recommendation system" of Consilium Crypto Inc., 2020).

Finding an optimized execution strategy suited for an investor's investment style and goal is a highly non-trivial task, given that a plethora of determinants, such as exchange-dependent fee schedules, order types or order book depth, influence execution costs (Cont & Kukanov, 2017; Kissell, Glantz, & Malamut, 2004). As Roşu (2012, p. 42) notes, "given the importance of limit order markets, there have been relatively few models that describe price formation and order choice in these markets". Even if a model that accounts for these complexities could be formulated, it would not be easily solvable in closed form (Goettler, Parlour, & Rajan, 2009). Hence, current literature on optimal order execution typically has to rely on simplified models of limit order dynamics that focus on selected aspects of the problem, and their solution may require substantial approximations or numerical approaches.

With this study, we deviate from these classical approaches by deploying recent advances in deep reinforcement learning (DRL) for execution optimization. Model-free reinforcement learning algorithms promise a data-driven and therefore flexible approach to optimal execution by "learning what to do—how to map situations to actions—so as to maximize a numerical reward signal" (Sutton and Barto, 2018, p. 1). Inspired by recent successes in solving complex tasks such as playing Go or controlling a robot's hand to solve the Rubik's Cube (see, e.g., Akkaya et al., 2019; Silver et al., 2018), investors could hope to train DRL algorithms to directly learn an execution strategy that minimizes a given objective such as the implementation shortfall in the given execution environment.[4] To our knowledge, no work has systematically tested the promise of recent advances of DRL for limit order placement in a large-scale empirical study. We seek to close this gap by deploying state-of-the-art DRL algorithms to optimize the placement of limit orders.

First, we develop an environment suited to train and evaluate reinforcement learning agents, which are asked to decide on the price of submitted limit orders over the course of several time steps in order to find an optimized placement strategy. To this end, we introduce a problem-specific reward function based on the realized implementation shortfall, which allows the direct inclusion of exchange-specific constraints such as order commissions. Based on the literature, we construct a feature set that informs the agent about the current market state. To simulate order execution, we develop a virtual limit order exchange that is based on large samples of high-frequency cryptocurrency limit order data from major exchanges. Cryptocurrency exchanges represent a well-suited test environment for our study. Statistical arbitrage strategies in cryptocurrency markets are actively researched (see, e.g., Atsalakis, Atsalaki, Pasiouras, & Zopounidis, 2019; Fischer et al., 2019), and market data is easily accessible. Major cryptocurrency exchanges operate like established limit order exchanges and exhibit very similar stylized facts (Makarov & Schoar, 2020; Schnaubelt, Rende, & Krauss, 2019). One of their key idiosyncrasies, shallow limit order books and hence a relatively high level of liquidity costs, makes them an ideal test bed for optimal execution algorithms.

Second, we deploy two state-of-the-art DRL algorithms to optimize limit order placement. Specifically, we employ deep double Q-learning and proximal policy optimization (PPO), and compare their performance to several benchmark execution policies. We find that PPO realizes the lowest total implementation shortfall across exchanges and currency pairs. Compared to immediate market-order execution, PPO reduces total transaction costs by up to 36.93 percent. Given that a large part of trading costs of mutual funds can be attributed to price impact and in light of typical per-trade returns of statistical arbitrage strategies, these reductions are economically highly significant. The superior performance is found to be robust, as PPO achieves the lowest shortfall among all strategies in 89.06 percent of all studied cases, even when considering further permanent price impact of own limit orders.

Third, we perform several analyses that shed light into the black box of the PPO algorithm and ask whether results are in line with economic rationale. Specifically, we demonstrate that the agent independently learns order submission strategies that are consistent with empirical evidence about strategic order posting in established markets. For example, we find that the agent uses a mix of market and limit orders, where the latter are preferred to avoid additional taker fees – a behavior that is empirically observed in U.S. broker data (Battalio et al., 2016). The placement of orders becomes more aggressive in anticipation of lower execution probabilities, indicated for example by order imbalances. This result aligns well with empirical research on the role of liquidity imbalances as short-term predictors of mid-price returns (Cao, Hansch, & Wang, 2009; Gould & Bonart, 2016) and empirical results on strategic order posting observed from real traders in established markets (see, e.g., Degryse, Jong, Ravenswaaij, & Wuyts, 2005; Griffiths, Smith, Turnbull, & White, 2000; Lehalle & Mounjid, 2017; Tripathi Abhinava & Dixit, 2020).

The remainder of this paper is organized as follows: Section 2 reviews relevant literature and our contribution. Section 3 presents our data set and Section 4 details our methodology. Section 5 presents our results and we conclude in Section 6.

2. Literature review

This study is related to two main strands of literature: First, it is related to literature applying machine learning and especially reinforcement learning in financial markets. In the majority of works, supervised machine learning is used to predict asset returns (see, e.g., Fischer & Krauss, 2018; Schnaubelt, Fischer, & Krauss, 2020; Tsantekidis et al., 2020). The survey of Henrique, Sobreiro, and Kimura (2019) alone covers more than 50 articles focusing on return prediction, often with (deep) neural networks. Fischer (2018) is a recent survey of applications of reinforcement learning in financial markets, most of which focus on portfolio allocation. With the rise of cryptocurrency markets, researchers started to apply machine learning for cryptocurrency return prediction (for more detailed reviews, see Atsalakis et al., 2019; Fischer et al., 2019), with Shah and Zhang (2014) being one of the first studies. In summary, this strand of literature is related to our study in terms of the applied class of algorithms. However, as the majority focuses on trading signal generation, i.e., the prediction of security returns with the goal of portfolio allocation, the addressed problem is distinct.[5]

Second, our study is related to literature on optimal execution. We follow Cont and Kukanov (2017) to further subdivide the decision making during execution into *order scheduling* and *order placement* stages. In the former stage, the volume allocated to a specific asset is split into smaller blocks that are to be executed over a time frame of several minutes to days. In the latter stage, volume slices are translated into a series of orders submitted to a trading venue, and further order details such as type and limit price are specified. Beginning with the seminal works of Bertsimas and Lo (1998) and

that brokers execute customer trades at the best possible prices across exchanges. Compare also Nagy and Gellasch (2018) for a review of the regulatory situation.

[4] Hence, reports that proprietary RL is used by market professionals such as JPMorgan Chase & Co. (Noonan, 2017) or the broker firm Instinet Inc. (Cheng, 2017) to optimize execution are not surprising.

[5] One notable exception is the work of Calvez and Cliff (2018), who use supervised neural networks to replicate the action of existing algorithms in a simulated market environment.

Almgren and Chriss (2001), optimal execution problems have been studied by academics.[6] Both analyze optimal order scheduling as a stochastic control problem, which is solved by dynamic programming to minimize the implementation shortfall assuming of explicit models of price dynamics and market impact. Subsequently, extensions to multiple assets, dynamic liquidity or actual limit order data were considered (see, e.g., Ha & Zhang, 2020; Obizhaeva & Wang, 2013; Tsoukalas, Wang, & Giesecke, 2017). By contrast, other works focus on the stage of optimal order placement, for example, by proposing models to optimize limit order prices, the ratio between market and limit orders or the allocation across exchanges (see, e.g., Bayraktar & Ludkovski, 2014; Cartea & Jaimungal, 2015; Cont & Kukanov, 2017). However, these approaches typically require substantial approximations and may be difficult to extend to real-world applications.

Compared to the number of works modeling optimal execution as a stochastic control problem, empirical research on the use of RL for optimal execution is limited. In their seminal work, Nevmyvaka, Feng, and Kearns (2006) propose a problem-specific RL algorithm for order placement. Over several discrete time steps, the agent has to decide on the price of a limit order with the goal of selling all volume with minimal implementation shortfall. The study is based on a simulated limit order exchange backed by 1.5 years of high-frequency data for three stocks and shows that the proposed algorithm reduces realized shortfalls compared to a submit-and-leave strategy. Despite the 173 citations of the study of Nevmyvaka et al. (2006) on Google Scholar at the time of writing this article, subsequent works with focus on optimal order placement are rare. The exception is the work of Rantil and Dahlen (2018), who apply Sarsa(λ) and PPO to 1.5 years of data from four future contracts. Their results indicate that PPO reduces the implementation shortfall in comparison to their benchmarks in one setting. Most remaining applications of RL for execution optimization focus on the stage of order scheduling: Hendricks and Wilcox (2014) study how to split volume between market orders by extending the model of Almgren and Chriss (2001) to tabular Q-learning. Their empirical analysis is based on one year of market data with five levels of the order book. Similarly, Ning, Ling, and Jaimungal (2018) apply deep double Q-networks in optimizing the volume trajectory of market orders. To evaluate the approach, they employ time series of mid prices from nine stocks. Other works study RL for order scheduling with simulated market environments and synthetic data (Bao & Liu, 2019; Daberius, Granat, & Karlsson, 2019).

In light of the available literature, the contribution of our study is threefold: First, we introduce a self-contained and general framework for the application of DRL to optimize limit order placement with the flexibility to adapt the approach to a trader's real-world requirements. We formulate an improved reward function that includes exchange commissions and differentiates between maker and taker fees. Second, this study is unique in deploying different state-of-the-art DRL algorithms on a one-in-a-kind extensive limit order data set, which allows a robust comparison of methods. Our empirical evaluation is based on high-frequency limit order data from major exchanges and currency pairs covering 18 months. In total, the data comprises 300 million historic trades and more than 3.5 million order book states. Third, our study has a strong focus on a holistic performance evaluation and detailed analysis of the superiority of the PPO strategy. Hence, we are able to derive novel insights regarding the economic rationale behind the decisions of the models, and relate the learned strategy to previous empirical observations on real-world order placement strategies from established exchanges. In a nutshell, this work contributes towards exploring the potential of DRL for execution optimization – an important problem in asset management – with relevant insights for future academic research as well as practical applications.

3. Data

3.1. Data retrieval

Our empirical study is based on a large sample of cryptocurrency limit order data which comprises the 18 months from January 1st, 2018 to June 30th, 2019. In total, we have obtained over 264 GB of raw data. For our empirical evaluation, we select the largest exchanges and currency pairs in terms of traded volume. For the currency pair Bitcoin against US Dollar (BTC/USD), we obtain data from the exchanges BitFinex, Kraken and Coinbase, which have covered an estimated 65 percent of the BTC/USD limit order trading volume during our sample period.[7] To also evaluate algorithms for other currency pairs, we obtain data for the currency pair Ethereum to Bitcoin (ETH/BTC) and Ethereum to US Dollar (ETH/USD). Panel A of Table 1 depicts some main characteristics of these exchanges. Following Schnaubelt et al. (2019), we obtain data directly from the exchanges using their application programming interfaces (API). Specifically, the data consist of transaction data (timestamp with millisecond resolution, price, volume and buy/sell flag) and data from the limit order book. The latter records the limit order volume at all price steps of the order book that were available during the time of retrieval via the exchange's API. To prepare limit order data for our models, we reconstruct the limit order book at a minutely sampling frequency.

