MOT: A Mixture of Actors Reinforcement Learning Method by Optimal Transport for Algorithmic Trading

Xi Cheng[1,2], Jinghao Zhang[1,2], Yunan Zeng[1,2], and Wenfang Xue[1,2]

[1] School of Artificial Intelligence, University of Chinese Academy of Sciences
[2] Institute of Automation, Chinese Academy of Sciences
{xi.cheng, jinghao.zhang, yunan.zeng}@cripac.ia.ac.cn,
wenfang.xue@ia.ac.cn

Abstract. Algorithmic trading refers to executing buy and sell orders for specific assets based on automatically identified trading opportunities. Strategies based on reinforcement learning (RL) have demonstrated remarkable capabilities in addressing algorithmic trading problems. However, the trading patterns differ among market conditions due to shifted distribution data. Ignoring multiple patterns in the data will undermine the performance of RL. In this paper, we propose MOT, which designs multiple actors with disentangled representation learning to model the different patterns of the market. Furthermore, we incorporate the Optimal Transport (OT) algorithm to allocate samples to the appropriate actor by introducing a regularization loss term. Additionally, we propose Pretrain Module to facilitate imitation learning by aligning the outputs of actors with expert strategy and better balance the exploration and exploitation of RL. Experimental results on real futures market data demonstrate that MOT exhibits excellent profit capabilities while balancing risks. Ablation studies validate the effectiveness of the components of MOT.

Keywords: Algorithmic trading · Reinforcement learning · Optimal transport.

1 Introduction

The goal of algorithmic trading is to maximize long-term profits while keeping risks within an acceptable range [21]. Compared to the traditional approach of relying on the expert judgment of trading timing, algorithmic trading is highly automated and efficient.

Traditional technical analysis methods include mean reversion [10], momentum investing [11], multi-factor models [4], etc. However, financial market data is non-stationary with a low signal-to-noise ratio. Expert-designed technical analysis methods can't generate profits under diverse market conditions. Deep learning methods excel at capturing intricate price patterns and enhance models' performance [28,29,15]. However, the process from supervised models' output to actual investment still requires the construction of strategies, which introduces expert knowledge and subjectivity. RL methods don't require carefully designed strategies by humans. They take market information as states and output trading decisions directly, which makes it easy to incorporate the unique financial constraints (e.g. transaction costs and slippage) into environments. RL has achieved SOTA in many quantitative investment tasks [16,19,30].

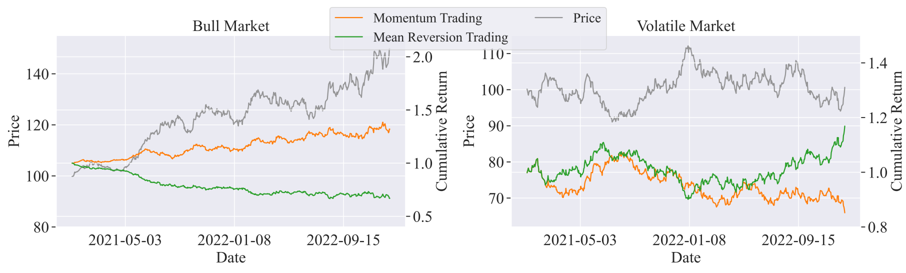


Fig. 1: Profit of strategies in different market conditions. A bull market is suitable for momentum trading, while a volatile market is suitable for mean reversion trading.

However, these methods rely on the assumption that financial data always follow the same distribution. Data patterns often switch in real scenarios. E.g. the most common way to classify market patterns is into two categories: stable (momentum) and volatile (reversal) markets, which require two categories of strategies [14]. These two phenomena are not independent but intertwined with each other. As shown in Figure 1, when bullish forces > bearish forces or are evenly balanced, the market is in a stable upward (bull) or a volatile state respectively. Momentum trading strategy models the momentum effect of stable market and mean reversion strategy models the reversal effect of volatile market. The same strategy can yield significantly different returns in different market conditions. Inspired by a mixture of experts [5], we propose MOT, which models multiple actors with disentangled representation learning and extracts various pattern information in RL. To allocate samples to agents appropriately, we introduce the Allocation Module with Optimal Transport (OT) regularization loss.

Previous research [16] has introduced imitation learning to RL, allowing agents to learn information from expert knowledge. However, in the early stages of imitation learning, the sampled action used in the training process is not from the agent's generation but is directly given by the expert which is stored in the buffer. As a result, the true output action of the agent differs significantly from the action stored in the buffer. To solve this problem, MOT introduces a pre-training method based on supervised learning to imitation learning. We expect the output generated by the agent to be closer to the expert strategy in imitation learning so the model can be initialized in a better stage.

