Applied Energy 314 (2022) 118912

Contents lists available at ScienceDirect

Applied Energy

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

Statistical arbitrage trading on the intraday market using the asynchronous advantage actor–critic method[☆]

Sumeyra Demir[a,*], Bart Stappers[b], Koen Kok[a], Nikolaos G. Paterakis[a]

[a] *Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands*
[b] *Scholt Energy, Valkensvard, The Netherlands*

A R T I C L E   I N F O

Keywords:
Algorithmic trading
Actor–critic
A3C
Behaviour cloning
Deep reinforcement learning
Intraday markets
Non-physical virtual trader
Single intraday coupled market
Statistical arbitrage

A B S T R A C T

In this paper, we focus on statistical arbitrage trading opportunities involving the continuous exploitation of price differences arising during an intraday trading period with the option of closing positions on the balancing market. We aim to maximise the reward–risk ratio of an autonomous trading strategy. To find an optimal trading policy, we propose utilising the asynchronous advantage actor–critic (A3C) algorithm, a deep reinforcement learning method, with function approximators of two-headed shared deep neural networks. We enforce a risk-constrained trading strategy by limiting the maximum allowed position, and conduct state engineering and selection processes. We introduce a novel reward function and goal-based exploration, i.e. behaviour cloning. Our methodology is evaluated on a case study using the limit order book of the European single intraday coupled market (SIDC) available for the Dutch market area. The majority of hourly products on the test set return a profit. We expect our study to benefit electricity traders, renewable electricity producers and researchers who seek to implement state-of-art intelligent trading strategies.

1. Introduction

1.1. Motivation and background

Arbitrage opportunities arise when price differences between markets emerge. Alternatively, they can also arise when prices deviate from their medium or long term mean. Arbitrage traders, otherwise referred to as virtual traders, try to profit from such price differences without exposing themselves to long term physical commitments. Several studies, such as [1–7], have shown that arbitrage trading can reduce price differences between electricity markets and increase market liquidity and efficiency while yielding profits for arbitrage traders. In this study, we investigate the profitability of arbitrage trading on the intraday market.

Due to the markets' unique properties, setting up arbitrage trades when trading electricity is often more difficult than when trading other assets. The volatility of electricity prices often surpasses the volatility of other asset classes [8]. Presently, the supply of energy from renewable sources, such as wind and solar, is difficult to predict. This uncertainty impacts the accuracy of electricity price forecasts, increasing the risk for traders. Unlike on most other financial markets, traders are forced to close their open positions at the end of the trading session at volatile market prices; because electricity demand must always match electricity supply across the grid. Short trading windows, during which trades can be executed, complicate the optimisation of trading opportunities. Efficient trade execution is thus critical.

An arbitrage trading strategy employing mathematical and autonomous systems to make trades is termed statistical arbitrage. Trading decisions, namely buying, selling and holding, need to be automated when exploiting statistical arbitrage to optimise the risk-reward ratio. Deep reinforcement learning (DRL), employed by [9,10] to solve similar decision processes, can be used to automate statistical arbitrage trading decisions. Elaborating, various DRL methods, such as value-based, policy-based and actor–critic, can be used because of their consideration of future outcomes when making decisions and their potential to solve sequential decision and control problems.

Actor–critic methods combine the benefits of both value-based and policy-based methods and suffer from fewer shortcomings, as shown in [11,12]. Policy-based methods, for instance, suffer from high variance during back-propagation. Actor–critic methods reduce this variance. Combining actor–critic methods with generalised advantage estimation further reduces the variance of gradient updates [11]. The most successful algorithm in the space of advantage actor–critic methods

☆ This research project is funded by Scholt Energy.
* Corresponding author.
E-mail address: s.demir@tue.nl (S. Demir).

https://doi.org/10.1016/j.apenergy.2022.118912
Received 18 January 2022; Received in revised form 17 February 2022; Accepted 5 March 2022
Available online 22 March 2022

has been shown to be the asynchronous advantage actor–critic (A3C) method [12].

The performance of A3C depends on: state engineering and selection processes to increase the performance of function approximators; the choice of reward function to motivate agents; behaviour cloning to force exploration; the risk-constrained trading strategy to account for the risk and market challenges. Consequently, employing domain knowledge is critical to successfully deploying the algorithm.

A3C has thus far been successfully applied to equity trading on the stock markets [13,14] and wind energy trading on the reserve electricity markets [15]. To the best of our knowledge, the use of A3C, in the context of statistical arbitrage trading on the intraday market, has not been explored yet. Considering the above-mentioned advantages of A3C and similar successful implementations of A3C across other markets, it should be possible to successfully apply A3C to statistical arbitrage trading on the intraday market.

1.2. Relevant literature

Most arbitrage trading, or virtual bidding, studies that have explored maximising the profits of purely financial traders, have focused on exploiting the price differences between the day-ahead and balancing markets. Among these studies, [16,17] implemented a stochastic optimisation approach to exploit the price difference. Similarly, [18] implemented a min–max two-level optimisation model, [19] a data-driven approach, [20] a machine-learning approach and [2] an online-learning algorithm. Each method has been shown to yield positive returns on United States markets.

Analysing studies that considered intraday markets, [21] explored the profitability of arbitraging between the intraday and balancing markets using a rule-based trading method. As part of the strategy, a short or long position was opened on the intraday market, before being closed on the balancing market. To decide whether to place an initial long or short position, [21] forecast demand. Imbalance volume was additionally forecast. If the forecast of imbalance volume was long, then a short position on the intraday market was taken and vice versa. By executing this trading rule every 30 min with a fixed 2 MW quantity for the British market, [21] was able to achieve annual revenues of £73407.

Not all evaluated intraday market trading strategies have centred on arbitrage trading. Some, such as [22–24], have focused on evaluating bidding strategies for asset-backed physical traders, i.e. energy producers. Others, such as [25–27], have focused on bidding strategies for storage device operators. Among these studies, [25,26] have implemented DRL methods. [25] evaluated how REINFORCE, a policy gradient DRL algorithm, could be used to optimise bidding strategies. On the German market, their proposed method surpassed the profitability of a rolling intrinsic policy by 17.8%. Boukas et al. [26] investigated a value function approximation method, an asynchronous distributed version of the fitted Q iteration, to maximise the total return. On the German market, their proposed method improved the profitability of a rolling intrinsic policy by 2.7%. A state selection process, however, was not conducted, and as a result, their study was limited because state engineering and selection processes are vital to function approximators in DRL methods. The optimal number of important states (features) leads to better forecasts and decisions.

1.3. Contributions and organisation

As the above review highlights, the literature on continuous intraday trading is growing; however, papers focusing purely on financial trading are still scarce. Given the potentials for increasing market stability, this paper focuses on developing a statistical arbitrage trading strategy. While a statistical arbitrage trading strategy has already been explored by [21] using a simple trading rule with forecasts, intelligent methods should be further developed and investigated. We utilise a DRL method implemented with the A3C algorithm to maximise the reward–risk ratio of the arbitrage trading strategy. An autonomous agent capable of trading continuously on the intraday market with the opportunity to close an outstanding position on the balancing market is developed. Summarising the three contributions of our study, to the best of our knowledge, we are the first to:

- utilise A3C algorithm in developing a risk-constrained arbitrage trading strategy on the intraday market,
- conduct state engineering and state selection processes for a DRL-based arbitrage trading strategy on the intraday market,
- propose a novel reward function and behaviour cloning (goal-based exploration) for a DRL-based arbitrage trading strategy on the intraday market.

Outlining the structure of this paper, the intraday market and arbitrage trading are introduced in Section 2. Details of our algorithmic trading method are provided in Section 3. In Section 4, our case study and its results are outlined. Finally, we conclude our study in Section 5.

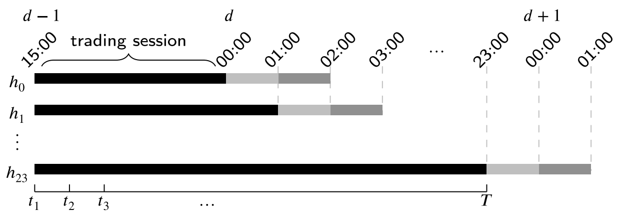

Fig. 1. The SIDC trading timeline for all hourly contracts, $h \in H$. Dark grey bars are delivery periods. Black bars are trading sessions starting at 15:00 on day $d-1$.