3.2. Trade summary statistics

Table A.6 in the online appendix reports summary statistics of our trade data. In terms of traded volume, BitFinex is the largest exchange serving the BTC/USD market in our sample (average daily volume of 56111 BTC). The Kraken exchange ranks second (45410 BTC) before the Coinbase exchange (23706 BTC). With 0.3483 BTC and 0.3782 BTC, average trade sizes at the BitFinex and Kraken exchanges are comparable. The Coinbase exchange presents an exception with an average trade size of 0.1723 BTC. In an average minute, about 100 single trades are recorded at the exchanges. Figure A.2 in the online appendix depicts day-end prices and weekly traded volume for the BTC/USD market at the BitFinex exchange. Turning to the other currency pairs, we note a lower number of trades. On average, 42.20 and 55.90 trades occur per minute for the ETH/USD and ETH/BTC pairs, respectively. Also, the average value of the trades is lower. For the BitFinex ETH/USD market, we find a mean trade value of 1528 USD, which is lower than the average trade value for the BTC/USD pair at the same exchange (2393 USD). Similarly, the average trade value for the ETH/BTC pair is only 0.1058 BTC, which corresponds to less than one third of the mean trade volume at the BitFinex BTC/USD market (0.3483 BTC).

3.3. Estimating price impact

To provide an intuition for the price impact of orders, we perform two analyses. First, we analyze liquidity measures using the reconstructed limit order book data (Panel B of Table 1). Mean relative bid-ask spreads for the BTC/USD currency pair at the BitFinex and Coinbase exchanges are at 0.679 bp and 0.385 bp, which is roughly one order of magnitude smaller than at the Kraken exchange. Compared to the bid-ask spread, mean liquidity costs,

[6] See also Cartea, Jaimungal, and Penalva (2015) for a survey of further works.

[7] To estimate exchange trading volume shares, we obtain data from data.bitcoinity.org for our sample period.

Table 1
Estimation of price impact. *Panel A* reports tick sizes, i.e., the smallest possible price increments, and relative exchange commissions. Commissions are split into a maker and taker fees for providing and removing liquidity from the order book, respectively. *Panel B* depicts descriptive statistics for the order book data. Data is shown for the sample period from January 1st, 2018 to June 30th, 2019. Value for the bid/ask side of the order book are averaged. *Panel C* reports regression results of minutely returns on the contemporaneous signed volume: $r_t = \lambda s_t + \varepsilon_t$. The t-statistics given in parentheses are computed using heteroskedasticity robust standard errors.

	BitFinex			Coinbase	Kraken
	BTC/USD	ETH/USD	ETH/BTC	BTC/USD	BTC/USD
Panel A: Overview of exchange properties					
Tick size	10^{-1}	10^{-2}	10^{-5}	10^{-2}	10^{-1}
Maker fee [bp]	10	10	10	0	16
Taker fee [bp]	20	20	20	30	26
Panel B: Summary statistics of limit order book data					
Count	726,722	719,326	723,055	759,005	629,457
Relative bid-ask spread [bp]	0.679	1.283	2.822	0.385	3.465
Best-bid/ask volume	7.584	66.357	25.537	1.395	2.466
Rel. bid/ask VWAP for 10 BTC [bp]	3.303	–	–	2.978	8.098
Rel. bid/ask VWAP for 100 ETH [bp]	–	5.674	9.913	–	–
Panel C: Estimation of price impact					
$\lambda \times 10^6$	4.771	1.332	0.083	22.222	2.641
	(16.295)	(27.010)	(3.319)	(41.991)	(45.054)
N	726,722	719,326	723,055	759,005	629,457
R^2	0.079	0.097	0.008	0.179	0.032

i.e., the relative change of the volume-weighted-average price of market-order execution, for a volume of 10 BTC are considerably higher than the mean bid-ask spread. For example, selling a volume of 10 BTC with a market order at the BitFinex exchange increases the obtained volume-weighted price by 3.303 bp on average relative to the mid price. The largest average volume at the best-bid/ask price for the BTC/USD pair is found at the BitFinex exchange (7.584 BTC), and the lowest at the Coinbase exchange (1.395 BTC). Looking at the results for the BitFinex ETH/USD and ETH/BTC pairs, we find bid-ask spreads (1.283 bp and 2.822 bp, respectively) to be considerably wider than those of the BTC/USD currency pair.

Second, inspired by Cont, Kukanov, and Stoikov (2014) as well as Makarov and Schoar (2020), we estimate the price impact over one-minute windows with the linear impact model

$$r_t = \lambda s_t + \varepsilon_t, \tag{1}$$

where r_t denotes the mid-price return over a one-minute window and s_t is the contemporaneous signed trade volume.[8] Results are given in Panel C of Table 1. Results for the BTC/USD pair possess a similar order of magnitude when compared to the impact coefficients obtained by Makarov and Schoar (2020) for 5-minute intervals. For example, selling a volume of 10 BTC at the BitFinex BTC/USD exchange decreases the price by 0.477 bp on average, which is considerably lower than the volume-weighted immediate impact of a market order of the same size (3.303 bp, Panel B).

4. Methodology

This section outlines our methodology. First, we describe our virtual exchange that simulates limit order matching based on actual market data (Section 4.1). Second, we formulate optimal order placement in the reinforcement learning framework (Section 4.2). Third, we train reinforcement learning algorithms (Section 4.3) and benchmark the learned strategies against static execution policies (Section 4.4). Section 4.5 outlines the performance evaluation of the learned strategies.

4.1. Simulating order matching in a virtual limit order exchange

To train and evaluate algorithms for the task of optimal order placement, we set up a virtual limit order exchange that is based on high-frequency market data. In the following, we describe the simulation of the execution of a limit order submitted at time step t, $x_t = (p_t, \omega_t)$, where $p_t \in \mathbb{R}_>$ and $\omega_t \in \mathbb{R}$ denote its limit price and volume. For $\omega_t > 0$, the order is a sell order, and else a buy order. We assume that the order is valid until the next time step $t+1$. The simulation of order execution proceeds in two consecutive steps similar to the price-time prioritized matching found at most real limit order exchanges.[9] In the first step, the simulation checks whether the order can be matched immediately with orders in the order book. In case of an ask order, the algorithm compares the order's limit price p_x with the prices of limit orders on the bid side of the limit order book, starting with the best-bid order. If prices of order pairs are compatible, i.e., ask prices are below bid prices, a virtual trade is executed. Afterwards, ω_t^{im} (with the same sign as ω_t) and p_t^{im} store the resulting immediately matched volume and volume-weighted execution price, respectively. In the second step, the remaining order volume $\omega_t - \omega_t^{\mathrm{im}}$ enters the virtual limit order book and execution is simulated based on the stream of trades between time steps t and $t+1$. Same-side orders in the order book at time t with equal or better limit prices are given priority, since these would have been matched before our newly submitted order. The matching of limit order volume then proceeds as follows: Any trade from the time-ordered stream of trades following time step t is checked in terms of execution price. If the trade was executed at a price higher (lower) than the ask (bid) limit price p_t, the traded volume reduces the remaining volume of our own limit order. After all trades between time steps t and $t+1$ have been checked, the limit order matching results in the execution of a volume $|\omega_t^{\mathrm{lim}}| \in [0, \omega_t - \omega_t^{\mathrm{im}}]$ at the volume-weighted execution price p_t^{lim}.

This simulation of order matching makes several approximations. First, we assume that the submission of our own orders is instantaneous, i.e., that there is no time delay between the observation of the order book and the subsequent own order submission. In practice, technological means such as short-distance links

[8] Note that we are estimating the exchange-specific price impact of signed volume, as we are interested in approximating the permanent price impact of a new own order in the context of our exchange simulation. Specifically, we do not decompose signed volumes and returns into idiosyncratic and common components as in Makarov and Schoar (2020). This way, we are able to also take into account ETH/USD and ETH/BTC pairs with the same approach.

[9] Compare, for example, Gould et al. (2013). Similar data-based limit order exchange simulations are used, e.g., by Nevmyvaka et al. (2006), and our approach is inspired by the simulation used by Rantil and Dahlen (2018).

to the exchange and optimized hardware can minimize this delay. Second, we assume that there is no hidden liquidity in the order book. As only one exchange has supported hidden liquidity in our sample period (the *BitFinex* exchange, compare Schnaubelt et al., 2019), we are able to study the effect of this assumption in our experiments. Third, we assume that the submission of our own limit orders does not have a permanent impact on market price or liquidity and does not impact the behavior of other traders.[10] A rule of thumb is that permanent price impact is insignificant as long as the daily participation rate is below 10 percent (Almgren and Chriss, 2001, p. 24). Limit order markets are typically found to possess a relatively high resilience of liquidity, i.e., liquidity levels revert to their pre-trade average within a short period of time (Degryse et al., 2005; Gomber, Schweickert, & Theissen, 2015). For cryptocurrency exchanges, Schnaubelt et al. (2019) similarly find that liquidity close to the bid-ask spread recovers within a few seconds, even for trades in the top percentile of trading volume. As our empirical study focuses on optimal order placement, i.e., assumes that large volumes have been split into smaller blocks during order scheduling, we select initial volumes υ_0 that are in the order of magnitude of typical average minutely trading volume, hence well within the limits of these approximations. Also, our sampling frequency of one minute is considerably larger than typical recovery time scales.

4.2. *A reinforcement learning formulation of optimal order placement*

Next, we formulate the optimal order placement problem as given in Nevmyvaka et al. (2006) in the framework of reinforcement learning.[11] The agent, which we can think of as an investor, faces the following problem: Over the course of an episode of T discrete time steps, the agent has to liquidate an initial position of $\upsilon_0 \in \mathbb{R}$ units of an asset at a limit order exchange. The initial volume υ_0 is positive when selling, and negative when buying. The goal of the agent is to minimize total execution costs for υ_0, consisting of an implementation shortfall (Perold, 1988) and an exchange commission. In every time step $t \in \{0, \dots, T-1\}$, the agent observes the current environment through a state variable s_t and chooses an action a_t from a discrete set of actions. This action encodes the price of the submitted limit order. The agent follows a policy π that specifies which action to choose given the observed state. Our virtual exchange then simulates the (potentially partial) order execution. Based on the volume and price of the executed trade, the agent receives an immediate reward r_{t+1}, and the remaining volume is the initial volume of the next time step, υ_{t+1}. In the following, we complete this description of the reinforcement learning environment. First, Section 4.2.1 describes the action space. Second, Section 4.2.2 provides details on the reward function. Third, we discuss the state space in Section 4.2.3.