The training process of MOT can be divided into three stages: first, the Pretrain Module uses supervised learning to train only the actor with expert strategy. Then we use the expert strategy to fill the buffer and train the RL model by imitation learning. After that, MOT uses multiple actors to model different market patterns and uses OT to solve the problem of pattern allocation. The contributions are summarized as follows:

1) MOT is the first that introduces OT algorithm to RL for mining various trading patterns. Allocation Module allocates different samples to appropriate actors.

2) MOT is also the first study that addresses the imitation learning gap between the actor's output and the buffer. MOT introduces a supervised Pretrain Module before imitation learning, which allows the real actor's output to be closer to the expert strategy.

Table 1: Changes in Position Based on Trading Signals

Po	**Action**	**Po´**	**Operation**	**Po**	**Action**	**Po´**	**Operation**
0	1	1	Take a long position	0	-1	-1	Take a short position
1	1	1	No operation	-1	-1	-1	No operation
-1	1	1	Close the position then go long	1	-1	-1	Close the position then go short

3) Experiments show MOT has great profitability in different market modes while balancing risks. Further studies confirm the effectiveness of three components of MOT.

2 Problem Formulation

The algorithmic trading problem can be represented as Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ represents the state space provided by the environment, $\mathcal{A}$ represents the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the probability function of the conditional state transitions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. The specific definition of the five-tuple for MDP is as follows:

The State space $\mathcal{S}$: The state $\mathbf{S}_t = [\mathbf{S}_t^m; \mathbf{S}_t^a] \in \mathcal{S}$. The account indicators $\mathbf{S}_t^a = [s_t^{a_1}, s_t^{a_2}, ...]$ describe the trader's positions, account cash balance, margin, returns, and other related information of the trader's account. $\mathbf{P}_t = [p_t^o, p_t^h, p_t^l, p_t^c, v_t^o, v_t^a]$ represents the Opening-High-Low-Closing (OHLC) prices, trading volume, and trading value. $\mathbf{Q}_t = [q_t^1, q_t^2, ..., q_t^i]$ are derived from $\mathbf{P}_t$ and technical analysis. The market indicators $\mathbf{S}_t^m = [\mathbf{P}_t; \mathbf{Q}_t]$ include the volume-price data $\mathbf{P}_t$ and the technical indicators $\mathbf{Q}_t$.

The Action space $\mathcal{A}$: The action $a_t \in \{-1, 1\}$ represents the trading signal output by the agent. -1 corresponds to short selling and 1 corresponds to a long position. We define the agent to trade in units of contracts. The actual execution of trades depends on the trading signal and the trader's existing positions. The specific changes in position and action are summarized in Table 1, where Po means position.

The Transition Function $\mathcal{P}$: We assume that the actions of individual traders do not affect the overall asset price in the market. This implies that the observation transition function of market indicators is independent of trading behavior, i.e. $\mathcal{P}(\mathbf{S}_{t+1}^m | \mathbf{S}_t) = \mathcal{P}(\mathbf{S}_{t+1}^m | \mathbf{S}_t, a_t)$. However, the observation transition function of account prices is influenced by trading behavior, i.e. $\mathcal{P}(\mathbf{S}_{t+1}^a | \mathbf{S}_t) \neq \mathcal{P}(\mathbf{S}_{t+1}^a | \mathbf{S}_t, a_t)$.

The Reward $\mathcal{R}$: We choose the closing price p_t^c to calculate profit r_t. To better simulate real market, we set transaction fee rate μ [3] and slippage σ [4]. The profit r_t is defined as $r_t = (p_t^c - p_{t-1}^c - 2\sigma) \cdot a_{t-1} - \mu \cdot p_t^c \cdot |\Delta po|$, where $\Delta po = Po' - Po$. When setting rewards, it is inappropriate to consider only the profit without taking into account the risk. The Sharpe ratio is the most widely used indicator for balancing risk and returns [24], defined as $SR = \frac{mean(r_t)}{std(r_t)}$. To measure the impact of the profit on SR each step, we adopt the Differential Sharpe Ratio (DSR) [17] as the reward. Considering that the adjacent data is more important than distant previous data in algorithmic trading, DSR

[3] Transaction costs are charged as a percentage of the contract.

[4] Slippage refers to the difference between the expected and the actual execution price.