2. Arbitrage trading on the continuous intraday market

Fig. 1 shows trading timelines of all available hourly contracts, $h \in \{h_0, h_1, \dots h_{23}\} = H$, for the European single intraday coupled market (SIDC). Orders are continuously submitted to and cleared by the SIDC at any time during the trading session. Initiator traders who want to sell/buy a desired amount of electricity submit orders with ask/bid prices for $\forall h \in H$. The order matching engine follows a first-come first-served rule prioritising the best price and early submission. At time step $t \in [t_1, T]$, the best ask price p_t^a is the lowest of all available ask prices. Meanwhile, the best bid price p_t^b is the highest of all available bid prices. Note that their available quantities are q_t^a and q_t^b respectively. When a new order is submitted, orders are matched if the submitted bid price is higher than the best ask price, or if the submitted ask price is lower than the best bid price. The transaction occurs based on the lower quantity and any remaining quantity is kept in the order book.

Initiator traders can place multiple limit orders of varying sizes and prices. Theoretically, initiator traders have an infinite-dimensional action space. Aggressor traders, on the other hand, have a low-dimensional action space by trading available orders, i.e. $\{B, S, H\}$, where B buys the best ask order, S sells the best bid order and H holds. In this study, we develop agents which behave as aggressor traders.

The aggressor agent starts with a volume of $v_{t_0} = 0$. At any time, this agent can open a long position ($v_t > 0$) with a buy action or a short position ($v_t < 0$) with a sell action on the intraday market. As shown in Fig. 2, such an agent, for example, opens a long position and trades continuously $[B, B, H, S, H]$ (left) and $[B, H, S, S, H]$ (right). Assume q_t^a and q_t^b are 1 MW. This agent's total bought quantity $\sum q^a$ is then 2 MW (left) and 1 MW (right), and total sold quantity $\sum q^b$ is 1 MW (left) and 2 MW (right). Agent's total traded quantity $\sum q = min\{\sum q^a, \sum q^b\}$ is 1 MW, and the remaining quantity, i.e. final position, $\sum q^a - \sum q^b = v_T$ is 1 MW (left) and −1 MW (right).

Even though hundreds of transactions are executed during the trading session, some positions are not closed on the intraday market, either because a counter-party is not found or because closure on

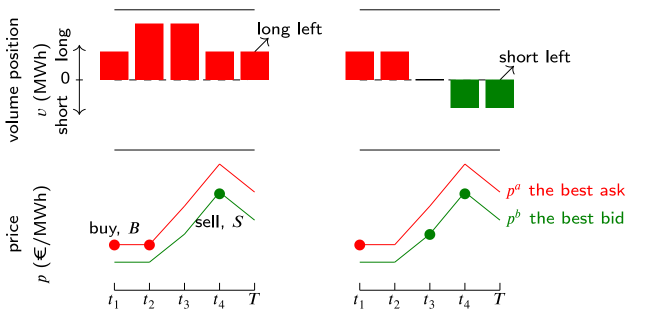

Fig. 2. Two examples of arbitrage aggressor trading opportunities on the intraday market for any $h \in H$. Trading timeline is discretised into 5 for a simple visualisation.

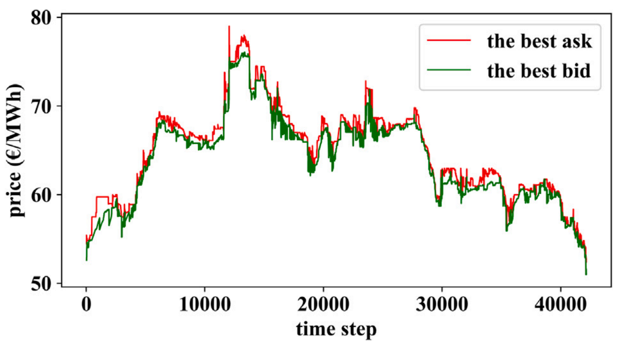

Fig. 3. The best ask/bid prices for h_9 on 01/09/2020.

the imbalance market is desired, such as in [21]. In these situations, open positions are cleared on the balancing market. Balancing market prices are announced for each quarter-hour. For hourly contracts, four quarterly balancing prices are averaged separately for short and long positions, p^{take} and p^{feed} respectively. Note that balancing market prices display significantly higher volatility ranging from negative prices to high positive prices.

The arbitrage trading strategy aims to maximise profit PnL. On the intraday market, the agent receives cash $\sum_t p_t^b \times q_t^b$ from selling electricity and pays cash $\sum_t p_t^a \times q_t^a$ for buying electricity. The cash made on the intraday market is then $C^I = \sum_t p_t^b \times q_t^b - \sum_t p_t^a \times q_t^a$. The remaining quantity $\sum_t q_t^a - \sum_t q_t^b = v_T$ is closed on the balancing market. PnL is thus calculated as in (1).

$$PnL = \begin{cases} C^I + v_T \times p^{take} - TC, & \text{if } \sum_t q_t^b > \sum_t q_t^a \\ C^I + v_T \times p^{feed} - TC, & \text{if } \sum_t q_t^b < \sum_t q_t^a \\ C^I - TC, & \text{otherwise} \end{cases} \tag{1}$$

where TC is the trading cost: $TC = 0.232 \times \sum q$. Note that to factor in the frictions associated with trading, we assume a market operator charges traders €0.232 for opening and subsequently closing a position of 1 MW.

More detailed information about the intraday market SIDC can be found in [28,29]. More information about the balancing markets can be found in [30].

3. Proposed method for arbitrage trading

3.1. Background: Reinforcement learning

In this section, we introduce the basics of reinforcement learning (RL) following [31]. RL aims to solve a sequential decision-making problem. In the context of RL, we define a decision as an *action* and the decision-maker as an *agent*. At time step $t \in [t_1, T]$, the agent executes its action a_t after receiving the state measurement $\mathbf{s}_t$. The *state* is a vector that holds all the necessary information about the environment in order to enable an optimal decision. Based on the agent's action, the agent receives an immediate scalar feedback with a *reward* signal $r_{t+1} \in \mathbb{R}$. The interaction described above proceeds iteratively for an *episode* starting from the initial state s_{t_1} to the terminal state s_T.

An RL system is typically designed to find an optimal behaviour by maximising the total reward over a given number of episodes. The behaviour, a mapping from states to actions, is defined by the *policy* function $\pi(a|\mathbf{s}) = \mathbb{P}[a_t = a \mid \mathbf{s}_t = \mathbf{s}]$. The expected total reward, which is accumulated until the terminal state s_T, is defined by the *state-value* function V. Under the policy π, $V_\pi(\mathbf{s}) = \mathbb{E}[R_t \mid \mathbf{s}_t = \mathbf{s}]$, where $R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T$ is the return and $\gamma \in [0,1]$ is the discount factor. The maximum value among all policies gives an optimal value function, and thus an optimal policy.

3.2. Discrete time steps

The continuous trading timeline is discretised, i.e. time step $t \in [t_1, T]$. Typically in the literature, fixed time intervals are used to separate time steps. For example, the time interval Δt between t and $t+1$ is set to 15 min in [26], 5 min in [32] and 1 min in [33]. This approach, however, is not sufficiently flexible for our arbitrage strategy. When a profitable ask/bid order is submitted to the intraday market, the agent should buy/sell this order sooner than its competitors. Additionally, the agent should adjust itself to trading sessions' quiet early hours using fewer time steps and busy final hours using more time steps.

To create flexible time steps, in this study the time interval Δt is defined by the order book revision number, i.e. update number for each change in the order book. Most of the updates, however, are not important and change neither the best ask price nor the best bid price. We drop these revisions and keep only important updates to improve training performance and reduce training time. Fig. 3, for example, is plotted after this process.

3.3. Actions

The statistical arbitrage trading strategy aims to maximise profit while minimising risk, i.e. the volatility of profit. The challenge of this strategy is rooted in the uncertainty of future intraday market prices, balancing prices and liquidity. To limit these uncertainties and risk, we impose a risk-constrained arbitrage trading strategy by constraining bought/sold quantity at each time step, position and total bought/sold quantity.