4.2.1. *Action space*

The agent chooses an action a_t from a set of actions $\mathcal{A} = \{0, 1, \dots, 2N_{\text{action}}\}$ at every time step t, where N_{action} specifies the size of the action space. The first action ($a = 0$) encodes the order of zero volume. In line with Nevmyvaka et al. (2006), the remaining actions encode the number of ticks that the order is placed away from the current best-bid or best-ask price. The size of the order is always the currently remaining volume, υ_t. Specifically, if $\upsilon_t > 0$, the submitted order $x_t = (p_t, \omega_t)$ for action a_t is

$$x_t^{\text{sell}} = \begin{cases} (\text{ask}_t + \Delta p(a_t - N_{\text{action}}),\ \upsilon_t), & \text{if } a_t > 0, \\ (0,\ 0), & \text{if } a_t = 0. \end{cases} \quad (2)$$

Herein, Δp denotes the tick size, i.e., the smallest possible price increment, and ask_t is the best-ask price. If $\upsilon_t < 0$, the order is specified similarly with a plus sign instead of the first minus sign, and relative to the best-bid price bid_t. The order persists until complete execution or until it is updated in time step $t+1$. In the last time step, any remaining volume to sell (buy) is executed with a market order, i.e., with a limit order price $p_{T-1} = -\infty$ ($p_{T-1} = \infty$) to ensure full execution.

4.2.2. *Reward function*

After the execution of action a_t, the agent is rewarded with a signal $r_{t+1} \in \mathbb{R}$ given by

$$r_{t+1} = \frac{\omega_t^{\text{ex}}}{\upsilon_0}\left(\frac{\omega_t^{\text{ex}} p_t^{\text{ex}} - c_t^{\text{ex}}}{\omega_t^{\text{ex}}\,\text{mid}_0} - 1\right). \quad (3)$$

Here, $\omega_t^{\text{ex}} = \omega_t^{\text{im}} + \omega_t^{\text{lim}}$ denotes the executed volume, and $p_t^{\text{ex}} = (p_t^{\text{im}}\omega_t^{\text{im}} + p_t^{\text{lim}}\omega_t^{\text{lim}})/\omega_t^{\text{ex}}$ is the corresponding volume-weighted average execution price for time step t. The total exchange commission is given in terms of a fraction of trade value, i.e., $c_t^{\text{ex}} = C^{\text{im}}|\omega_t^{\text{im}}|p_t^{\text{im}} + C^{\text{lim}}|\omega_t^{\text{lim}}|p_t^{\text{lim}} \geq 0$, where $C^{\text{im}}, C^{\text{lim}} \in \mathbb{R}_{\geq}$ specify exchange fees for taking and providing liquidity, respectively. We measure the realized implementation shortfall relative to the mid price at the first time step, i.e., $\text{mid}_0 = (\text{ask}_0 + \text{bid}_0)/2$. Instead of considering absolute proceeds from the trade as in Nevmyvaka et al. (2006), we use a scaled reward function to account for potential changes in market prices over the course of the training and testing periods. This form of the reward function has the favorable property that the sum of all rewards of an episode is the relative total implementation shortfall.

4.2.3. *State space and feature generation*

To support its choice of action, the agent observes the market state represented as a vector

$$s_t = (\vartheta_t, \rho_t, \tilde{x}_{t,1}, \dots, \tilde{x}_{t,N_{\text{feature}}}) \in \mathcal{S} = (0, 1] \times [-1, 1] \times \mathbb{R}^{N_{\text{feature}}}. \quad (4)$$

The first entry encodes the time remaining to liquidate the position, scaled to the unit interval, i.e., $\vartheta_t = 1 - t/T$. For the second entry, we calculate the remaining volume as a fraction of the initial volume as $\rho_t = \upsilon_t/|\upsilon_0|$. The remaining features $x_{t,1}, \dots, x_{t,N_{\text{feature}}}$ capture the current state of the order book and previous trading activity. Based on the literature[12], we determine several measures of market liquidity and potential predictors of temporary and permanent price impact. All features employed in this study are listed in Table 2. Following common procedure for data preprocessing for neural networks, we scale every feature with a rolling-window standardization, $\tilde{x}_{t,i} = \frac{x_{t,i} - \bar{x}_{t,i}}{\sigma_{t,i}}, \quad \forall \ i \in \{1, \dots, N_{\text{features}}\}$, where $\bar{x}_{t,i}$ and $\sigma_{t,i}$ denote the rolling mean and standard deviation over the past 1440 values (i.e., one trading day) of the i-th feature.

4.3. *Reinforcement learning algorithms*

The goal of the agent is to maximize cumulated future rewards, which are expressed as the sum of all discounted future returns following time step t, i.e.,

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k, \quad (5)$$

[10] This assumption is not uncommon in the study optimal execution, see, for example, Nevmyvaka et al. (2006) or Hendricks and Wilcox (2014). Please note that the temporary and permanent price impact of other trader's orders is part of the simulation, as these are part of our data set. Furthermore, we study the effect of this assumption in an extended robustness check in Section 5.1.5 by including a linear permanent price impact, and find similar results.

[11] In the following, we use the common notation for reinforcement learning from Sutton and Barto (2018).

[12] A survey on the topic can be found in Gould et al. (2013).

Table 2
Description of features. This table lists all features used within the reinforcement learning models.

Variable	Description	Reference(s)
Transaction imbalances		
TC-IMBAL	*Trade count imbalance:* Difference of the number of sell to buy trades in the past minute, normalized by the total count	Gopikrishnan, Plerou, Gabaix, and Stanley (2000); Plerou, Gopikrishnan, Gabaix, and Stanley (2002)
TV-IMBAL	*Trade volume imbalance:* Difference of the sell to buy trade volume in the past minute, normalized by the total traded volume	Cao et al. (2009); Cont et al. (2014)
Best-order volumes and imbalances		
BO-IMBAL	*Best-order imbalance:* Difference between the current best-bid and best-ask volume, scaled by the sum of both volumes	Gould and Bonart (2016)
VOL-BID/ASK	*Best-order volumes:* Current best-bid/best-ask volume	Ranaldo (2004)
Q-IMBAL(N_p)	*Queue imbalance:* Difference between the bid/ask volumes at the $N_p \in \{5, 10, 15, 20\}$ best price steps, scaled by their sum	Cao et al. (2009)
CVOL-BID(N_p) *CVOL-ASK*(N_p)	*Cumulated order volumes:* Current cumulated volume in the order book at the best $N_p \in \{10, 15, 20\}$ bid or ask price steps	Cao et al. (2009)
Volatility and current price drift		
VOLA	*Volatility:* Square root of the mean of squared logarithmic mid-price returns over the past 30 minutes	Danielsson and Payne (2001); Ranaldo (2004)
DRIFT	*Mid-price return drift:* Simple return of the mid price relative to the mid price of the first time step	
Current cost of liquidity		
LC-BID(V) *LC-ASK*(V)	*Relative buy/sell liquidity cost:* Execution cost of a buy or sell market order, relative to current mid price. We use volumes of $V \in \{10, 20, 30, 50\}$*BTC* and $V \in \{10, 20, 30, 50, 100\}$*ETH* for BTC and ETH denominated pairs, respectively.	Gomber et al. (2015)
BA-SPREAD	*Bid-ask spread:* Current bid-ask spread, calculated as the difference between ask and bid price scaled by the mid price	Cao et al. (2009); Ranaldo (2004)

where $\gamma \in [0, 1]$ denotes the discount factor.[13] The behavior of the agent is described by a policy $\pi(a|s)$, which is the probability of choosing action $A_t = a \in \mathcal{A}$ conditional on the observed state $S_t = s \in \mathcal{S}$. The action-value function under policy π describes the expected return of an action a for an agent in some state s, and is defined as $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$, where $\mathbb{E}_\pi[\bullet]$ denotes an expected value given that the agent behaves according to policy π. An optimal policy π_* has the optimal action-value function $q_*(s, a) = \max_\pi q_\pi(s, a)$, and fulfills a Bellman optimality equation (Sutton & Barto, 2018) which states that the optimal action-value function is given by the expected reward of action a plus the discounted expected value of the best action in the subsequent state S_{t+1}, i.e.,

$$q_*(s, a) = \mathbb{E}_{S_{t+1}, R_{t+1}}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \,|\, S_t = s, A_t = a\right]. \quad (6)$$

In the following, we describe the reinforcement learning algorithms studied in this paper, specifically, the problem-specific backward-induction Q-learning algorithm of Nevmyvaka et al. (2006) (Section 4.3.1), deep double Q-networks (Section 4.3.2) and proximal policy optimization (Section 4.3.3). We choose the latter two algorithms as they represent recent advances in deep reinforcement learning and are representatives of two common reinforcement learning approaches, i.e., value-based and policy-based algorithms. They are suitable for our discrete action space, have a stable implementation in Hill et al. (2018), and have been successfully applied to various different general tasks (van Hasselt, Guez, & Silver, 2015; Mnih et al., 2015; Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017b) and specifically to optimal execution problems (Ning et al., 2018; Rantil & Dahlen, 2018; Daberius et al., 2019).

4.3.1. Backward-induction Q-learning

As a first benchmark, we employ the problem-specific reinforcement learning algorithm of Nevmyvaka et al. (2006). It is similar to the tabular Q-learning method, however exploits the specific structure of the optimal placement problem and the fact that all market data used in training are known a priori. Tabular Q-learning (Watkins, 1989) approximates the optimal action-value function q_* by iteratively updating the so-called Q table according to the update rule

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right], \quad (7)$$

where $\alpha \in (0, 1]$ denotes a constant step size. By contrast, the approach of Nevmyvaka et al. (2006) assumes that the optimal placement problem can be approximated by a Markov decision process. Hence, the optimal action does not depend on any previous action, and the optimal action-value function can be learned backwards in time, starting at the last time step, $t = T - 1$. As market data are finite, we can introduce the following sampling scheme to collect all possible state-action-reward experience tuples that can occur in time step t: First, quantile-based discretization functions for the remaining volume, $c_\rho : [-1, 1] \mapsto \{0, \ldots, k_\rho\}$, and for the market variables, $c_x : \mathbb{R}^{N_{\text{feature}}} \mapsto \{0, \ldots, k_x\}^{N_{\text{feature}}}$, are introduced, which return the indices of one of $k_\rho + 1$ (for c_x, $k_x + 1$) classes. In a second step, we iterate over all possible discretized state variables $\tilde{s}_t = (T - t, c_\rho(\rho_t), c_x(\tilde{x}_t))$, $\forall \rho_t \in \mathcal{R}, \tilde{x}_t \in \mathcal{T}$, where $\mathcal{T}$ denotes the available standardized training data, and $\mathcal{R}$ is the set of discretized remaining volumes given by $\mathcal{R} = \{0, 1/k_\rho, 2/k_\rho, \ldots, 1\}$ in the case of selling the asset. Third, for every state, we simulate the execution of all actions $a \in \mathcal{A}$, and add the resulting new states

$$\tilde{s}_{t+1,a} = (T - t - 1, c_\rho(\rho_{t+1}), c_x(\tilde{x}_{t+1})), \ \forall a \in \mathcal{A}, \quad (8)$$

and rewards $r_{t+1,a}$ as experience tuples $e_a = (\tilde{s}_t, a_t, r_{t+1,a}, \tilde{s}_{t+1,a})$, $\forall a \in \mathcal{A}$, to the experience set $\mathcal{D}_t$ for this time step. Finally, the action-value table can be updated for all states of time step t by averaging over expected returns, i.e.,

$$Q(\tilde{s}, a) = \frac{1}{|\mathcal{D}_t(\tilde{s})|} \sum_{(\tilde{s}_t, a_t, r_{t+1}, \tilde{s}_{t+1}) \in \mathcal{D}_t(\tilde{s})} r_{t+1} + \max_{a'} Q(\tilde{s}_{t+1}, a'), \quad (9)$$

where $\mathcal{D}_t(\tilde{s}) = \{(\tilde{s}_t, a_t, r_{t+1}, \tilde{s}_{t+1}) \in \mathcal{D}_t : \tilde{s}_t = \tilde{s}\}$ denotes the subset of experience for the updated state $\tilde{s}$. The elements of the Q-table for the next state, $Q(\tilde{s}_{t+1}, a')$, are already fully populated as time step $t + 1$ has been visited in the last iteration. The algorithm

[13] For small values of γ, the agent's goal is to maximize immediate rewards; if instead γ is close to one, the agent behaves long-sighted (Sutton & Barto, 2018). As we assess the agent's performance in terms of the total implementation shortfall after T steps, we set $\gamma = 1$ to obtain an equal weight for all time steps.

proceeds backwards until the first time step is reached. In line with Nevmyvaka et al. (2006), we use a volume discretization with $k_\rho = 8$ and feature discretization with $k_x = 10$, where bin edges are based on the feature's deciles.