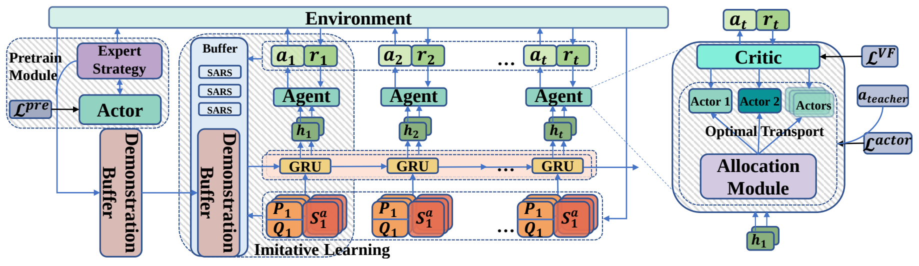


Fig. 2: The architecture of MOT. First, we pretrain the actor using the expert strategy and then proceed with imitation learning. We model different market patterns using multiple actors and allocate samples to the actors using the Allocation Module.

employs the smoothing technique of Exponential Moving Average (EMA). DSR_t is defined as:

$$DSR_t = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}}, \tag{1}$$

representing the impact of each new profit r_t on SR after applying EMA. A_t is the first moment and B_t is the second moment of profits r_t estimated by EMA. We utilize the DSR_t as the reward $\mathcal{R}$. If the account money is insufficient, the trading will be terminated in advance, and we simulate this by setting a margin threshold.

3 Methodology

The overview of MOT is present in Figure 2. First, to ensure alignment between the actions in Demonstration Buffer and the actual outputs of the actor, we introduce Pretrain Module. Second, we leverage imitation learning to initialize the RL algorithm. Third, we use multiple actors with disentangled representation learning and model various market conditions. Last, Allocation Module allocates samples to different actors by OT algorithm.

3.1 Imitation Learning

In RL-based algorithmic trading, the initial exploration phase is often inefficient and yields low profits. Imitation learning leverages expert knowledge and provides the actor with a favorable starting point. We employ PPO [23] as the backbone to address the MDP problem. To capture the temporal patterns of states $\mathbf{S}_t$, we utilize Gated Recurrent Units (GRU) [1] to obtain the hidden representation $h_t = GRU(h_{t-1}, \mathbf{S}_t)$ of states $\mathbf{S}_t$. h_t is then fed into the actor and critic networks as inputs.

The actor network aims to find the optimal policy π by maximizing the advantage function. The input is the environment state $\mathbf{S}_t$ and the output is the action a_t. To ensure sufficient exploration by the agent, we add noise ε to the output of the actor network. The actual executed action $a_t = \pi^\theta(h_t) + \varepsilon$, where ε represents the noise, π is the policy given by actor network with parameters θ. The trading experience trajectories (SARS:

state, action, reward, new state) are stored in the buffer $\mathcal{B}$. After sampling, we update the gradients of the actor network and the critic network using the data from $\mathcal{B}$.

The value function V, computed by the critic network with parameters ω, estimates the value of the sample under state $\mathbf{S}_t$. It is optimized through the loss function:

$$\mathcal{L}^{VF}(\omega) = \mathbb{E}\left[(V_\omega(\mathbf{S}_t) - V_t)^2\right], \tag{2}$$

where $V_t = \sum_{t'=t}^{T-1} \mathbb{E}[\gamma^{T-t'-1} DSR_{t'}(\mathbf{S}_{t'}, a_{t'})]$ represents the empirical value of the accumulated future rewards DSR and T is the total number of time steps.

Let $\delta_t^V = DSR_t + \gamma V(\mathbf{S}_{t+1}) - V(\mathbf{S}_t)$ represent the advantage value estimation. In our research, the advantage function is computed by generalized advantage estimator (GAE) [23]: $\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{k=t}^{T-1}(\gamma\lambda)^{k-t}\delta_k^V$, where γ is the discount factor, λ represents the trade-off between variance and bias.

PPO introduces a surrogate objective function to measure the similarity between the updated policy and the previous policy. The policy ratio formula is $\frac{\pi_\theta(a_t|\mathbf{S}_t)}{\pi_{\theta_{\text{old}}}(a_t|\mathbf{S}_t)}$. $\pi_{\theta_{\text{old}}}$ and π_θ represents the original and updated policy respectively. The objective function $\mathcal{L}^{CLIP}(\theta)$ for policy update is as Equation 3, ϵ is the clipping threshold.

We employ the commonly used Dual Thrust [13] as the expert strategy to provide demonstration actions. We store the demonstration trajectory SARS in Demonstration Buffer (DB) and train the agent using samples from DB. The training of the actor-critic network in imitation learning follows the same approach as the PPO algorithm, with the only difference being that the training data is from DB. Subsequently, the actor-critic network continues to train by PPO method, as shown in Equation 2 and Equation 3:

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}\left[\min\left(\frac{\pi_\theta(a_t|\mathbf{S}_t)}{\pi_{\theta_{old}}(a_t|\mathbf{S}_t)}\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]. \tag{3}$$

3.2 Pretrain Module

The Pretrain Module is used to align the actions in the buffer $\mathcal{B}$ with the outputs of the actor. As mentioned before, it can be observed that a_{expert} in DB is directly provided by Dual Thrust strategy rather than generated by the actor network π_θ. Therefore, when using the demonstration data for gradient descent of the network, there is a significant discrepancy between the distribution of the actor network's output action π_θ and the action a_{expert} [16]. This has a negative impact on the stability of the RL network.