An agent can either avoid placing a trade (hold, H) or execute an existing order (buy, B) $q_t^a \le q^{max}$ or (sell, S) $q_t^b \le q^{max}$ MW of electricity at time step t. The agent with a maximum allowed position can buy/sell until its long/short position reaches v^{max}/v^{min}. Additionally, the agent with a maximum allowed total trading quantity can buy/sell until its total bought/sold quantity, $\sum q^a / \sum q^b$, reaches q^{high}. To implement the above conditions, we constrain the agent's action space, as shown in (2).

$$a_t \in \begin{cases} \{H\}, & \text{if } \sum q^a = q^{high} \text{ and } \sum q^b = q^{high} \\ \{B, H\}, & \text{if } v_t = v^{min} \text{ or } \sum q^b = q^{high} \\ \{S, H\}, & \text{if } v_t = v^{max} \text{ or } \sum q^a = q^{high} \\ \{B, S, H\}, & \text{otherwise} \end{cases} \tag{2}$$

3.4. Rewards

A reward function that motivates the agent to learn to avoid punishment is implemented. For a buy action, the best/worst ask order to buy is the lowest/highest ask across the entire trading session l^a/h^a and offers a buy reward of $0/-2$. For the remaining ask orders $l^a < p_t^a < h^a$, the agent receives negative buy rewards $r_t^b \in (-2, 0)$. The more expensive the asking price the lower the buy reward. Two buy thresholds, the highest bid price across the entire trading session $b^{th} = h^b$ and $b^{th} = p^{feed}$, are considered to separate possible profit and losses within the intraday market and between the intraday and balancing markets. The agent receives a buy reward of -1, if it buys at $p^a_t = b^{th}$. Above/below the threshold, the agent receives a buy reward that takes on values in $[-2,-1)/(-1,0]$. The immediate buy reward $r^b_t \in [-2,0]$ at time step t is calculated as (3).

$$r^b_t = 1/2\ \left(f^b(h^b, p^a_t) + f^b(p^{feed}, p^a_t)\right), \tag{3}$$

where

$$f^b(b^{th}, p^a_t) = \begin{cases} -1 - \left((p^a_t - b^{th})/(h^a - b^{th})\right), & \text{if } p^a_t > b^{th} \\ -\left((p^a_t - l^a)/(b^{th} - l^a)\right), & \text{otherwise} \end{cases}$$

For a sell action, the best/worst bid order to sell is the highest/lowest bid across the entire trading session h^b/l^b and offers a reward of $0/-2$. For the remaining bid orders, the more expensive the bid price the higher the sell reward $r^s_t \in (-2,0)$. Two sell thresholds, $s^{th} = l^a$ and $s^{th} = p^{take}$, are considered. At time step t, the immediate sell reward $r^s_t \in [-2,0]$ is calculated as (4).

$$r^s_t = 1/2\ \left(f^s(l^a, p^b_t) + f^s(p^{take}, p^b_t)\right), \tag{4}$$

where

$$f^s(s^{th}, p^b_t) = \begin{cases} -2 + \left((p^b_t - l^b)/(s^{th} - l^b)\right), & \text{if } p^b_t < s^{th} \\ -1 + \left((p^b_t - s^{th})/(h^b - s^{th})\right), & \text{otherwise} \end{cases}$$

Hold actions are evaluated using the opportunity cost of not buying/selling. If buy and sell decisions may lead to losses, the agent should hold and receive the best possible hold reward of 0. If a buy or sell decision may lead to profit, the agent should not hold and shall receive a hold reward $r^h_t \in [-1,0)$. The more profitable ask or bid orders are the lower the hold reward is as shown in (5).

$$r^h_t = \begin{cases} 0, & \text{if } r^b_t \ \&\ r^s_t < -1 \\ -1 - max\{r^b_t,\ r^s_t\}, & \text{otherwise} \end{cases} \tag{5}$$

Additionally, at the end of the trading session for each contract, the end of contract reward $r^e_t = PnL/100$ is provided to motivate pursuing profitable actions. This reward can also be used as a stop-loss threshold SLT. If losing €300 per contract is not desired, for instance, a condition $r_t < -3$ can be incorporated into training as an episode terminator, as further explained in Section 3.6.

3.5. States

The state vector $\mathbf{s}$ should contain the information required to predict the policy $\pi(\mathbf{s})$ or the state-value under the policy $V_\pi(\mathbf{s})$. Each component of the state vector is a state variable. Together, all variables should be a compact and sufficient representation of the environment. To predict π or V_π accurately, only the most important state variables should be used. We thus conduct a feature engineering process using domain knowledge and a feature selection process using automated feature elimination methods. Below, we first describe the types of state variables, subsequently, we provide possible engineered variables for each type of variable and finally we elaborate upon our state selection process.

States are split into internal and external, $\mathbf{s}_t = \mathbf{s}^i_t \cup \mathbf{s}^e_t$. While internal states reflect the agent's portfolio, external states come from the market environment. Example internal states for a trading agent are given below.

- *Agent:* volume position, cash, total traded volume, etc.

External states provide necessary information about the market and the market price drivers. The limit order book, which contains all submitted orders with their type, price, volume, and time information, offers the best insight for continuous market trading. The order book, however, cannot be used as it is for learning. Its inconsistent dimension size across time steps $t \in [t_1, T]$ is incompatible with A3C's function approximators which require consistent input size. A technique adopted in this study to achieve consistent dimension size for each time step is to extract some statistical information about the order book. Additionally, to reduce partial observability of the RL environment, lags of the best ask and bid prices, are considered. Example order book features calculated from the order book are given below.

- *Order book:* the best ask price, the best bid price, bid–ask spread, the first quantile of ask prices, the third quantile of bid prices, lags of best ask prices, lags of best bid prices, total ask quantities, total bid quantities, total quantity spread, the first quantile of cumulative ask quantities, the third quantile of cumulative bid quantities, the second quantile of cumulative bid quantities, etc.

Other important price drivers offering executed trade information, such as the highest traded price for a contract (high price), the lowest traded price (low price), the volume-weighted average price of trades (vwap), and the latest traded price (last price), are extracted from the trade book.

- *Trade book:* total traded volume, high price, low price, vwap, last price, etc.

Widely considered seasonal and time-based features are also created.

- *Time-based:* minutes to end of trading session, day, hour, holiday, etc.

Finally, some important forecasts assisting agents to make better decisions are considered.

- *Forecasts:* low price forecast, high price forecast, vwap forecast, wind forecast, etc.

Among all the aforementioned engineered external states, we select the most important features. This can be done with a recursive feature elimination method using, for example, a logistic regression model. Since this method is based on supervised learning $f(\mathbf{s}^e) = a$, the target $[a_1, a_T]$ needs to be created. Intuitively, this target is the sequence of desired trading actions. To make a profit on the intraday market, the agent should buy when p^a_t is lower than h^b and sell when p^b_t is higher than l^a. To make a profit between the intraday market and the balancing market, the agent should similarly buy when p^a_t is lower than p^{feed} and sell when p^b_t is higher than p^{take}. Our aforementioned conditions are given in (6).

$$a_t = \begin{cases} B, & \text{if } p^a_t < h^b \text{ or } p^a_t < p^{feed} \\ S, & \text{if } p^b_t > l^a \text{ or } p^b_t > p^{take} \\ H, & \text{otherwise} \end{cases} \tag{6}$$

Creating our multi-class target $[a_1, a_T]$, the feature selection process starts by removing correlated variables using the Pearson correlation method. The process continues by recursively pruning the least important variables until the performance of the logistic regression model worsens. The remaining variables are considered the most important external state variables.