4.3.2. Deep double Q-networks

As second reinforcement learning algorithm, we employ deep double Q-networks (DDQN), which have been successfully applied to complex tasks such as playing Atari video games (van Hasselt et al., 2015). The following summary is based on the initial papers (van Hasselt et al., 2015; Mnih et al., 2013; 2015) as well as the implementation in Hill et al. (2018). For high-dimensional state spaces, tabular Q-learning does not generalize well. Therefore, the action-value function is often estimated by a function approximator. Specifically, deep Q-learning (Mnih et al., 2013; 2015) uses a neural network (Q-network) with network weights θ to estimate the optimal action-value function, $Q_\theta(s,a) \approx q_*(s,a)$. To train the Q-network, one iteratively minimizes a sequence of loss functions,

$$L_i(\theta_i) = \mathbb{E}_{S,A}\Big[\big(y_i(S,A) - Q_{\theta_i}(S,A)\big)^2\Big], \tag{10}$$

where θ_i denotes the optimized network weights, and the target $y_i(s,a)$ is the expected action value obtained from a target Q-network with parameters θ_i^-, i.e.,

$$y_i^{\mathrm{DQN}}(s,a) = \mathbb{E}_{S'}\left[R_{t+1} + \gamma \max_{a'} Q_{\theta_i^-}(S',a') \,|\, S_t = s, A_t = a\right]. \tag{11}$$

The target network parameters θ_i^- are set to the current parameter values θ_i after every C steps. The expectation in Equation (10) is taken with respect to the behavior distribution, i.e., the probability distribution over states S and actions A. To decorrelate the sequence of training observations, increase data efficiency and avoid unstable parameters during training, deep Q-learning introduces a replay memory comprised of the agent's last N experience tuples $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$. In every time step, the agent first executes an action according to an ϵ-greedy policy: With probability $1-\epsilon$, the algorithm selects the optimal (greedy) action according to the current policy and with probability ϵ, it selects a random action to explore the environment. Then, previous experience is replayed: A minibatch consisting of random observations from the replay memory and the experience from the currently executed action is used to minimize the loss function with stochastic gradient descent. In Equation (11), the use of the maximum operator implies that the same network weights θ_i^- are used to select and evaluate action a', which bears the risk of overoptimistic value estimates. To prevent this, deep double Q-networks (van Hasselt et al., 2015) decouple the selection and evaluation of the action by replacing the target with

$$y_i^{\mathrm{DDQN}}(s,a) {=} \mathbb{E}_{S'}\big[R_{t+1} + \gamma Q_{\theta_i^-}(S', \mathrm{argmax}_{a'} Q_{\theta_i}(S',a')) \,|\, S{=}s, A{=}a\big]. \tag{12}$$

In our study, we use the implementation of deep double Q-learning from Hill et al. (2018), and keep most parameters at their default values. Specifically, we use a feed-forward network architecture with two hidden layers, each consisting of 64 neurons. The size of the replay memory is 5×10^4 tuples, and C is set to 500 steps. We increase the minibatch size to 256 to reduce noise by averaging over larger experience samples. The algorithm is trained over a total of 10^7 time steps. The probability of selecting a random action, ϵ, is 1 at the beginning of the training, and reduced linearly in time over the first quarter of training time steps to a value of 0.02 for the remaining training.

4.3.3. Proximal policy optimization

As third reinforcement learning algorithm, we employ the proximal policy optimization (PPO) algorithm. The following summary is based on Schulman, Levine, Moritz, Jordan, and Abbeel (2017a) and Schulman et al. (2017b), as well as the implementation in Hill et al. (2018). PPO is an actor-critic method that targets data efficiency and robustness, rendering it an attractive choice in light of noisy financial data. As a policy gradient method, PPO seeks to directly learn a parameterized stochastic policy $\pi_\theta(a|s) : \mathcal{S} \times \mathcal{A} \mapsto [0,1]$ (also called policy surrogate), by maximizing an objective function $L(\theta)$ with respect to θ. In our implementation, the policy function is approximated by a neural network. PPO introduces the objective function

$$L_t^{\mathrm{PPO}}(\theta) = \hat{\mathbb{E}}_N\Big[L_t^{\mathrm{CLIP}}(\theta) - c_{\mathrm{VF}} L_t^{\mathrm{VF}}(\theta) + c_{\mathrm{H}} H[\pi_\theta](s_t)\Big], \tag{13}$$

where $\hat{\mathbb{E}}_N$ denotes an average over a finite sample of experience. Typically, the current policy π is run for N time steps, and then the sampled experience trajectory is used to update the policy π with minibatch stochastic gradient descent or Adam (Kingma & Ba, 2017). With the first term, PPO introduces a clipped objective function,

$$L_t^{\mathrm{CLIP}}(\theta) = \min\Big(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big), \tag{14}$$

which suppresses large policy updates by clipping the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ such that it falls within the interval $[1-\epsilon, 1+\epsilon]$. The advantage function $A^\pi(s,a)$ measures how much better the action is compared to the policy's default behavior, i.e., $A^\pi(s,a) = Q_\pi(s,a) - v_\pi(s)$. The advantage estimator depends on the state-value function $v_\pi(s_t)$, which is estimated with a second network, the critic network. As parameters are shared between the state-value function estimator and the policy surrogate $\pi(a|s;\theta)$, a value-function error term $L_t^{\mathrm{VF}} = \big(v_\theta(s_t) - v_t^{\mathrm{targ}}\big)^2$ is added to the objective function (scaled by a positive constant parameter c_{VF}), where v_t^{targ} denotes the state-value function's training target. Following Williams (1992) and Mnih et al. (2016), PPO also adds the entropy of the policy π_θ for state s_t, $H[\pi_\theta](s_t)$, scaled by a constant positive parameter c_{H}, to the objective in order to discourage early convergence to purely deterministic policies, and hence to improve exploration. Our parameter settings are based on the default values of Hill et al. (2018). Specifically, the policy network is a feed-forward network with two hidden layers, each consisting of 64 neurons. We sample from a single environment and use the default trajectory length of $N = 128$. The weighting factor of the generalized advantage estimator is set to $\lambda = 0.95$. The clipping parameter ϵ is set to 0.2, the value-function error and entropy terms are scaled by $c_{\mathrm{VF}} = 0.5$ and $c_{\mathrm{H}} = 0.01$, respectively. The objective is maximized with an Adam minibatch size of 4 and a learning rate of 2.5×10^{-4}. The algorithm is trained over a total of 10^7 time steps.

4.4. Benchmark execution strategies

As a first benchmark, we employ a submit-and-leave strategy (Nevmyvaka et al., 2006; Nevmyvaka, Kearns, Papandreou, & Sycara, 2005). In every time step, the strategy submits a limit order with the remaining volume v_t at the current best limit price. Specifically, for a sell volume, $v_0 > 0$ (buy volume, $v_0 < 0$), the limit order is placed at the best-ask price, i.e., at $p_0^{\mathrm{sell}} = \mathrm{ask}_0$ ($p_0^{\mathrm{buy}} = \mathrm{bid}_0$). As a second benchmark, we consider a strategy that immediately executes all volume v_0 at the initial time step using an appropriate market order. As third benchmark, we employ time-weighted execution. This strategy executes the volume fraction v_0/T in every time step as a market order. Under the assumption of a linear price impact and in absence of short-term price predictability, this strategy is optimal (Bertsimas & Lo, 1998).

4.5. Study design and performance evaluation

We validate reinforcement learning models in a rolling-window scheme. Specifically, we train our models on six months of training data, and apply them on the subsequent out-of-sample validation period of three months. We then roll forward the validation period by three months.[14] Each episode in the execution environment corresponds to a block of consecutive observations from our data set. For the training of deep reinforcement learning algorithms, we randomly select a root observation from the data set to initialize the state of the environment. For out-of-sample testing, we iterate over all validation observations and select them once as root observation to initialize the environment.[15] To evaluate the performance of execution strategies, we consider the average relative implementation shortfall, i.e., the realized shortfall of an episode relative to the mid price at the start of the episode.

5. Results

We present our results in three steps: First, we compare the out-of-sample performance of reinforcement learning algorithms and benchmark strategies (Section 5.1). Next, we perform in-depth analyses to shed light into the black box of deep reinforcement learning. Specifically, we analyze the contribution of features (Section 5.2). Finally, we examine the chosen actions conditional on individual feature values to understand the underlying decision making (Section 5.3).

5.1. Overview of empirical findings

First, we compare the out-of-sample performance of reinforcement learning algorithms with our benchmark execution strategies. To this end, we independently train the agents on our four training sets for different combinations of exchange, currency pair and initial trading volume v_0. Following Nevmyvaka et al. (2006), we select an episode length of $T = 4$. For the BTC/USD market, we calculate results for initial trading volumes v_0 of 10 BTC, 20 BTC and 50 BTC. On the one hand, these volumes correspond to approximately 25 to 300 percent of average minutely trading volume (compare Table A.6), hence seem still in line with the approximations outlined in Section 4.1. On the other hand, these volumes exceed typical best-bid/best-ask order book depths (compare Table A.7), hence their execution would typically incur high liquidity costs. Similarly, we select initial volumes of 100 ETH and 200 ETH for the ETH/USD and ETH/BTC markets. We then evaluate the trained agents in validation environments that are based on the respective test data sets.