To address this issue, we aim to align the output action $a_t = \pi_\theta$ of the actor network with the expert-provided action a_{expert} by training the actor network using supervised learning. The loss function is defined as Equation 4:

$$\mathcal{L}^{pre} = CrossEntropy(a_{expert}, \pi_\theta(h_t)) \tag{4}$$

Pretrain Module accelerates the actor's understanding of the task by mimicking expert strategies and enhances the actor's ability to effectively engage in the imitation learning process. Pretrain Module is positioned before imitation learning as Figure 2.

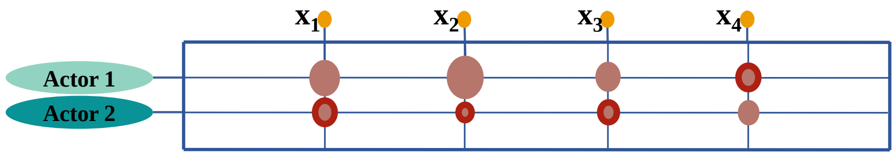


Fig. 3: OT refers to assigning x to the actor with the minimum L_{err}^{ij} while achieving a balanced allocation proportion, $\frac{x\ to\ Actor\ 1}{x\ to\ Actor\ 2} \approx \frac{w_1}{w_2}$. The pink circles represent L_{err}^{ij}.

3.3 Multiple Actors

We employ multiple actors to model strategies in different patterns. Futures data is derived from the trading activities of numerous participants and reflects different trading patterns [22]. Ignoring multiple patterns will reduce the performance of models [8]. All k actors of MOT are constructed in the same manner, as depicted in Figure 2 and Equation 3. For convenience, we illustrate how the model is trained with $k = 2$.

To integrate the outputs of the two actors, we use an Allocation Module to assign weights to them. Regarding the construction of the Allocation Module, we first consider what inputs should be provided to it. The historical sequence of futures states $\mathbf{S}_t$ plays a significant role in determining the current market patterns. Additionally, the historical decision errors of different actors represent their decision-making performance and also influence the current sample allocation. We use GRU to extract latent feature representations from $\mathbf{S}_t^i$, denoted as $\hat{h}_t^i = GRU(\hat{h}_{t-1}^i, \mathbf{S}_t^i)$, where i means i-th sample. As the calculation of sample decision errors, we provide posterior teacher actions on the training set. The teacher action $a_{teacher} = 1$ when the futures close price p_t^c increases in the next time step and -1 otherwise. Let a^{i1} and a^{i2} represent the action output by actor 1 and actor 2. The sample decision error $\mathbf{e}_t^i$ is then computed as $[a_{teacher\ t}^i - a_t^{i1}, a_{teacher\ t}^i - a_t^{i2}]$. To avoid introducing future information, we utilize the previous error $\mathbf{e}_{t-1}^i$. Subsequently, we concatenate $\hat{h}_t^i$ and embedding of error sequence $\mathbf{d}_{t-1}^i = GRU(\mathbf{d}_{t-2}^i, \mathbf{e}_{t-1}^i)$ and feed them into a fully connected layer to predict the allocation results, denoted as $\mathbf{b}_t^i = FC(\hat{h}_t^i, \mathbf{d}_{t-1}^i)$. In different patterns Allocation Module should have different attention for the two actors in Equation 5, where q_t^i represents the allocation weights, and a_t^i represents the final action. To ensure the discrete differentiability of the Allocation Module, we utilize the gumbel-softmax method [9] to compute Equation 5. It is worth noting that the allocation of samples is not binary, but rather a soft allocation ranging $0 < \mathbf{q}_t^i < 1$.

$$\mathbf{q}_t^i = softmax(\mathbf{b}_t^i),\ a_t^i = \mathbf{q}_t^{i\,T}[a_t^{i1}, a_t^{i2}], \tag{5}$$

However, if the actors want to learn different patterns, the representations should be as dissimilar as possible. Inspired by disentangled representation learning, we take the inputs x of the actors' last layers as the representations and design a disentangled loss to enable the agent to learn different patterns, $\mathcal{L}^{dis} = \sum_{i=1}^{N} x_{i1} \cdot x_{i2}$.