3.6. Asynchronous advantage actor–critic (A3C)

As Fig. 4 shows, the A3C works with multiple agents which are connected through the global network. Agents interact with their environments and collect different experiences to increase overall exploration. For each thread, the *critic* updates the value and the *actor* updates the policy based on the feedback from the critic [31]. We use differentiable function approximators, namely deep neural networks (DNN), to estimate the value and the policy, e.g. $V_\pi(\mathbf{s}_t) \approx V_\pi(\mathbf{s}_t; \theta_v)$ and $\pi(a_t|\mathbf{s}_t) \approx \pi(a_t|\mathbf{s}_t; \theta_\pi)$ respectively. Note that we reserve θ_v and θ_π for

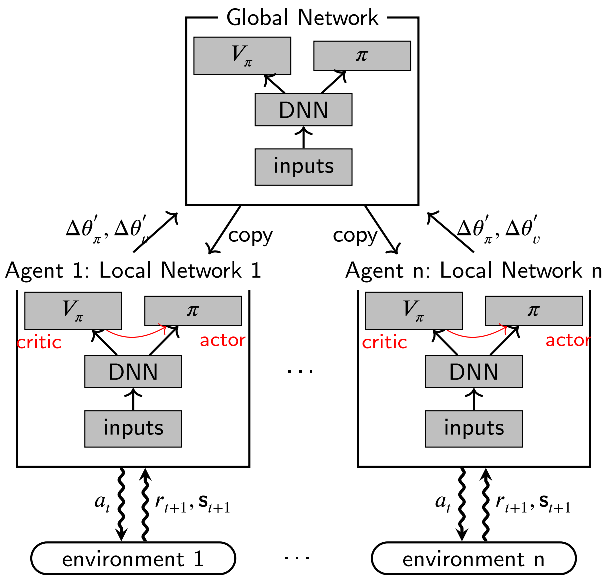

Fig. 4. The global network keeps the best value and policy of the most successful local network. The global network cannot receive all agents' gradients at once. Agents get in the queue. Deep neural networks (DNN) are used as function approximators. The red arrow represents the critic's feedback.

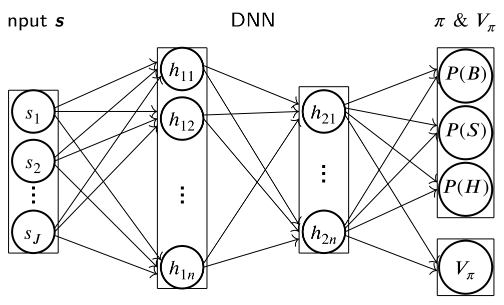

Fig. 5. Individual network architecture showing the input–output relationship. DNN has two hidden layers. The value V_π has a linear activation function and the policy π has a softmax function converting logits to probabilities.

the global network parameters and use θ'_v and θ'_π for the local network parameters.

Fig. 5 presents a detailed individual agent network. The input–output relationship is shown. Our policy is stochastic, i.e. $\pi(a|\mathbf{s}) = \mathbb{P}[a_t = a \mid \mathbf{s}_t = \mathbf{s}]$. The agent predicts a probability for each action based on the state inputs. If the agent predicts $\{P(B) = 0.6, P(S) = 0.1, P(H) = 0.3\}$, for example, it may with a 0.6 probability *exploit* by choosing its best action B, or with a 0.4 probability *explore* by choosing a random action of either S or H.

In the figure, the DNN – a shared two-headed two-layer neural network – is shown. Note that the network type or the number of layers can vary. Hyper-parameter tuning is implemented to select (1) the type of network for π and V_π, (i.e. whether to employ a two-headed shared network or separate networks), (2) the number of layers, (3) the number of neurons, (4) the type of hidden layer activation function, (5) the number of agents, (6) the type of optimiser, and (7) the hyper-parameters defined in Algorithm 1.

Algorithm 1 details the proposed A3C training procedure for each agent. At the start of an episode $e \in [1, e^{max}]$ gradients are reset (line 2), thread parameters are synchronised (line 3), an episode index multiplier is calculated (line 4), a discount rate is computed (line 5), an epsilon factor is calculated (line 6), the time step is initialised (line 7), and the first state is extracted (line 8). Experience is subsequently collected during an episode (lines 9 to 23). The action space is constrained accordingly to Section 3.3 (line 10).

Goal-based exploration is implemented to ensure that an agent learns effectively during the initial training stage. With goal-based exploration, i.e. behaviour cloning, the first $b < n$ number of agents are provided with exemplary behaviours during their initial episodes (line 12). One agent follows (7) to learn how to exploit arbitrage opportunities between the intraday and balancing markets.

$$a_t = \begin{cases} B, & \text{if } p^a_t < p^{feed} \;\&\; v_t < v^{max} \;\&\; \sum q^a < q^{high} \\ S, & \text{if } p^b_t > p^{take} \;\&\; v_t > v^{min} \;\&\; \sum q^b < q^{high} \\ H, & \text{otherwise} \end{cases} \tag{7}$$

Another agent follows (8) to learn arbitrage opportunities within the intraday market.

$$a_t = \begin{cases} B, & \text{if } p^a_t < h^b \;\&\; v_t < v^{max} \;\&\; \sum q^a < q^{high} \\ S, & \text{if } p^b_t > l^a \;\&\; v_t > v^{min} \;\&\; \sum q^b < q^{high} \\ H, & \text{otherwise} \end{cases} \tag{8}$$

Finally, another agent follows (9) pursuing high rewards based on a reward threshold $r^{th} \in [-1, 0)$.

$$a_t = \begin{cases} B, & \text{if } r^b_t > r^{th} \;\&\; v_t < v^{max} \;\&\; \sum q^a < q^{high} \\ S, & \text{if } r^s_t > r^{th} \;\&\; v_t > v^{min} \;\&\; \sum q^b < q^{high} \\ H, & \text{otherwise} \end{cases} \tag{9}$$

If agents show unsuccessful repetitive behaviours, exemplary behaviours using (9) with $r^{th} \in [-0.3, 0)$ can be further provided to agents during training to improve their exploration.

To ensure that agents efficiently explore, decayed/decaying epsilon greedy exploration [31] is additionally implemented. With decayed epsilon greedy exploration, agents are forced to explore with probability ϵ and exploit with probability $1 - \epsilon$ (lines 15 and 17). The ϵ is decreased slowly starting from the first episode to the last episode to ensure a higher exploration rate in the early episodes and a higher exploitation rate in the final episodes.

Algorithm 1 A3C: Pseudocode for each thread

```
Require: hyper-parameters, $\gamma_{start}, \gamma_{end}, \epsilon_{start}, \epsilon_{end}, t^{max}, e^{max}, \beta, SLT, b$; global parameters,
    $\theta_\pi, \theta_v$; thread parameters, $\theta'_\pi, \theta'_v$; global counter, $e \leftarrow 1$; and $t_{start} \leftarrow 1$
1: while $e < e^{max}$ do
2:     reset gradients $d\theta_\pi \leftarrow 0$, $d\theta_v \leftarrow 0$
3:     synchronise thread parameters $\theta'_\pi = \theta_\pi$, $\theta'_v = \theta_v$
4:     calculate episode index $e_i = 1 - (e/e^{max})$
5:     calculate discount $\gamma = (\gamma_{start} - \gamma_{end}) * e_i + \gamma_{end}$
6:     calculate epsilon $\epsilon = (\epsilon_{start} - \epsilon_{end}) * e_i + \epsilon_{end}$
7:     $t = t_{start}$
8:     get state $\mathbf{s}_t$
9:     while $\mathbf{s}_t \neq \mathbf{s}_T$ and $t - t_{start} < t^{max}$ do
10:        constrain the action space
11:        if $e < b$ then
12:            clone $a_t$ according to (7), (8) or (9)
13:        else
14:            if $\epsilon_{random} < \epsilon$ then
15:                explore: random $a_t$ according to $\pi(a_t|\mathbf{s}_t;\theta'_\pi)$
16:            else
17:                exploit: max $a_t$ according to $\pi(a_t|\mathbf{s}_t;\theta'_\pi)$
18:            end if
19:        end if
20:        receive reward $r_{t+1}$ and new state $\mathbf{s}_{t+1}$
21:        $t \leftarrow t + 1$
22:        $\mathbf{s}_t = \mathbf{s}_T$, if $r_t < SLT$
23:    end while
24:    $R_\pi = \begin{cases} 0, & \text{for terminal } \mathbf{s}_T \\ V_\pi(\mathbf{s}_t, \theta'_v), & \text{for non-terminal } \mathbf{s}_t \end{cases}$
25:    for $i \in \{t-1, \ldots t_{start}\}$ do
26:        $R_\pi \leftarrow r_i + \gamma R_\pi$
27:        $A_\pi \leftarrow R_\pi - V_\pi(\mathbf{s}_i,; \theta'_v)$
28:        $d\theta_\pi \leftarrow d\theta_\pi + \beta \nabla_{\theta'_\pi} \log \pi(a_i \mid \mathbf{s}_i ; \theta'_\pi)(A_\pi)$
29:        $d\theta_v \leftarrow d\theta_v + \beta\ \partial(A_\pi)^2/\partial\theta'_v$
30:    end for
31:    $e \leftarrow e + 1$
32:    update asynchronously $\theta_\pi$ using $d\theta_\pi$, and $\theta_v$ using $d\theta_v$
33: end while
```

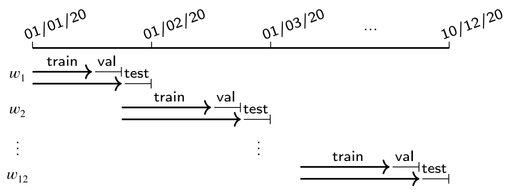

Fig. 6. Rolling windows for training, validation and test sets.