5.1.1. Comparing algorithms by total out-of-sample shortfall

Panel A of Table 3 depicts mean realized implementation shortfalls including exchange commissions relative to the mid price at the beginning of the episode.[16] For example, if the agent is asked to sell starting from an initial mid price $mid_0 = 5000USD$, a realized implementation shortfall of -15bp corresponds to an average execution price including exchange commissions of 4992.5 USD. As expected, executing all volume in the first time step with a market order (strategy IM) yields the worst realized shortfall in all settings, as the market order 'eats into the order book' to fill the requested volume. In comparison, splitting the market order equally over all T time steps (strategy TW) significantly reduces the realized implementation shortfall. For the example of selling $v_0 = 20BTC$ at the BitFinex BTC/USD market, the IM strategy results in a shortfall of -24.98bp, which is reduced by 13.29 percent to -21.66bp with TW execution. A further improvement over the TW strategy is achieved by the submit-and-leave strategy (S&L), which reduces the total implementation shortfall on average by 11.28 percent over the TW execution strategy. Switching to the more flexible backwards-induction Q-learning approach (BQL) results in a marginal improvement of 3.72 percent over the static S&L strategy.

Turning to the performance of deep reinforcement learning agents, we find that deep double Q-networks (DDQN) achieve an average performance improvement of 3.74 percent over the S&L strategy. In comparison, proximal policy optimization (PPO) reduces the overall implementation shortfall by an average of 8.65 percent over the S&L strategy. For executing a volume of 20 BTC at the Coinbase exchange for the currency pair BTC/USD, the realized shortfall with the PPO agent is at -20.99bp, a significant improvement when compared to the value of -34.65bp for IM execution and -22.53bp for the second-best result realized with DDQN. In unreported experiments, we find that implementation shortfalls for selling and buying are very similar. As bid- and ask-side statistics of limit order books exhibit a large degree of symmetry (compare, for example, Biais, Hillion, & Spatt, 1995; Potters & Bouchaud, 2003; Schnaubelt et al., 2019), similar behaviors for buying and selling are unsurprising. We also observe that the relative performance improvement of strategies involving limit orders declines with increasing execution volumes. While using the PPO agent to execute 10 BTC results in a shortfall reduction of 36.93 percent over IM execution averaged over all three exchanges, trading 20 BTC (50 BTC) results in a relative improvement of 33.11 (27.17) percent.

We hypothesize that the superior performance of PPO might be due to several factors: First, in cases where the policy function is easier to approximate than the action-value function, PPO as a policy gradient method has an advantage over value-based algorithms such as DDQN (Sutton and Barto, 2018, p. 266). We conjecture that the action-value function might be difficult to approximate because of the high degree of noise in limit order data. Second, policy gradient methods are able to learn stochastic policies, which could, compared to the ϵ-greedy policy of DDQN, result in a better exploration of the action space. Third, PPO explicitly targets robustness, fast convergence and high data efficiency (Schulman et al., 2017b), which might represent a key advantage for applications with noisy financial data. As "the design of an ANN [artificial neural network] is more of an art than a science" (Zhang, Eddy Patuwo, and Y. Hu, 1998, p. 42) and we have not been able to tune the hyperparameters of DRL algorithms because of their computationally very expensive training, we conjecture that there might be parameter settings with even better performance.

The economic significance of shortfall differences between placement strategies becomes apparent when comparing them to typical excess returns of statistical arbitrage strategies. In comparison to the S&L strategy, PPO reduces average execution costs by roughly 2 bp. Taking the random forest-based cryptocurrency trading strategy of Fischer et al. (2019) as an example, mean returns per trade are at 3.8 bp after transaction costs. A reduction of transaction costs by 2 bp would increase mean excess returns to 5.8 bp, corresponding to a performance improvement of the strategy of over 50 percent. Hence, even relatively small reductions of execution costs may greatly impact the financial performance of trading strategies, especially when higher volumes are traded.

[14] Hence, the first validation period comprises the months July to September 2018. We use a total of four non-overlapping validation periods, i.e., train models in four runs for every setting of exchange, market and initial volume.

[15] To be precise, we do not select an observation as root observation during training or validation if observations are missing in the subsequent block of $T - 1$ contiguous time steps.

[16] We follow Nevmyvaka et al. (2006) and Hendricks and Wilcox (2014) with this approach. As additional robustness check, we evaluated model performance in terms of median realized shortfalls and find similar results.

Table 3
Comparison of execution strategies by realized shortfall. This table compares the performance of execution strategies for different exchanges, markets and initial execution volumes v_0 for out-of-sample test data. All results are averaged over four model runs with individual training and test data sets. We depict results for immediate (*IM*) and time-weighted market-order execution (*TW*), submit-and-leave execution (*S&L*), backwards-induction Q-learning (*BQL*), deep double Q-networks (*DDQN*) and proximal policy optimization (*PPO*). Panel A (B) shows the mean realized implementation shortfall including (excluding) exchange commissions, in basis points. Panel C shows the mean realized implementation shortfall including exchange commissions when adding an additional linear price impact of own orders.

Exchange	Market	Volume v_0	IM	TW	S&L	BQL	DDQN	PPO
Panel A: Mean relative shortfall, including all exchange commissions [bp]								
BitFinex	BTC/USD	10 BTC	-22.8517	-21.0325	-16.8143	-16.9482	-16.9093	-15.7672
	(485984 obs.)	20 BTC	-24.9760	-21.6577	-18.5294	-18.2759	-18.1973	-17.1093
		50 BTC	-30.0132	-23.3858	-22.9998	-21.5530	-21.3757	-21.1172
	ETH/BTC	100 ETH	-27.8538	-23.7508	-22.8057	-21.4498	-22.9759	-21.2470
	(482924 obs.)	200 ETH	-31.9487	-25.3424	-26.7186	-24.3196	-25.7515	-24.6528
	ETH/USD	100 ETH	-24.2229	-21.5278	-18.8991	-18.4362	-18.8688	-17.3880
	(478633 obs.)	200 ETH	-26.9632	-22.5119	-21.1788	-20.1197	-20.2406	-19.2230
Kraken	BTC/USD	10 BTC	-31.6315	-28.8502	-23.7581	-23.2141	-23.5195	-22.4810
	(435983 obs.)	20 BTC	-34.3564	-29.9004	-26.1868	-25.1410	-25.3096	-24.5956
		50 BTC	-40.9416	-32.3358	-32.1192	-30.0486	-30.1216	-30.1082
Coinbase	BTC/USD	10 BTC	-32.7858	-31.1540	-19.7012	-19.7012	-17.8253	-16.1073
	(518911 obs.)	20 BTC	-34.6456	-31.7403	-24.2416	-24.0410	-22.5267	-20.9857
		50 BTC	-39.4232	-33.2700	-32.3775	-30.5668	-30.5148	-29.4034
Panel B: Mean relative shortfall, excluding exchange commissions [bp]								
BitFinex	BTC/USD	10 BTC	-2.8576	-1.0346	-2.2899	-2.2513	-1.3583	-1.3373
	(485984 obs.)	20 BTC	-4.9863	-1.6610	-3.2626	-2.8805	-2.2582	-1.9664
		50 BTC	-10.0338	-3.3928	-6.4723	-4.6790	-4.3049	-4.2280
	ETH/BTC	100 ETH	-7.8695	-3.7583	-5.8203	-3.7579	-4.4381	-3.7656
	(482924 obs.)	200 ETH	-11.9727	-5.3531	-9.1389	-5.7677	-6.9593	-6.2346
	ETH/USD	100 ETH	-4.2314	-1.5308	-3.6133	-2.9864	-2.2410	-1.7288
	(478633 obs.)	200 ETH	-6.9772	-2.5169	-5.0705	-3.5393	-3.2318	-2.6883
Kraken	BTC/USD	10 BTC	-5.6462	-2.8576	-2.9150	-2.1588	-1.8293	-1.7620
	(435983 obs.)	20 BTC	-8.3782	-3.9106	-4.6408	-3.2633	-3.2253	-3.2507
		50 BTC	-14.9806	-6.3523	-9.4454	-7.3105	-7.4165	-7.4708
Coinbase	BTC/USD	10 BTC	-2.8001	-1.1598	-2.5853	-2.5853	-1.1749	-1.4980
	(518911 obs.)	20 BTC	-4.6670	-1.7493	-3.7709	-3.3530	-2.1598	-2.3942
		50 BTC	-9.4656	-3.2852	-7.8789	-5.2631	-4.8892	-5.5259
Panel C: Mean relative shortfall with additional linear price impact, including all exchange commissions [bp]								
BitFinex	BTC/USD	10 BTC	-22.8517	-21.2114	-16.8430	-16.9759	-16.8883	-15.8398
		50 BTC	-30.0142	-24.2804	-23.2188	-21.9753	-21.9431	-21.2618
	ETH/BTC	100 ETH	-27.8538	-23.7831	-22.8097	-21.4621	-22.4132	-21.5119
		200 ETH	-31.9487	-25.4071	-26.7275	-24.3526	-25.1146	-25.1344
	ETH/USD	100 ETH	-24.2229	-22.0213	-18.9831	-18.5697	-19.0458	-17.6040
		200 ETH	-26.9632	-23.4989	-21.3850	-20.5299	-20.8229	-19.7378
Kraken	BTC/USD	10 BTC	-31.6315	-28.9551	-23.7738	-23.2271	-23.4979	-22.5071
		50 BTC	-40.9416	-32.8603	-32.2306	-30.2653	-30.4159	-30.2457
Coinbase	BTC/USD	10 BTC	-32.7874	-32.0128	-19.9216	-19.9216	-18.5978	-16.5540
		50 BTC	-39.4232	-37.4323	-33.3077	-32.6026	-32.6841	-31.2493

5.1.2. *Influence of permanent price impact*

So far, our execution environment assumed a negligible permanent price impact of the agent's own orders. To check the validity of this assumption for our results, we now introduce an additional permanent price impact to our simulated limit order exchange, which also affects the agent's reward function during training.[17] Specifically, we implement the linear price impact model from Eq. 1 to add a permanent mid-price shift resulting from the agent's own orders and take the values from Table 1 for coefficient λ. To obtain the results in Panel C of Table 3, we retrain and evaluate all models in the modified environment. As expected, realized shortfalls increase in magnitude for most settings due to the additional price impact of own orders. The only general exception is the immediate strategy, which is not affected by the additional permanent price impact as all volume is executed in the first time step. In few cases, we find DDQN to achieve slightly better results than in the baseline environment, which might be due to the additional price impact leading to a less noisy reward signal and hence more successful training. Comparing the results to the baseline results of Panel A, we find that the ranking of execution strategies is not compromised by the additional price impact. This is also the case for larger initial volumes v_0, for which the additional price impact has the highest effect.