Algorithm 1 Training process of MOT

```
1: Initialize actor network parameters $\theta_0$, critic network parameters $\omega_0$ and epochs $K$
2: Obtain the expert strategy
3: Pretrain the actor by $\mathcal{L}^{pre}$ in Equation 4
4: Add the expert strategy to DB and train by imitation learning, get the dual policies $\pi_{\theta_j}(a|\mathbf{S})$, $j = 1, 2$
5: for $k = 0, 1, 2, ...$ do
6:     Collect the trajectory $\tau_t = (\mathbf{S}_t, a_t, DSR_t, \mathbf{S}_{t+1})_{t=0}^{T-1}$ by allocating the policy in Equation 5
7:     Compute advantages $\hat{A}_t$ by current value $V_{\omega_t}(\mathbf{S}_t)$
8:     Compute the policy ratio $\frac{\pi_{\theta_t}(a_j|\mathbf{S}_t)}{\pi_{\theta_{t-1}}(a_j|\mathbf{S}_t)}$
9:     Compute the loss $\mathcal{L}^{OT}$ and $\mathcal{L}^{dis}$ in Equation 7
10:    Update the policy network by maximizing the clipped objective using $\mathcal{L}^{actor}(\theta)$ in Equation 7 (both for actor 1 and actor 2)
11:    Update the critic network by minimizing loss $\mathcal{L}^{VF}(\omega)$ in Equation 2
12: end for
```

3.4 Optimal Transport Regularization

However, the model lacks a mechanism to ensure the effective allocation of samples to actors. Sometimes, the majority of samples are assigned to one actor. We incorporate OT techniques to ensure that the Allocation Module assigns more appropriate samples to each actor, thereby capturing diverse patterns more accurately.

We need to consider two main requirements. Firstly, the Allocation Module should allocate the samples to the actor with the smallest decision error. In other words, if $|a^i_{teacher\ t} - a^{i1}_t| > |a^i_{teacher\ t} - a^{i2}_t|$, we tend to assign the sample to actor 2. Secondly, the allocation of samples to the actors should be proportional to their respective patterns.

Below, we formally define the allocation problem. Assume we utilize N samples in each epoch of PPO's gradient descent process. Based on the error vector, we can construct an error matrix denoted as $L_{err} \in [N \times 2]$. Each element L^{ij}_{err} in it represents the decision error of the i-th sample on the j-th actor, given by $L^{ij}_{err} = a^i_{teacher} - a^{ij}$. Corresponding to L_{err} is the allocation matrix $M \in [N \times 2]$, where each element $M^{ij} \in \{0, 1\}$. The value of 1 in the allocation matrix M indicates that Allocation Module assigns the i-th sample to the j-th actor, while the value of 0 indicates no allocation.

The OT method is particularly suitable for solving allocation problem. OT involves determining an optimal allocation of resources from one location to another while minimizing overall cost or distance. It is also commonly employed to measure the difference between two probability distributions. Our research aims to find the optimal allocation scheme that minimizes L_{err}. The specific formulation of the problem is as follows,

$$\begin{aligned} &\min_{M} (L \cdot M) \\ &s.t. \frac{\sum_{i=1}^{N} M^{i1}}{N} = w_1,\ \frac{\sum_{i=1}^{N} M^{i2}}{N} = w_2,\ M^{i1} + M^{i2} = 1, \forall i = 1, 2, \ldots, N, \end{aligned} \tag{6}$$

where w_1 and w_2 represent the proportions corresponding to different modes (assumed to be $\frac{1}{2}$). We employ the Sinkhorn method to solve the OT problem [2]. Figure 3 provides a visual explanation of the problem we aim to address.

To align the distribution of the output $\mathbf{q}^i$ from the allocation module with M^i of the OT problem, we incorporate a cross-entropy loss term. Considering Allocation Module as part of actors, Equation 3 can be expanded to Equation 7, λ_O is the hyperparameter. The third term is L^{OT}. The pseudocode for the MOT is shown in Algorithm 1.

$$\mathcal{L}^{actor}(\theta) = \mathcal{L}^{CLIP}(\theta) + \mathcal{L}^{dis} + \lambda_O \sum_{k=1}^{2} M_t^{ik} log(\mathbf{q}_t^{ik}). \tag{7}$$

4 Experiments

4.1 Dataset

We utilize the IF stock index futures dataset whose underlying asset is the CSI 300 Index. The dataset provides minute-level trading data of contracts. Each minute bar includes OHLC, trading volume, etc. The total trading duration in a day is 240 minutes. We collected it from ricequant.com[5], and divided the data into a training set from 2015-12-31 to 2018-05-08 and a test set from 2018-05-09 to 2019-05-09.

4.2 Baselines, Evaluation Metrics and Hyperparameters

Baselines: Long Position Hold (buy futures and hold), Short Position Hold (borrow contracts and hold), Dual Thrust [13] (a technical analysis trading strategy commonly used for intraday trading), GRU [1] (a variant of RNNs[6]), PPO [23] (a RL method that improves stability by preventing large policy changes[7]), iRDPG [16] (SOTA: an off-policy algorithm that incorporates expert strategy and behavior cloning).

Evaluation Metrics: We will measure the model's performance by Accumulated Rate of Return (ARR, the overall profitability), Volatility (VO, measures by standard deviation of profit r), Annualized Sharpe Ratio (ASR, annualized version of SR), Maximum Drawdown (MDD, the maximum decline of an asset's value from its peak to the lowest over a period), Calmar Ratio (CR=$\frac{ARR}{MDD}$, risk-adjusted ARR based on MDD) and Sortino Ratio (SoR=$\frac{mean(r)}{std(min(r,0))}$, excess return per unit of downside risk).