After selecting an action a_t, the agent receives its reward and next state (line 20). The agent collects experiences until either the terminal state, $\mathbf{s}_T$ or t^{max} are reached (line 22). The $\mathbf{s}_T$ is defined to be the last state in the training set. Early episode termination is employed when the agent, following its policy, reaches a predefined stop-loss threshold SLT, linked to the end of contract reward r^e. In this way, the number of contracts in each episode varies. We, however, can promptly update unsuccessful policies for the better and reduce training time.

To update the actor, critic and global network, the critic first estimates the value of a state V_π, line 24. Then, for each gradient step the value of the state–action R_π is calculated (line 26). Intuitively, this value is an improvement compared to V_π. The difference is called the *advantage* function A_π [12]. It can be estimated by the temporal difference (TD) error and offers information about the extra reward attainable when following the policy's actions (line 27). If A_π is positive, gradients of the actor and critic are pushed in that direction by the learning rate β. If A_π is negative, gradients are pushed in the opposite direction (lines 28 and 29). Using updated actor and critic gradients, the global network is also updated (line 32). The network weights are updated until a predefined maximum number of episodes e^{max} is reached.

Note that A3C is not without its weaknesses. Elaborating, it is very data-intensive, sensitive to hyper-parameter configurations, and may suffer from delayed rewards. Our architecture and design avoid and mitigate these shortcomings. Data size is increased by using more discrete time steps for each trading session. Hyper-parameters are optimised similar to any other deep learning approach. Finally, the long delays are averted by using a reward distribution for immediate rewards.

4. Numerical case study

4.1. Data

The primary data sources are the limit order book and trade book of the hourly SIDC contracts available for the Dutch market area. The order and trade books of SIDC are not publicly available. Data spanning from 01/01/2020 to 10/12/2020 were provided by the energy supplier Scholt Energy [34]. Forecasts, such as wind speed and temperature, were also provided by Scholt Energy. Publicly available data, namely balancing market prices of The Netherlands and day-ahead market prices of The Netherlands and Belgium, are collected from the ENTSO-E Transparency Platform [35].

For the period under study, data for 8280 hourly contracts are available. The data is split into training (4148 contracts), validation (1238 contracts) and test sets (1760 contracts). Contracts not used in training, validation, or testing are excluded because of data quality issues. For example, numerous contracts, especially in January and October, are found to have a significant number of missing observations.

To continuously train and evaluate our agent, a rolling window method illustrated in Fig. 6 is employed. First, feature elimination is implemented to select the most important features and A3C is shortly trained to tune hyper-parameters. These are evaluated on the validation set. Defining the hyper-parameters and features, A3C is trained on combined training and validation set and evaluated on the test set.

Table 1
External states.

minutes to end of trading session
spread between p_t^b and p_t^a
spread between $(p_t^a + p_t^b)/2$ and its average forecast
spread between p_t^b and Dutch day-ahead market price
spread between p_t^b and h^b forecast
spread between p_t^b and p^{feed} forecast
spread between p_t^b and $1/6\sum_{d-3}^{d-1}\sum_{h-2}^{h} p^{take}$
spreads between p_t^b and its lags $[p_{t-8}^b : p_{t-1}^b]$
the best bid quantity q_t^b
the number of bid orders
the third (upper) quantile of cumulative bid quantities
the best ask price p_t^a
the first (lower) quantile of ask prices
the second quantile (median) of ask prices
the third (upper) quantile of ask prices
spread between p_t^a and $1/6\sum_{d-3}^{d-1}\sum_{h-2}^{h} p^{take}$
the best ask quantity q_t^a
total ask quantities
the number of ask orders

4.2. Data preprocessing

Order book and trade book data are processed by creating state variables indexed by important revision numbers. Processing each contract, we vertically stack all contracts by time separately for training, validation and test sets. We extract min–max scaling parameters from the training set for each variable and scale all the data.

4.3. External state selection

Feature selection is used to find the most important external states as mentioned in Section 3.5. Initially, collinearity between features is measured using the Pearson correlation. Correlated features with a correlation larger than 0.8 are removed. Subsequently, logistic regression is used to eliminate the least important features. The regularisation penalty λ is set to 0.001 and the mixing parameter between lasso and ridge regressions α is set to 0.9. Class balancing is employed to increase f1 validation scores of buy and sell classes.

Using weights of the logistic regression as feature importance scores, we eliminate 50% of the least important features in the first round and recursively eliminate 10% of the least important features in the following rounds. We stop the recursive feature elimination when a stopping criterion is reached; the f1 validation score decreases by a total of 0.1. Note that instead of performing recursive feature elimination with a fixed step size, a variable step size is used to reduce the training time. Using a variable step size ensures that fewer features are eliminated after each round. Features with a high importance are thus less likely to be removed. Features selected by our method are presented in Table 1.

4.4. A3C modelling

In our RL environment, actions and rewards are constructed according to Sections 3.3 and 3.4. States, consisting of external and internal states, are provided in Tables 1 and 2. In Table 2, q^{high}, PnL^{low} and PnL^{high} are set per contract to 40, −600 and 10000 respectively. These are given to scale agents' cash and volume portfolios, similarly to the min–max scaling applied for the external states. The categorical trade rule variable is added to guide agents using forecast decision thresholds. This variable is updated for every rolling window, based on the best validation set performance, resulting from one of the following forecasts: the average traded price; vwap; high price and low price; and p^{take} and p^{feed}. For instance, the buy category is assigned if p_t^a is lower than a low price forecast, sell is assigned if p_t^b is higher than a high price forecast, and hold is assigned otherwise.

Table 2
Internal states.

categorical trade rule based on forecasts
scaled total bought quantity $\sum_t q_t^a / q^{high}$
scaled total sold quantity $\sum_t q_t^b / q^{high}$
scaled position $(v_t + v^{max})/(2 \times v^{max})$
scaled cash $(PnL - PnL^{low})/(PnL^{high} - PnL^{low})$

DRL training is implemented with A3C per Section 3.6. We assign the probability of buy/sell to 0 if the volume position reaches $v^{max} = 3/v^{min} = -3$ or the total bought/sold quantity reaches to $q^{high} = 40$. The subtracted probability is added to another action using the categorical trade rule variable. For example, when $P(S) = 0.1$ and $v_t = v^{min}$, we simply assign $P(S) = 0$ and $P(H) = P(H) + 0.1$, if the categorical trade rule variable advises to hold. At the end of each episode, a callback function checks whether the total reward and PnL of the training set increased and saves improved models. We evaluate the last saved model.

As mentioned in Section 3.6, we implement hyper-parameter tuning. Specifically, we set trials on the number of agents $n \sim [4, 8]$, discount rate $\gamma_{start} \sim [0.001, 0.9]$, the number of time steps to update the global network $t^{max} \sim [300, 5000]$, learning rate $\beta \sim [0.00001, 0.1]$, optimiser $\sim \{\text{Adam}, \text{SGD}, \text{RMSProp}\}$, and the type of network $\sim \{\text{one-headed}, \text{two-headed}\}$, activation function for hidden layers $\sim \{\text{relu}, \text{tanh}\}$, the number of hidden layers $h \sim [1, 3]$, the number of neurons for the first, second and third hidden layers $n_1 \sim [4, 512]$, $n_2 \sim [4, 256]$ and $n_3 \sim [4, 128]$ respectively. A3C is trained until $e^{max} = 100$ and evaluated on the validation set by PnL for each trial. The process is performed by an automated Python tuning library, `optuna`, which prunes unsuccessful hyper-parameters and makes the tuning process more efficient [36]. The resulting hyper-parameters are $n = 8$, $\gamma_{start} = 0.29$, $t^{max} = 2906$, $\beta = 0.0003$, $h = 2$, $n_1 = 216$, $n_2 = 193$, the activation function of hidden layers is tanh, the optimiser is Adam, and the type of network is a two-headed shared network.