5.1.3. *The role of exchange commissions*

To study the role of exchange commissions for total shortfall results, Panel B of Table 3 reports raw shortfalls, i.e., realized shortfalls before exchange commissions, that arise due to price impact and limit order dynamics. Comparing these results to the results including exchange commissions (Panel A), we find that the larger part of total realized implementation shortfalls is caused by exchange commissions. For the smallest execution volume of 10 BTC, roughly ten percent of execution costs are due to the raw implementation shortfall. For larger volumes, the fraction of shortfall that is due to exchange commissions is still well above 50 percent. Hence, the avoidance of exchange commissions is a major factor in the reduction of total implementation shortfalls at all studied cryptocurrency exchanges. As exchange commissions are part of the agent's reward function, reinforcement learning models are trained to reduce the total implementation shortfall. Nevertheless, we find similar rankings of algorithms before and after the inclusion of exchange commissions. In many cases, PPO realizes the lowest magnitude in implementation shortfall before exchange commissions. There is one exception in the similarity of strategy rankings: Time-weighted execution with market orders (TW) yields the best re-

[17] We thank an anonymous referee for suggesting this analysis.

Table 4
Fraction of volume executed with limit orders. This table depicts the average fraction of volume executed with limit orders instead of market orders. We depict results for submit-and-leave execution (*S&L*), backwards-induction Q-learning (*BQL*), deep double Q-networks (*DDQN*) and proximal policy optimization (*PPO*), for different exchanges, markets and initial volumes v_0. Results for immediate and time-weighted execution are not shown as they execute all volume in market orders, thus exhibit a zero limit order fraction.

Exchange	Market	Volume v_0	S&L	BQL	DDQN	PPO
BitFinex	BTC/USD	10 BTC	0.5476	0.5303	0.4449	0.5570
		20 BTC	0.4733	0.4605	0.4061	0.4857
		50 BTC	0.3473	0.3126	0.2929	0.3111
	ETH/BTC	100 ETH	0.3015	0.2308	0.1462	0.2519
		200 ETH	0.2420	0.1448	0.1208	0.1582
	ETH/USD	100 ETH	0.4714	0.4550	0.3372	0.4341
		200 ETH	0.3892	0.3420	0.2991	0.3465
Kraken	BTC/USD	10 BTC	0.5157	0.4945	0.4310	0.5281
		20 BTC	0.4454	0.4122	0.3916	0.4655
		50 BTC	0.3326	0.3262	0.3295	0.3363
Coinbase	BTC/USD	10 BTC	0.4295	0.4295	0.4450	0.5130
		20 BTC	0.3176	0.3104	0.3211	0.3803
		50 BTC	0.1834	0.1565	0.1458	0.2041

sults in terms of raw shortfall, however looses its advantage after accounting for the elevated fees of market-order execution. In conclusion, we find that performance differences between strategies are driven by differences in realized exchange commissions. As commissions are higher for liquidity-taking orders at all studied exchanges, successful strategies are likely to achieve a higher probability of limit order execution.

5.1.4. *Fraction of limit order execution*

To analyze differences between strategies with respect to limit-order execution probabilities, we now calculate the fraction of volume that is executed with limit orders. We present results in Table 4 and make the following observations: First, with increasing initial volume v_0, the fraction of volume executed in limit orders decreases. We conjecture that liquidity is insufficient to execute larger volumes with limit orders over a relatively short time frame. Second, for the BTC/USD currency pair, the BitFinex exchange exhibits the highest fraction of limit orders. Taking into account the descriptive statistics of our data set (Table 1 and Table A.6 in the online appendix), BitFinex also has the tightest bid-ask spread and the highest trade volume and frequency, which might well explain a higher probability of limit order execution. Also, the reduction in the fraction of limit order execution with increasing volume is less pronounced for BitFinex than for Coinbase. Third, the S&L strategy, which submits orders at the best-ask price, executes a larger share of volume in limit orders than the BQL or DDQN strategies, and in some cases also more than the PPO agent. Yet, PPO execution still yields superior overall results in most of these cases, for example in executing 20 BTC at the Coinbase BTC/USD market (Panel A of Table 3). We conjecture that PPO compensates the lower limit order execution probability by posting limit orders at less aggressive prices, which are updated in every time step and potentially better reflect the expected probability of execution given the current market state.[18] In general, using a mix of limit and market orders allows to balance between unfavorable prices achieved with market orders and the uncertainty of execution of limit orders. The preference of the PPO agent for limit orders serves to avoid the additional taker-fees of market orders – a behavior is empirically observed in U.S. broker data (Battalio et al., 2016). Theoretical models suggest that the optimal mix of limit and market orders is based on a trader's information and the state of the limit order book (Cont & Kukanov, 2017; Foucault, 1999; Goettler et al., 2009).

5.1.5. *Robustness of empirical results*

To assess whether the superiority of the PPO strategy is driven by a single exceptional subperiod of data or beneficial hyperparameter setting, we perform several robustness checks. Table C.8 in the appendix depicts the average ranks of execution strategies in all validation subperiods by exchange and market. In three out of five exchange-market configurations, PPO achieves an average rank of 1.0, indicating that PPO always performs best, i.e., yields the lowest shortfall including commissions for all volume settings and in all four train-test splits of the data. Over all exchange-market-subperiod combinations, PPO achieves the highest rank in terms of total out-of-sample shortfall in 89.06 percent of cases. Next, we check whether the superior performance of PPO is due to a favorable choice of seed value or hyperparameter configuration. First, we train the PPO agent 10 times with consecutive seed values of the random number generator for all subperiods, and evaluate the distribution of out-of-sample shortfalls. From the results in Panel A of Table C.9 in the online appendix, we observe that a variation of seed value leads to small variations in shortfalls only. Second, we investigate the sensitivity of results to changes of the hyperparameter configuration. To this end, we retrain the PPO agent with different numbers of neurons in the hidden layers and different trajectory lengths (Panel B of Table C.9 in the online appendix). We observe that, despite larger variations than obtained from the seed study, results are still fairly similar to the baseline results that use the algorithm's default hyperparameters.

5.2. *Importance of feature sets*

How does PPO achieve a superior performance in minimizing execution shortfalls? To further elaborate on this question, we now analyze which information from the limit order book provides the highest value for the decision making of the PPO strategy. To this end, we retain the PPO algorithm in different environments, each of which omits a single feature, and calculate respective mean total realized shortfalls for the BTC/USD currency pair for selling an initial volume of $v_0 = 20BTC$. We calculate a feature importance score from the inverse distance to the total shortfall using the full feature set. Table 5 presents the results for different exchanges and allows for the following observations:

First, we find that the most important features are related to the current shape of the limit order book. For the BitFinex exchange, by far the most important features are related to queue imbalances, i.e., volume differences between the bid and ask sides of the limit order book (*BO-IMBAL* and *Q-IMBAL*(N_p)). Second and

[18] We will further investigate the limit order prices chosen by the agent in Section 5.3.

Table 5
Importance of features. This table depicts results from an analysis on the importance of features on proximal policy optimization performance. We retrain the PPO agent in environments with different features and calculate respective mean total realized shortfalls (columns *Total shortfall*) for the BTC/USD currency pair traded at the indicated exchanges for selling an initial volume of $v_0 = 20BTC$. Columns *Importance score* calculate a feature importance score from the inverse distance to the total shortfall using the full feature set.

	Total shortfall			Importance score		
Feature(s)	BitFinex	Kraken	Coinbase	BitFinex	Kraken	Coinbase
TC-IMBAL	-18.1673	-25.2428	-24.0553	0.0605	0.0757	0.0331
TV-IMBAL	-18.4534	-25.2474	-24.0613	0.0476	0.0752	0.0330
BO-IMBAL, *Q-IMBAL*(N_p)	-17.3317	-25.0991	-22.0463	0.2877	0.0973	0.0958
VOL-BID, *CVOL-BID*(N_p)	-17.5296	-25.1236	-22.4134	0.1523	0.0928	0.0712
VOL-ASK, *CVOL-ASK*(N_p)	-17.9558	-25.2673	-22.8120	0.0756	0.0729	0.0556
VOLA	-18.2896	-25.3367	-24.0051	0.0542	0.0661	0.0336
DRIFT	-18.1472	-25.2615	-23.8873	0.0617	0.0736	0.0350
LC-BID(V)	-17.6075	-24.7595	-21.1852	0.1285	0.2988	0.5093
LC-ASK(V)	-18.0681	-25.2360	-22.0444	0.0668	0.0765	0.0960
BA-SPREAD	-18.0901	-25.2836	-23.7010	0.0653	0.0712	0.0374

third important are features on cumulated order volumes (*VOL-BID* and *CVOL-BID*(N_p)) and liquidity costs related to the bid side (*LC-BID*(V)). For the exchanges Kraken and Coinbase, the same feature sets appear among the most important ones. However, liquidity costs related to the bid side (*LC-BID*(V)) now lead to the largest reduction in shortfalls. Second, we find bid-side features to be more important than ask-side features irrespective of the exchange considered. For the BitFinex exchange, cumulated bid volume features (*VOL-BID* and *CVOL-BID*(N_p)) result in a total shortfall of -17.53bp, compared to a value of -17.96bp for the ask-side features (*VOL-ASK* and *CVOL-ASK*(N_p)). Third, we find a further reduction of total shortfall when providing all features to the agent, as the PPO agent seems to be able to successfully exploit a larger feature set in learning suitable policies. For example, with the best single feature set for the Coinbase exchange, bid-side liquidity (*LC-BID*(V)), total shortfall is at -21.1852 bp, which is further improved to -20.9857 bp (see Table 3) when providing all features to the agent.

5.3. Exploring the agent's decision making

Finally, we analyze the impact of feature variables on the decision making of the PPO agent. To this end, we evaluate the agent's action conditional on the value of individual features. Specifically, we compute average action values for all observations with feature values in the respective quantile intervals. Fig. 1 plots the mean difference of the chosen limit price to the current best-ask price, in units of the tick size, as a function of a feature's value for selling different volumes at the BTC/USD market. Negative values indicate a more aggressive order posting, i.e., setting the limit price below the current best-ask price.

Looking first at features related to the current volume present in the first ten price steps of the order book (top row of Fig. 1), we find that sell pressure, i.e., an overweight of order volume in the ask side of the order book, gives rise to a more aggressive placement of the agent's own sell order. By contrast, positive values of the queue imbalance ($Q\text{-}IMBAL(10) > 0$), arising from larger bid than ask volumes, is followed by less aggressive orders. A similar behavior is evident when looking at the volumes at both sides of the order book separately (*CVOL-BID*(10) and *CVOL-ASK*(10)), with a larger influence of the opposite (bid) side of the order book. We can understand these results by interpreting the queue imbalance as a short-term predictor of mid-price returns (compare, for example, Cao et al., 2009; Gould & Bonart, 2016): The smaller the value of the queue imbalance, the higher is the supply overweight in the order book, which is indicative of a downshift of the mid price over the subsequent time period. Consequently, we find the agent to place orders more aggressively to ensure execution. We find a similar influence of the cost of immediate execution on the agent's decision making (middle row of Fig. 1), which could likely be due to a high degree of correlation between features that are based on the current order book.