Hyperparameters: We set transaction fee rate $\mu = 2.3 \times 10^5$ and slippage $\sigma = 0.2$. Insufficient account assets may trigger a forced liquidation. We set the margin threshold as 70% and initial capital $C = 50000\ CNY$. We repeated 6 experiments for each model.

[5] A well-known Chinese quantitative trading platform, https://www.ricequant.com/.

[6] We chose it as a baseline because we employed the GRU method in the Pretrain Module before imitation learning. The results of GRU demonstrate the performance of the Pretrain Module.

[7] We enhance PPO using imitation learning mentioned in Methodology Section.

Table 2: Experimental Results (↑ indicates the higher the better, ↓ indicates the opposite)

Methods	**ARR (↑)**	**VO (↓)**	**ASR (↑)**	**MDD (↓)**	**CR (↑)**	**SoR (↑)**
Long Hold	−2.598	0.261	−0.638	113.121	−0.001	−0.080
Short Hold	3.163	0.259	0.782	0.894	0.041	0.093
Dual Thrust	10.130	0.253	2.628	0.033	3.962	0.365
GRU	11.342(1.12)	0.242(0.00)	3.004(0.31)	0.016(0.02)	4.280(0.23)	0.399(0.05)
iRDPG	14.453(0.98)	0.254(0.01)	3.955(0.18)	0.023(0.03)	5.881(3.21)	0.537(0.03)
PPO	12.245(0.23)	0.243(0.00)	3.223(0.05)	0.022(0.02)	4.281(0.23)	0.436(0.01)
MOT-ND	15.322(1.25)	0.246(0.01)	4.252(0.24)	**0.005(0.01)**	**7.277(3.51)**	0.587(0.07)
MOT-NO	17.236(1.05)	0.248(0.01)	4.447(0.18)	0.026(0.01)	5.558(0.75)	0.529(0.08)
MOT	**20.379(0.85)**	**0.228(0.00)**	**5.395(0.26)**	0.011(0.02)	6.582(0.66)	**0.605(0.05)**

4.3 Experimental Results

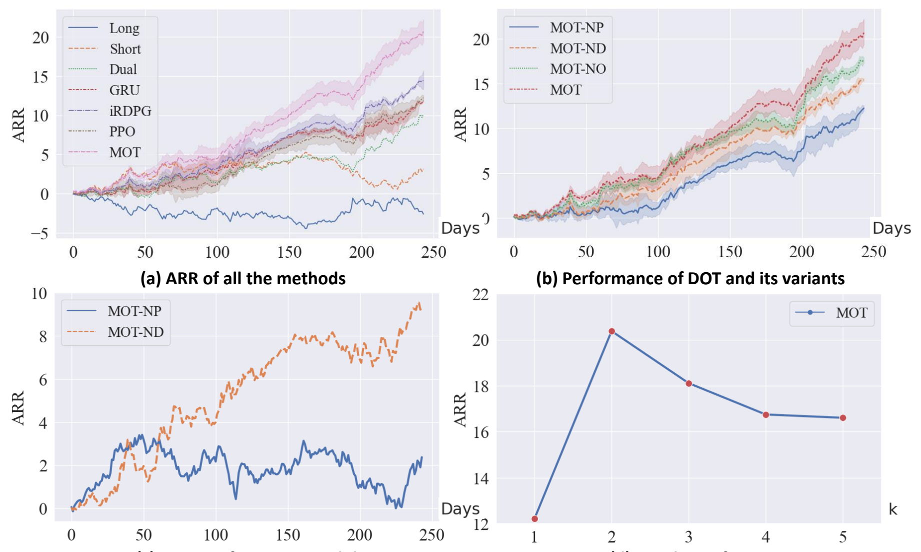
Table 2 provides a summary of the results. Figure 4 (a) depicts ARR of all the methods. From Table 2, MOT outperforms other methods in terms of profit and risk-reward balance. ARR is the most crucial indicator, and our model achieves the highest ARR. The ARR of PPO is about 1.0 higher than that of GRU, indicating that PPO exhibits greater robustness. The ASR, CR, and SoR are composite metrics that consider both risk and return. Deep learning methods (last 6 rows in Table 2) outperform the technical indicator models (first 3 rows in Table 2) in these three metrics, which suggests the former better represents complex states under high-noise conditions. MOT performs second in terms of MDD, indicating that MOT only requires a short time period to recover from losses. RL models outperform time-series models, as the latter primarily focuses on predicting price trends without considering the high costs caused by incorrect predictions. Since greater risk leads to greater returns, profits are higher when there are significant price fluctuations. So the correlation among most methods is very high.