Other hyper-parameters are defined by user preferences. We set discount rate $\gamma_{end} = 0.9999$, stop loss threshold $SLT = -300$ (i.e. the end of contract reward $r^e < -3$), epsilon $\epsilon_{start} = 0.9$ and $\epsilon_{end} = 0.01$, maximum buy/sell quantity $q^{max} = 1$, maximum allowed position $[v^{min}, v^{max}] = [-3, 3]$, maximum allowed total traded quantity for both buy and sell sides $q^{high} = 40$, and the number of agents who clone behaviour $b = 5$. Note that the first agent clones (7), the second agent clones (8), and the next three agents clone (9) with different reward thresholds for their first episodes. Additionally, during training agents clone (9) for the last t^{max} states if the last state $s_t = s_T$ is reached by an early episode ending and is the same for the last four episodes.

Using the selected hyper-parameters, A3C is trained for each month using the rolling window method. If a policy manages to profit from most of the training contracts in an episode, we define this policy as an optimal policy. If an optimal policy is reached, we stop training as early as $e^{max} = 1000$. If not, training continuous until $e^{max} = 20000$. Although training times varied across monthly rolling windows, as a result of early stopping and differing data sizes, on a GeForce GTX 1080 we were able to train our proposed A3C method using the PyTorch library in roughly $1 - 2$ days for each monthly rolling window. The storage costs of training were between 1.7 and 3.2 megabytes. After training, A3C can execute a decision instantly. The storage cost of this is negligible.

4.5. Benchmark strategies

Similarly to A3C, explored benchmarks execute an existing order $q_t^a \le q^{max} = 1$ and $q_t^b \le q^{max} = 1$ for buy and sell actions at time step t. Their positions are constrained between $[v^{min}, v^{max}] = [-3, 3]$. And the total bought and sold quantities, $\sum q^a$ and $\sum q^b$, are limited to $q^{high} = 40$ MW. The first benchmark we program is VWAP using the volume weighted average price of trades until time step t, i.e. $p_t^{vwap} = \sum_{i=1}^t (\text{price}_i \times \text{volume}_i) / \sum_{i=1}^t (\text{volume}_i)$. Knowing the past traded prices, VWAP tries to maximise its revenue by following (10).

$$a_t = \begin{cases} B, \text{ if } p_t^a < p_t^{vwap} \\ S, \text{ if } p_t^b > p_t^{vwap} \\ H, \text{ otherwise} \end{cases} \tag{10}$$

The remaining benchmarks that are programmed, BENCHPLUS and BENCH, are expected to perform well because all the true values provided to these benchmarks are valuable and not known ex-ante. If these two benchmarks indeed result in success, a simple trading rule using forecasts of the provided values could be a promising trading strategy. If, on the other hand, these two benchmarks result in poorer performance than A3C, an intelligent agent who can consider various outcomes is needed. BENCHPLUS sets its trading rules knowing l^a, h^b, p^{take} and p^{feed} as given in (11).

$$a_t = \begin{cases} B, \text{ if } p_t^a < h^b \ \& \begin{cases} p_t^b > l^a \ \& \ p_t^a < p^{feed} \\ \text{or } p_t^b < l^a \end{cases} \\ S, \text{ if } p_t^b > l^a \ \& \begin{cases} p_t^a < h^b \ \& \ p_t^b > p^{take} \\ \text{or } p_t^a > h^b \end{cases} \\ H, \text{ otherwise} \end{cases} \tag{11}$$

BENCH sets its trading rules knowing l^a, h^b, and the averages of 30 future time steps of the best ask and bid prices. Its trading rule is given in (12).

$$a_t = \begin{cases} B, \text{ if } p_t^a < h^b \ \& \begin{cases} p_t^a < 1/30 \sum_{i=t+1}^{t+30} p_i^b \\ \text{or } p_t^b < l^a \end{cases} \\ S, \text{ if } p_t^b > l^a \ \& \begin{cases} p_t^b > 1/30 \sum_{i=t+1}^{t+30} p_i^a \\ \text{or } p_t^a > h^b \end{cases} \\ H, \text{ otherwise} \end{cases} \tag{12}$$

4.6. Evaluation

We evaluate our A3C trading algorithm on the test set using both pure profit and reward–risk metrics. For the profit metric, we calculate PnL_c made for each contract $c \in [c_1, C]$, where C is the total number of contracts in the test set. A policy is defined as being successful if more than half of the contracts return positive PnL_c and $\sum_{c=1}^C PnL_c$ is positive. For the reward–risk metrics, we calculate the profit to deviation ratio $PD = \sum_{c=1}^C PnL_c / \sigma$, where σ is the standard deviation of the PnL distribution. To assess the overall quality of actions, we also check the profit per trade (PT) ratio calculated as the cumulative PnL divided by the total traded quantity, $PT = \sum_{c=1}^C PnL_c / \sum_{c=1}^C \sum_{t=1}^T q_t$. Considering our pre-defined trading cost, a successful policy should have a $PT > 0.232$. The higher the PD and PT the better the trading strategy. All the above metrics together inform us about the trade-off between the profit and risk, i.e. the volatility of profits.

4.7. Results and discussion

Figs. 7 and 8 present the cumulative traded quantity and the cumulative PnL for the three evaluated benchmarks and our A3C trading algorithm across the test set. Evaluating the performances of VWAP and BENCH, relative to our A3C algorithm the two benchmarks place three times as many trades. Despite a high traded quantity, however, they generate lower revenues. Given high trade quantities but low revenues, one may expect trading costs, i.e. €0.232 per 1 MW, to be responsible for the two benchmarks' poor performances. An inability to manage balancing risk is, however, also responsible. VWAP and BENCH perform reasonably well across the intraday market. Their performance, however, is broadly negatively impacted when the requirements of re-balancing are factored-in. VWAP and BENCH fail to profit from arbitrage opportunities between the intraday and the balancing markets.

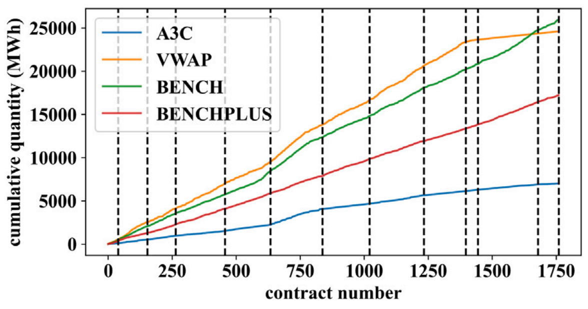

Fig. 7. $\sum q$ across test set contracts. The dashed black lines represent monthly rolling window splits.

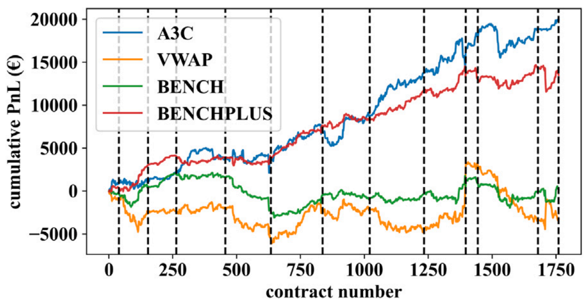

Fig. 8. $\sum PnL$ on the test set contracts. The dashed black lines represent monthly rolling window splits.

Table 3
Statistical Description of Traded Quantity (in MW).

	A3C	VWAP	BENCH	BENCHPLUS
$\sum q$	7017.20	24614.60	25943.20	17245.80
mean	3.99	13.99	14.74	9.80

Moreover, they fail to efficiently plan for necessary position closures. This restricts the final attainable PnL of the strategies.

Evaluating the performance of the third benchmark, BENCHPLUS places significantly fewer trades than VWAP or BENCH. Despite placing fewer trades, BENCHPLUS generates significantly higher profits. Knowledge of balancing market prices allows BENCHPLUS to outperform VWAP and BENCH. BENCHPLUS, however, cannot surpass A3C's $\sum PnL$. The granularity of time-steps may partially explain the lower $\sum PnL$. Across our case study, an agent is required to make thousands of decisions per contract (see Fig. 3). Implementing an efficient rule-based trading strategy is challenging and complex when an agent is required to make thousands of decisions. An intelligent agent who can autonomously learn to consider and plan for future outcomes is needed. A3C manages this, steadily increasing its revenue from 0 to €19927.22 and surpassing all trading rule-based benchmarks.