The chosen action is fairly independent of the bid-ask spread (*BA-SPREAD*). This could be a consequence of a high level of noise in this variable, generated for example by frequent jumps of best-ask and best-bid prices, which renders it an unattractive feature for the model. The effect of the trade count imbalance (*TC-IMBAL*) and the trade volume imbalance (*TV-IMBAL*) is similar to the effect of the queue imbalance, however with a lower magnitude. Higher levels of past realized mid-price volatility (*VOLA*) slightly decrease order aggressiveness, potentially because less aggressive orders have a higher probability of execution in times of high volatility.

Generally, the shape of the feature-action dependence is – up to a shift towards more aggressive orders for larger volumes – fairly similar across different levels of executed volume v_0. While conditional actions for $10BTC$ and $20BTC$ are similar and the larger volume of $20BTC$ exhibits only a slight tendency towards more aggressive orders, PPO places significantly more aggressive orders for a volume of $50BTC$. Comparing the relative price tick of the market at the average trade price (0.14 bp at 7000 BTC, compare Table 1 and Table A.6 in the online appendix) to average bid-ask spreads (0.679 bp, Table A.7), we find that typical order placement occurs a few price ticks inside the bid-ask spread.

The PPO agent's placement strategy is in agreement with literature on the order aggressiveness in placement strategies in established markets (see Tripathi Abhinava and Dixit, 2020, for a survey of works). By interacting with the limit order execution environment, PPO seems to learn strategies similar to what is found in empirical studies of order flows. For example, Griffiths et al. (2000) and Degryse et al. (2005) find that aggressive sell orders tend to occur when the bid-ask spread is narrow and depth on the other side of the order book is small. While we find that PPO's decision making does not depend on the bid-ask spread[19], we find a qualitatively similar dependence on current bid/ask order volumes. Subsequent results of Lehalle and Mounjid (2017) demonstrate that a trader's decision depends on order book imbalance. Our results show that the PPO agent learns a strategy for cryptocurrency limit order placement that is highly dependent on order book imbalance, hence seems to pick up strategies similar to those of real-world traders. In summary, we find the order placement of the PPO agent to concur with economic rationale, lead-

[19] We conjecture that this might be due to the high level of noise in the bid-ask spread variable.

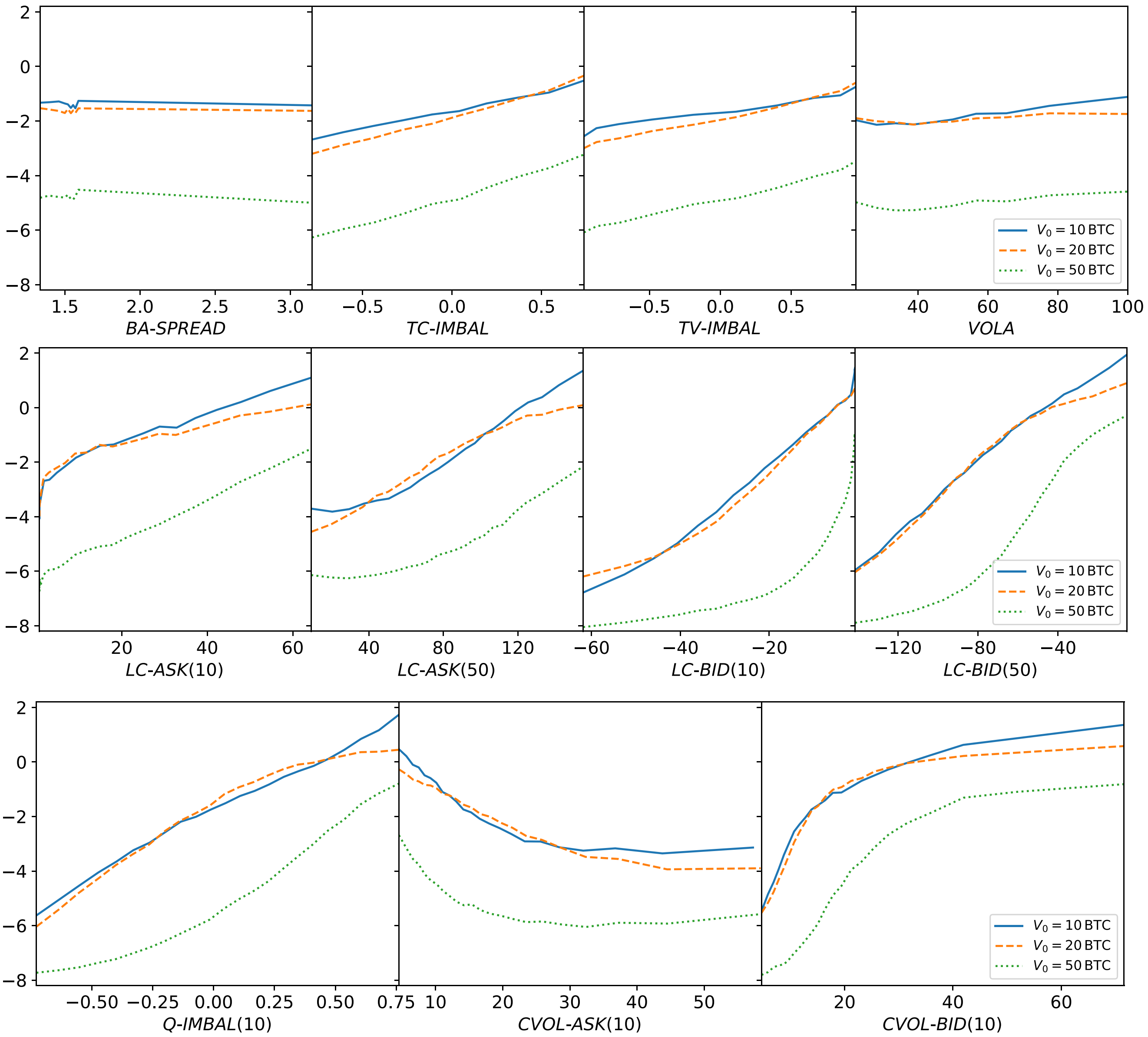

Fig. 1. Action limit price conditional on feature value. This figure depicts the limit price selected by proximal policy optimization conditional on feature values for selling initial volumes of $v_0 = 10, 20, 50BTC$, for the BitFinex BTC/USD market. The limit price (vertical axis) is shown relative to the current best-ask price in units of the tick size. Feature values are shown prior to rolling-window standardization. The top row depicts features related to the limit order volume in the first ten steps of the limit order book ($CVOL\text{-}BID(10)$, $CVOL\text{-}ASK(10)$: cumulated order volume in the ten best-bid/best-ask steps, $Q\text{-}IMBAL(10)$: queue imbalance), the middle row shows features related to current levels of liquidity costs ($LC\text{-}BID(V)$, $LC\text{-}ASK(V)$: cost of immediate execution for a volume of V, in basis points). The bottom row displays the bid-ask spread relative to the mid price (*BA-SPREAD*, in basis points), past trade count and volume imbalances (*TC-IMBAL*, $TV\text{-}IMBAL$) as well as past realized mid-price volatility ($VOLA$).

ing to more aggressive order placement in anticipation of disadvantageous movements of the mid price. By interacting with the simulated limit order exchange during training, it replicates order placement strategies similar to empirically observed behavior from real traders at established markets

6. Conclusion

With this paper, we demonstrate that state-of-the-art deep reinforcement learning is able to successfully learn superior order placement strategies to optimize execution. Execution optimization is highly relevant for both professional asset managers and private investors as execution quality affects portfolio performance at economically significant levels and is under regulatory supervision.

We contribute to the existing literature in the following ways: First, we design a problem-specific execution environment to train and test deep reinforcement learning agents at the task of optimal limit order placement. To this end, we introduce a reward function based on the implementation shortfall as well as exchange fees, hand-crafted market state features and a virtual data-based limit order exchange. Second, we compare the performance of state-of-the-art deep reinforcement learning algorithms to several benchmarks. While this study deliberately uses high-frequency data with 3.5 million order book states from several cryptocurrency exchanges, their nearly identical operating principles when compared to established exchanges provide a good argument for the generality of both the methodology and empirical results. We find proximal policy optimization to reliably learn superior order placement strategies, which reduce average total implementation shortfalls by up to 36.93 percent over an execution in a single market order. In light of the relevance and magnitude of price impact as well as the importance of exchange fees for overall trading costs (see, e.g., Edelen et al., 2013), these reductions are economically highly significant. Third, we shed light into the black-box of deep reinforcement learning and challenge the underlying economic rationale. By interacting with the simulated limit order exchange during training, PPO learns to optimize cryptocurrency limit order placement with strategies known empirically from established exchanges. The agent uses a mix of market and limit orders, where the latter are preferred to avoid elevated taker-fees. Its placement strategy becomes more aggressive in anticipation of lower limit order execution probabilities, indicated for example by order imbalances.

Our study points to several interesting directions for future research: First, our methodology may be used in future empirical studies applying deep reinforcement learning for optimal order placement for other assets, for example stocks. Second, another interesting extension would be to directly learn suitable feature representations from the data, for example with convolutional neural networks or the neural network architecture of Sirignano (2019). Third, extensions to adjacent problems of practical relevance, such as routing orders to multiple exchanges or deciding between further order types, could be studied.

Overall, we have provided a self-contained framework to leverage deep reinforcement learning for optimal order placement and successfully demonstrated its feasibility in a large-scale empirical study – representing a potential starting point for future research or practical applications addressing economic and regulatory requirements.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2021.04.050

References

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., … Zhang, L. (2019). Solving Rubik's Cube with a robot hand. *arXiv Preprint, 1910.07113*.

Almgren, R., & Chriss, N. (2001). Optimal execution of portfolio transactions. *The Journal of Risk, 3*(2), 5–39. https://doi.org/10.21314/JOR.2001.041.

Anand, A., Irvine, P., Puckett, A., & Venkataraman, K. (2012). Performance of institutional trading desks: An analysis of persistence in trading costs. *The Review of Financial Studies, 25*(2), 557–598. https://doi.org/10.1093/rfs/hhr110.

Atsalakis, G. S., Atsalaki, I. G., Pasiouras, F., & Zopounidis, C. (2019). Bitcoin price forecasting with neuro-fuzzy techniques. *European Journal of Operational Research, 276*(2), 770–780. https://doi.org/10.1016/j.ejor.2019.01.040.

Bao, W., & Liu, X.-y. (2019). Multi-agent deep reinforcement learning for liquidation strategy analysis. *arXiv Preprint, 1906.11046*.

Battalio, R., Corwin, S. A., & Jennings, R. (2016). Can brokers have it all? On the relation between make-take fees and limit order execution quality. *The Journal of Finance, 71*(5), 2193–2238.

Bayraktar, E., & Ludkovski, M. (2014). Liquidation in limit order books with controlled intensity. *Mathematical Finance, 24*(4), 627–650. https://doi.org/10.1111/j.1467-9965.2012.00529.x.