4.4 Ablation Study

We conducted ablation experiments to show the effectiveness of its three components. The experimental results and the trend of ARR are depicted in Table 2 and Figure 4 (b).

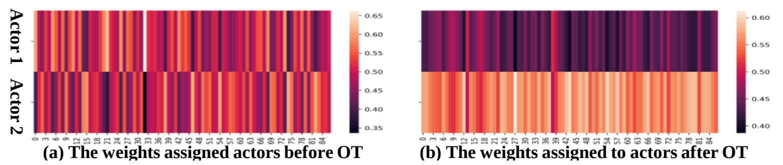
Overall performance. MOT-NP applies imitation learning based on PPO without Pretrain Module. MOT-ND is obtained by removing multiple actors from the final model, while MOT-NO eliminates the process of OT. From Figure 4 (b), we observe that ARR curve of MOT remains higher than other variants in most periods. Table 2 shows that MOT performs best in terms of ARR, VO, ASR, and SOR. Among the three modules, OT method contributes the most to the improvement of model performance, followed by the Pretrain Module. MOT-ND excels in MDD metric, indicating that the model without multiple actors' design tends to generate more conservative strategies. While a conservative trading strategy often misses the optimal investment opportunities. Since the calculation of CR relies on MDD, MOT-ND also exhibits higher CR.

Effectiveness of Pretrain Module. The influence of the expert strategy in DB diminishes over time and the benefit of imitation learning is mainly observed in the early



(a) ARR of all the methods
(b) Performance of DOT and its variants
(c) Impact of Pretrain Module
(d) Numbers of actors

Fig. 4: Performance of different models in terms of ARR



(a) The weights assigned actors before OT
(b) The weights assigned to actors after OT

Fig. 5: Effectiveness of OT modeling

stages. For the ablation experiment, we selected the agent trained for 100 epochs after imitation learning. Figure 4 (c) illustrates the impact of Pretrain Module on imitation learning and the yellow curve is the model with Pretrain Module. It can be observed that MOT-ND demonstrates a steady increase accompanied by minor fluctuations in profit. In contrast, MOT-NP experiences some declines and doesn't learn well. This indicates that Pretrain Module contributes to the improvement of imitation learning.

Effectiveness of multiple actors and OT modeling. Figure 5 demonstrates the variation in weights assigned to two actors before and after OT modeling. In a relatively volatile period, the model assigns weights more randomly without OT while assigns higher weights to actor 2 with OT. Notably, the introduction of OT leads to higher returns and enhances the ability to capture complex patterns. Figure 4 (d) illustrates the impact of actors' number to MOT. MOT achieves the best profitability when $k = 2$ while achieves the worst when $k = 1$. This indicates that only one actor is insufficient to capture all patterns, while an excessive number of actors may lead to redundancy. In our model, the optimal number of actors is 2.

5 Related Work

Investment strategies based on expert knowledge. The early method used expert knowledge to construct heuristic rules [10,20], which can be divided into two categories: fundamental analysis and technical analysis. Fundamental analysis captures diverse factors such as industry trends, company financial statements, and public opinion. This method is more commonly used by long-term investors to find undervalued assets. Popular technical indicators include Relative Strength Index [27], Average Direction Index [6], On-Balance Volume [26] , etc. Commonly used investment strategies include momentum trading [7] and mean reversion strategy [20]. However, interrelated technical indicators are correlated with each other, and building them directly from the market introduces too much market noise. Typically, rules constructed based on expert knowledge can only capture trading opportunities under specific market conditions [3].
Investment strategies based on RL. In contrast to supervised learning, which still requires expert knowledge to construct strategies, RL can optimize strategies in an end-to-end form. Moody et al. [18] made the first attempt to apply recurrent RL (RRL) algorithm to algorithmic trading. However, traditional RL methods are not well-suited for environments with large state spaces, making it challenging to select market features. Deep RL methods have partially addressed this problem. Si et al. [25] argue that strategies need to consider multiple factors and combine multi-objective optimization with deep RL to address this issue. Oliveira et al. [19] adopts SARSA, which maps states and actions to specific cells in a table to learn the value function. Since insufficient financial data causes overfitting, Jeong et al. [12] divided stocks into groups based on their correlations and introduced transfer learning into the Deep Q-Network (DQN). To shorten the inefficient random exploration phase, iRDPG [16] incorporates technical analysis through imitation learning. Yuan et al. [30] argue that daily frequency data cannot meet the high demands of RL and instead use minute frequency data. And PPO algorithm achieves more stable returns compared to DQN and SAC algorithms.