Expanding the analysis, Tables 3 and 4 present additional metrics for evaluating the performances of the benchmark and AC3 trading agents. Table 3 highlights that A3C on average trades 3.99 MW. This is marginally higher than the maximum position size (3 MW) that the agent is permitted to have open at any moment. The relatively low average traded quantity of the AC3 agent can be explained by our choice of hyperparameters. Our AC3 agent is set up to learn a risk-constrained trading strategy.

Analysing the percentage of positive PnLs, Table 4 shows that A3C and the benchmarks successfully return a positive $PnL > 0$ at least 50% of the test contracts. VWAP, with knowledge of only past prices, is found to be the worst-performing agent. It fails to beat trading rules with knowledge of future prices as would be expected.

Analysing risk-reward ratios, A3C has a high PD and PT indicating a favourable reward-to-risk ratio. The PD of BENCHPLUS is higher,

Table 4
Statistical description of PnL (in €).

	A3C	VWAP	BENCH	BENCHPLUS
$\sum PnL$	19927.22	−1631.73	2578.04	14862.57
mean	11.32	−0.92	1.46	8.44
std	139.54	152.26	96.50	92.91
$PnL > 0$	56.08%	50.23%	59.77%	64.21%
$PnL < 0$	38.64%	49.77%	40.22%	35.80%
PD	142.80	−10.72	26.72	159.96
PT	2.84	−0.07	0.10	0.86

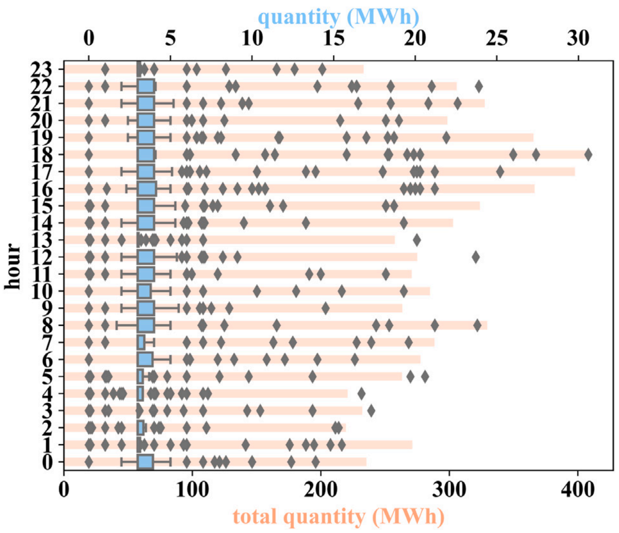

Fig. 9. Traded quantity distribution (blue) and total quantity (salmon) of A3C for each delivery hour on the test set.

however. This result is expected since BENCHPLUS uses true values in its decision-making process. In practice, were forecasts used instead of actual prices in BENCHPLUS, the risk (standard deviation of profits) would be higher and the PD would be lower due to the direct impacts of forecast errors. The PT of A3C is the highest out of all evaluated trading strategies. Based on its PnL, PD and PT performances, we consider AC3 to be the best trading agent.

A3C performance across delivery hours: Figs. 9 and 10 present the distribution of traded quantities and profits across delivery hours. Traded quantity distributions in Fig. 9 show that A3C on average trades around 4 MW for $h \in H$. AC3 trades as a maximum of roughly 30 MW, and a minimum of 0 MW. Considering the pre-defined maximum allowed total trading quantity of 40 MW, A3C thus successfully adjusts its desired trading quantity. It does this for each contract separately.

Total traded quantities in Fig. 9 show that A3C trades less during the night than during the day. While it only trades around 200 MW during the night, it trades as much as 400 MW during the day. A3C's ability to learn to exploit the increased liquidity available during day hours is impressive since we did not provide any rule regarding day and night trading.

As shown in Fig. 10, A3C manages to return a profit (total $PnL > 0$) for most of the hours. The total PnLs across night hours is modest but less volatile. The total PnLs across the day, on the other hand, is greater but more volatile. Figs. 9 and 10 are considered together, we can observe that a high number of trades do not necessarily result in a high total PnL. While the total traded quantity is the highest between hours 16 and 19, the total PnL is volatile without high returns.

To further analyse the relationship between the traded quantity and PnL, we plot a scatter plot of profits and traded quantity in Fig. 11. No significant correlation between profits and traded quantity is identified. The Pearson correlation between the PnL and traded

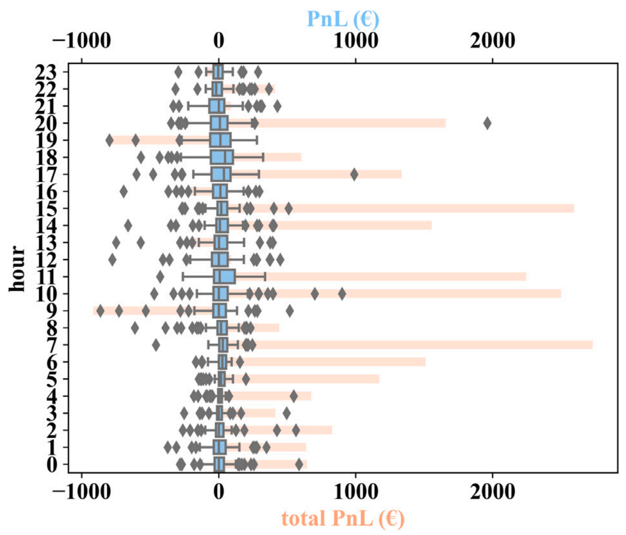

Fig. 10. PnL distribution (blue) and total PnL (salmon) of A3C for each delivery hour on the test set.

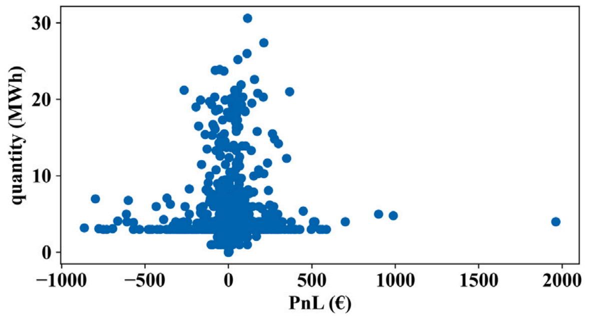

Fig. 11. Traded quantity of A3C is plotted against its PnL.

quantity is calculated to be 0.03. Note that this result is dependent on factors such as the maximum allowed position set. Even though for one contract the A3C agent is observed to trade as many as 30 MW, we constrain it to not leave more than +-3 MW to the balancing market. This is a crucial restriction since balancing market prices are highly volatile and can drastically change the PnL outcome. Keeping the maximum allowed position as low as 3 MW reduces the risk and, as a side effect, may reduce the correlation between the traded quantity and PnL.

5. Conclusion

Our study focuses on the statistical arbitrage trading strategy which aims to profit by trading continuously on the intraday market with the opportunity to close the remaining position on the balancing market. The objective of the study is to maximise profit while minimising any associated risk. To hedge against the risk, a constrained trading strategy is implemented with a maximum allowed position for both short and long trades, and a maximum total allowed trading quantity for total bought and sold quantities.

Our study demonstrates that asynchronous advantage actor–critic (A3C), a deep reinforcement learning (DRL) method, can be successfully utilised to develop an autonomous trading agent capable of exploiting arbitrage opportunities. In our study, states are selected from a large number of engineered features using a recursive feature elimination method. Exploration is enhanced by using decayed epsilon greedy exploration and behaviour cloning, i.e. goal-based exploration. A3C function approximators are designed with two-headed shared deep neural networks.

Our methodology is evaluated on a case study using the limit order book of the single intraday coupled market (SIDC) available for the Dutch market area. A3C is trained on 4148 hourly contracts and tested on 1760 contracts using a rolling window method. On test contracts, A3C trades a total of 7017.20 MW of electricity and generates profits of €19927.22 with an average profit of €2.84 per trade. The majority of contracts, 56.08%, return a profit.

We hope that our work spurs others to research DRL algorithms for statistical arbitrage trading. We recommend that researchers looking to extend our work consider evaluating actor–critic variants, such as advantage actor critic (A2C) and soft actor–critic, or, given the successes of our DRL environment, explore more sophisticated reward functions, novel exploration methods, and flexible volume and quantity constraints.