Ben-Rephael, A., & Israelsen, R. D. (2018). Are some clients more equal than others? An analysis of asset management companies execution costs. *Review of Finance, 22*(5), 1705–1736. https://doi.org/10.1093/rof/rfx043.

Bertsimas, D., & Lo, A. W. (1998). Optimal control of execution costs. *Journal of Financial Markets, 1*(1), 1–50. https://doi.org/10.1016/S1386-4181(97)00012-8.

Biais, B., Hillion, P., & Spatt, C. (1995). An empirical analysis of the limit order book and the order flow in the Paris Bourse. *The Journal of Finance, 50*(5), 1655–1689.

Calvez, A. l., & Cliff, D. (2018). Deep learning can replicate adaptive traders in a limit-order-book financial market. *arXiv Preprint, 1811.02880*.

Cao, C., Hansch, O., & Wang, X. (2009). The information content of an open limit-order book. *Journal of Futures Markets, 29*(1), 16–41.

Cartea, A., & Jaimungal, S. (2015). Optimal execution with limit and market orders. *Quantitative Finance, 15*(8), 1279–1291. https://doi.org/10.1080/14697688.2015.1032543.

Cartea, A., Jaimungal, S., & Penalva, J. (2015). *Algorithmic and high-frequency trading*. Cambridge, UK: Cambridge University Press.

Cheng, A. T. (2017). AI jumps into dark pools. https://www.institutionalinvestor.com/article/b15yx290rz5pcz/ai-jumps-into-dark-pools visited 2020-03-14.

Consilium Crypto Inc. (2020). Order recommendation system. https://consiliumcrypto.ai/order-recommendation-system/ visited 2021-02-17.

Cont, R., & Kukanov, A. (2017). Optimal order placement in limit order markets. *Quantitative Finance, 17*(1), 21–39. https://doi.org/10.1080/14697688.2016.1190030.

Cont, R., Kukanov, A., & Stoikov, S. (2014). The price impact of order book events. *Journal of Financial Econometrics, 12*(1), 47–88.

Daberius, K., Granat, E., & Karlsson, P. (2019). Deep execution – value and policy based reinforcement learning for trading and beating market benchmarks. *SSRN Scholarly Paper, 3374766*.

Danielsson, J., & Payne, R. (2001). Measuring and explaining liquidity on an electronic limit order book: Evidence from Reuters D2000-2. *SSRN Scholarly Paper, 276541*.

Degryse, H., Jong, F. D., Ravenswaaij, M. V., & Wuyts, G. (2005). Aggressive orders and the resiliency of a limit order market. *Review of Finance, 9*(2), 201–242.

Edelen, R., Evans, R., & Kadlec, G. (2013). Shedding light on "invisible" costs: Trading costs and mutual fund performance. *Financial Analysts Journal, 69*(1), 33–44. https://doi.org/10.2469/faj.v69.n1.6.

Fischer, T. G. (2018). Reinforcement learning in financial markets – a survey. *Discussion Papers in Economics 12/2018*. Erlangen/Nürnberg, Germany: Friedrich-Alexander-Universität.

Fischer, T. G., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research, 270*(2), 654–669.

Fischer, T. G., Krauss, C., & Deinert, A. (2019). Statistical arbitrage in cryptocurrency markets. *Journal of Risk and Financial Management, 12*(1), 31.

Foucault, T. (1999). Order flow composition and trading costs in a dynamic limit order market. *Journal of Financial Markets, 2*(2), 99–134. https://doi.org/10.1016/S1386-4181(98)00012-3.

Goettler, R. L., Parlour, C. A., & Rajan, U. (2009). Informed traders and limit order markets. *Journal of Financial Economics, 93*(1), 67–87. https://doi.org/10.1016/j.jfineco.2008.08.002.

Gomber, P., Schweickert, U., & Theissen, E. (2015). Liquidity dynamics in an electronic open limit order book: An event study approach. *European Financial Management, 21*(1), 52–78.

Gopikrishnan, P., Plerou, V., Gabaix, X., & Stanley, H. E. (2000). Statistical properties of share volume traded in financial markets. *Physical Review E, 62*(4), 4493–4496.

Gould, M. D., & Bonart, J. (2016). Queue imbalance as a one-tick-ahead price predictor in a limit order book. *Market Microstructure and Liquidity, 2*(2), 1650006. https://doi.org/10.1142/S2382626616500064.

Gould, M. D., Porter, M. A., Williams, S., McDonald, M., Fenn, D. J., & Howison, S. D. (2013). Limit order books. *Quantitative Finance, 13*(11), 1709–1742.

Griffiths, M. D., Smith, B. F., Turnbull, D. A. S., & White, R. W. (2000). The costs and determinants of order aggressiveness. *Journal of Financial Economics, 56*(1), 65–88. https://doi.org/10.1016/S0304-405X(99)00059-8.

Ha, Y., & Zhang, H. (2020). Algorithmic trading for online portfolio selection under limited market liquidity. *European Journal of Operational Research, 286*(3), 1033–1051. https://doi.org/10.1016/j.ejor.2020.03.050.

van Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double Q-learning. *arXiv Preprint, 1509.06461*.

Hendricks, D., & Wilcox, D. (2014). A reinforcement learning extension to the Almgren-Chriss model for optimal trade execution. In *IEEE Conference on Computational Intelligence for Financial Engineering & Economics* (pp. 457–464). https://doi.org/10.1109/CIFEr.2014.6924109.

Henrique, B. M., Sobreiro, V. A., & Kimura, H. (2019). Literature review: Machine learning techniques applied to financial market prediction. *Expert Systems with Applications, 124*, 226–251. https://doi.org/10.1016/j.eswa.2019.01.012.

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., & Wu, Y. (2018). Stable baselines. https://github.com/hill-a/stable-baselines.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. *arXiv Preprint, 1412.6980*.

Kissell, R., Glantz, M., & Malamut, R. (2004). A practical framework for estimating transaction costs and developing optimal trading strategies to achieve best execution. *Finance Research Letters, 1*(1), 35–46. https://doi.org/10.1016/S1544-6123(03)00004-7.

Lehalle, C.-A., & Mounjid, O. (2017). Limit order strategic placement with adverse selection risk and the role of latency. *Market Microstructure and Liquidity, 03*(01), 1750009. https://doi.org/10.1142/S2382626617500095.

Makarov, I., & Schoar, A. (2020). Trading and arbitrage in cryptocurrency markets. *Journal of Financial Economics, 135*(2), 293–319. https://doi.org/10.1016/j.jfineco.2019.07.001.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., … Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning: 48* (pp. 1928–1937).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv Preprint, 1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236.

Nagy, C., & Gellasch, T. (2018). Better 'best execution': An overview and assessment. In W. Mattli (Ed.), *Global algorithmic capital markets: High frequency trading, dark pools, and regulatory challenges*. Oxford University Press.

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on*

Machine Learning. In *ICML '06* (pp. 673–680). https://doi.org/10.1145/1143844.1143929.

Nevmyvaka, Y., Kearns, M., Papandreou, A., & Sycara, K. (2005). Electronic trading in order-driven markets: Efficient execution. In *Proceedings of the 7th IEEE International Conference on E-Commerce Technology* (pp. 190–197). https://doi.org/10.1109/ICECT.2005.42.

Ning, B., Ling, F. H. T., & Jaimungal, S. (2018). Double deep Q-learning for optimal execution. *arXiv Preprint, 1812.06600*.

Noonan, L. (2017). JPMorgan develops robot to execute trades. https://www.ft.com/content/16b8ffb6-7161-11e7-aca6-c6bd07df1a3c.

Obizhaeva, A. A., & Wang, J. (2013). Optimal trading strategy and supply/demand dynamics. *Journal of Financial Markets, 16*(1), 1–32. https://doi.org/10.1016/j.finmar.2012.09.001.

Perold, A. F. (1988). The implementation shortfall: Paper versus reality. *Journal of Portfolio Management, 14*(3), 4–9.

Plerou, V., Gopikrishnan, P., Gabaix, X., & Stanley, H. E. (2002). Quantifying stock-price response to demand fluctuations. *Physical Review E, 66*(2), 027104. https://doi.org/10.1103/PhysRevE.66.027104.

Potters, M., & Bouchaud, J.-P. (2003). More statistical properties of order books and price impact. *Physica A: Statistical Mechanics and its Applications, 324*(1–2), 133–140.

Ranaldo, A. (2004). Order aggressiveness in limit order book markets. *Journal of Financial Markets, 7*(1), 53–74.

Rantil, A., & Dahlen, O. (2018). *Optimized trade execution with reinforcement learning*. Sweden: Linkoeping University Master's thesis.

Roşu, I. (2012). Order choice and information in limit order markets. In F. Abergel, J.-P. Bouchaud, T. Foucault, C.-A. Lehalle, & M. Rosenbaum (Eds.), *Market microstructure* (pp. 41–60). Oxford, UK: John Wiley & Sons Ltd. http://doi.wiley.com/10.1002/9781118673553.ch2

Schnaubelt, M., Fischer, T. G., & Krauss, C. (2020). Separating the signal from the noise – financial machine learning for Twitter. *Journal of Economic Dynamics and Control, 114*, 103895.

Schnaubelt, M., Rende, J., & Krauss, C. (2019). Testing stylized facts of Bitcoin limit order books. *Journal of Risk and Financial Management, 12*(1), 25.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2017a). Trust region policy optimization. *arXiv Preprint, 1502.05477*.

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

Shah, D., & Zhang, K. (2014). Bayesian regression and Bitcoin. In *52nd Annual Allerton Conference on Communication, Control, and Computing* (pp. 409–414). https://doi.org/10.1109/ALLERTON.2014.7028484.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., … Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science, 362*(6419), 1140–1144. https://doi.org/10.1126/science.aar6404.

Sirignano, J. A. (2019). Deep learning for limit order books. *Quantitative Finance, 19*(4), 549–570. https://doi.org/10.1080/14697688.2018.1546053.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd). Cambridge, MA, USA: MIT Press.

Tripathi Abhinava, V., & Dixit, A. (2020). Limit order books: A systematic review of literature. *Qualitative Research in Financial Markets, 12*(4), 505–541. https://doi.org/10.1108/QRFM-07-2019-0080.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2020). Using deep learning for price prediction by exploiting stationary limit order book features. *Applied Soft Computing, 93*, 106401. https://doi.org/10.1016/j.asoc.2020.106401.

Tsoukalas, G., Wang, J., & Giesecke, K. (2017). Dynamic portfolio execution. *Management Science, 65*(5), 2015–2040. https://doi.org/10.1287/mnsc.2017.2865.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. UK: King's College, University of Cambridge PhD Thesis.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, 8*(3), 229–256. https://doi.org/10.1007/BF00992696.

Zhang, G., Eddy Patuwo, B., & Y. Hu, M. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting, 14*(1), 35–62. https://doi.org/10.1016/S0169-2070(97)00044-7.