CRediT authorship contribution statement

Sumeyra Demir: Conception and design of study, Acquisition of data, Analysis and interpretation of data, Writing – original draft. **Bart Stappers:** Writing – review & editing. **Koen Kok:** Writing – review & editing. **Nikolaos G. Paterakis:** Writing – review & editing.

Acknowledgements

We thank Scholt Energy for their support and Walter van Alst for his insightful guidance.

References

[1] Saravia C. Speculative trading and market performance: the effect of arbitrageurs on efficiency and market power in the New york electricity market. 2003, Center for the Study of Energy Markets.
[2] Baltaoglu S, Tong L, Zhao Q. Algorithmic bidding for virtual trading in electricity markets. IEEE Trans Power Syst 2019;34(1):535–43. http://dx.doi.org/10.1109/TPWRS.2018.2862246.
[3] Li R, Svoboda AJ, Oren SS. Efficiency impact of convergence bidding in the california electricity market. J Regul Econ 2015;48(3):245–84.
[4] Borenstein S, Bushnell J, Knittel CR, Wolfram C. Inefficiencies and market power in financial arbitrage: a study of California's electricity markets. J Ind Econ 2008;56(2):347–78.
[5] Mather J, Bitar E, Poolla K. Virtual bidding: Equilibrium, learning, and the wisdom of crowds. IFAC-PapersOnLine 2017;50(1):225–32.
[6] Hogan WW. Virtual bidding and electricity market design. Electr J 2016;29(5):33–47.
[7] Hadsell L. The impact of virtual bidding on price volatility in New York's wholesale electricity market. Econom Lett 2007;95(1):66–72.
[8] AlAshery MK, Xiao D, Qiao W. Second-order stochastic dominance constraints for risk management of a wind power producer's optimal bidding strategy. IEEE Trans Sustain Energy 2019;11(3):1404–13.
[9] Qiu D, Ye Y, Papadaskalopoulos D, Strbac G. Scalable coordinated management of peer-to-peer energy trading: A multi-cluster deep reinforcement learning approach. Appl Energy 2021;292:116940. http://dx.doi.org/10.1016/j.apenergy.2021.116940.
[10] Dorokhova M, Martinson Y, Ballif C, Wyrsch N. Deep reinforcement learning control of electric vehicle charging in the presence of photovoltaic generation. Appl Energy 2021;301:117504. http://dx.doi.org/10.1016/j.apenergy.2021.117504.
[11] Schulman J, Moritz P, Levine S, Jordan MI, Abbeel P. High-dimensional continuous control using generalized advantage estimation. 2015, arXiv:1506.02438.
[12] Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, et al. Asynchronous methods for deep reinforcement learning. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd international conference on machine learning. Proceedings of machine learning research, vol. 48, New York, New York, USA: PMLR; 2016, p. 1928–37.
[13] Kang Q, Zhou H, Kang Y. An asynchronous advantage actor-critic reinforcement learning method for stock selection and portfolio management. In: Proceedings of the 2nd international conference on big data research. ICBDR 2018, New York, NY, USA: Association for Computing Machinery; 2018, p. 141–5. http://dx.doi.org/10.1145/3291801.3291831.
[14] Li Y, Zheng W, Zheng Z. Deep robust reinforcement learning for practical algorithmic trading. IEEE Access 2019;7:108014–22. http://dx.doi.org/10.1109/ACCESS.2019.2932789.

[15] Cao D, Hu W, Xu X, Dragičević T, Huang Q, Liu Z, et al. Bidding strategy for trading wind energy and purchasing reserve of wind power producer – a DRL based approach. Int J Electr Power Energy Syst 2020;117:105648. http://dx.doi.org/10.1016/j.ijepes.2019.105648.

[16] Xiao D, do Prado JC, Qiao W. Optimal joint demand and virtual bidding for a strategic retailer in the short-term electricity market. Electr Power Syst Res 2021;190:106855. http://dx.doi.org/10.1016/j.epsr.2020.106855.

[17] Xiao D, Qiao W, Qu L. Risk-constrained stochastic virtual bidding in two-settlement electricity markets. In: 2018 IEEE power energy society general meeting. 2018, p. 1–5. http://dx.doi.org/10.1109/PESGM.2018.8586115.

[18] Mehdipourpicha H, Wang S, Bo R. Developing robust bidding strategy for virtual bidders in day-ahead electricity markets. IEEE Open Access J Power Energy 2021;8:329–40. http://dx.doi.org/10.1109/OAJPE.2021.3105097.

[19] Tang W, Rajagopal R, Poolla K, Varaiya P. Model and data analysis of two-settlement electricity market with virtual bidding. In: 2016 IEEE 55th conference on decision and control. 2016, p. 6645–50. http://dx.doi.org/10.1109/CDC.2016.7799292.

[20] Li Y, Yu N, Wang W. Machine learning-driven virtual bidding with electricity market efficiency analysis. IEEE Trans Power Syst 2022;37(1):354–64. http://dx.doi.org/10.1109/TPWRS.2021.3096469.

[21] Pozzetti L, Cartlidge J. Trading electricity markets using neural networks. In: Proceedings of the 32nd European modeling & simulation symposium. 2020, p. 311–8. http://dx.doi.org/10.46354/i3∼m.2020.emss.045.

[22] Garnier E, Madlener R. Balancing forecast errors in continuous-trade intraday markets. Energy Syst 2015;6:361–88. http://dx.doi.org/10.1007/s12667-015-0143-y.

[23] Aïd R, Gruet P, Pham H. An optimal trading problem in intraday electricity markets. Math Financial Econ 2016;10:49–85. http://dx.doi.org/10.1007/s11579-015-0150-8.

[24] Čović N, Braeuer F, McKenna R, Pandžić H. Optimal PV and battery investment of market-participating industry facilities. IEEE Trans Power Syst 2021;36(4):3441–52. http://dx.doi.org/10.1109/TPWRS.2020.3047260.

[25] Bertrand G, Papavasiliou A. Adaptive trading in continuous intraday electricity markets for a storage unit. IEEE Trans Power Syst 2020;35(3):2339–50. http://dx.doi.org/10.1109/TPWRS.2019.2957246.

[26] Boukas I, Ernst D, Théate T, Bolland A, Huynen A, Buchwald M, et al. A deep reinforcement learning framework for continuous intraday market bidding. Mach Learn 2021.

[27] Brijs T, Geth F, De Jonghe C, Belmans R. Quantifying electricity storage arbitrage opportunities in short-term electricity markets in the CWE region. J Energy Storage 2019;25:100899. http://dx.doi.org/10.1016/j.est.2019.100899.

[28] Demir S, Kok K, Paterakis NG. Exploratory visual analytics for the European single intra-day coupled electricity market. In: 2020 international conference on smart energy systems and technologies. 2020, p. 1–6. http://dx.doi.org/10.1109/SEST48500.2020.9203043.

[29] Narajewski M, Ziel F. Ensemble forecasting for intraday electricity prices: Simulating trajectories. Appl Energy 2020;279:115801. http://dx.doi.org/10.1016/j.apenergy.2020.115801.

[30] Stappers B, Paterakis NG, Kok K, Gibescu M. A class-driven approach based on long short-term memory networks for electricity price scenario generation and reduction. IEEE Trans Power Syst 2020;35(4):3040–50. http://dx.doi.org/10.1109/TPWRS.2020.2965922.

[31] Sutton RS, Barto AG. Reinforcement learning: an introduction. 2nd ed.. The MIT Press; 2018.

[32] Glas S, Kiesel R, Kolkmann S, Kremer M, von Luckner NG, Ostmeier L, et al. Intraday renewable electricity trading: advanced modeling and numerical optimal control. J Math Industry 2020;10(3). http://dx.doi.org/10.1186/s13362-020-0071-x.

[33] Kath C, Ziel F. Optimal order execution in intraday markets: Minimizing costs in trade trajectories. 2020, Pre-Print, arXiv:2009.07892.

[34] Control SE. https://www.scholt.com. [Accessed 12 December 2020].

[35] ENTSOE Transparency Platform. https://transparency.entsoe.eu/. [Accessed 12 December 2020].

[36] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD international conference on knowledge discovery and data mining. 2019.