6 Conclusion

In this paper, we propose MOT, an RL-based model for algorithmic trading problems. Specifically, we model the algorithmic trading problem as MDP and leverage imitation learning to enable the agent to learn from expert knowledge. To better initialize MOT, we introduce the Pretrain Module prior to the imitation learning phase. Considering that futures prices result from different patterns, we employ multiple actors with disentangled representation learning to model the patterns. We design the Allocation Module to integrate the outputs of multiple actors and incorporate OT techniques to guide the learning of the Allocation Module. Experimental results demonstrate that our model achieves superior profitability while controlling the risk, showcasing its robustness in financial markets with complex data patterns. Further ablation studies confirm the effectiveness of the three components of MOT.

Acknowledgements This work was supported by the National Natural Science Foundation of China (No. 72374201).

References

1. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint:1412.3555 (2014)
2. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. NIPS **26** (2013)
3. Deng, Y., Bao, F., Kong, Y., Ren, Z., Dai, Q.: Deep direct reinforcement learning for financial signal representation and trading. IEEE Trans Neural Netw Learn Syst **28**(3), 653–664 (2016)
4. Fama, E.F., French, K.R.: Multifactor explanations of asset pricing anomalies. The journal of finance **51**(1), 55–84 (1996)
5. Fedus, W., Zoph, B., Shazeer, N.: Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. JMLR **23**(1), 5232–5270 (2022)
6. Gurrib, I., et al.: Performance of the average directional index as a market timing tool for the most actively traded usd based currency pairs. Banks and Bank Systems **13**(3), 58–70 (2018)
7. Hong, H., Stein, J.C.: A unified theory of underreaction, momentum trading, and overreaction in asset markets. The Journal of finance **54**(6), 2143–2184 (1999)
8. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for nlp. In: ICML. pp. 2790–2799. PMLR (2019)
9. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint:1611.01144 (2016)
10. Jegadeesh, N., Titman, S.: Returns to buying winners and selling losers: Implications for stock market efficiency. The Journal of finance **48**(1), 65–91 (1993)
11. Jegadeesh, N., Titman, S.: Cross-sectional and time-series determinants of momentum returns. The Review of Financial Studies **15**(1), 143–157 (2002)
12. Jeong, G., Kim, H.Y.: Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. Expert Systems with Applications **117**, 125–138 (2019)
13. Kim, H.j., Shin, K.s.: A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets. Applied Soft Computing **7**(2), 569–576 (2007)
14. Li, Z., Tam, V.: A machine learning view on momentum and reversal trading. Algorithms **11**(11), 170 (2018)
15. Lin, H., Zhou, D., Liu, W., Bian, J.: Learning multiple stock trading patterns with temporal routing adaptor and optimal transport. In: Proceedings of the 27th ACM SIGKDD. pp. 1017–1026 (2021)
16. Liu, Y., Liu, Q., Zhao, H., Pan, Z., Liu, C.: Adaptive quantitative trading: An imitative deep reinforcement learning approach. In: Proceedings of the AAAI conference on artificial intelligence. vol. 34, pp. 2128–2135 (2020)
17. Moody, J., Saffell, M.: Reinforcement learning for trading. NIPS **11** (1998)
18. Moody, J., Wu, L.: Optimization of trading systems and portfolios. In: Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr). pp. 300–307. IEEE (1997)
19. de Oliveira, R.A., Ramos, H.S., Dalip, D.H., Pereira, A.C.M.: A tabular sarsa-based stock market agent. In: Proceedings of the First ACM International Conference on AI in Finance. pp. 1–8 (2020)
20. Poterba, J.M., Summers, L.H.: Mean reversion in stock prices: Evidence and implications. Journal of financial economics **22**(1), 27–59 (1988)
21. Pricope, T.V.: Deep reinforcement learning in quantitative algorithmic trading: A review. arXiv preprint arXiv:2106.00123 (2021)

22. Ritter, J.R.: Behavioral finance. Pacific-Basin finance journal **11**(4), 429–437 (2003)
23. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint:1707.06347 (2017)
24. Sharpe, W.F.: Mutual fund performance. The Journal of business **39**(1), 119–138 (1966)
25. Si, W., Li, J., Ding, P., Rao, R.: A multi-objective deep reinforcement learning approach for stock index future's intraday trading. In: 2017 10th ISCID. vol. 2, pp. 431–436. IEEE (2017)
26. Tsang, W.W.H., Chong, T.T.L., et al.: Profitability of the on-balance volume indicator. Economics Bulletin **29**(3), 2424–2431 (2009)
27. Wilder, J.W.: New concepts in technical trading systems. Trend Research (1978)
28. Xu, W., Liu, W., Wang, L., Xia, Y., Bian, J., Yin, J., Liu, T.Y.: Hist: A graph-based framework for stock trend forecasting via mining concept-oriented shared information. arXiv preprint arXiv:2110.13716 (2021)
29. Xu, W., Liu, W., Xu, C., Bian, J., Yin, J., Liu, T.Y.: Rest: Relational event-driven stock trend forecasting. In: Proceedings of the Web Conference 2021. pp. 1–10 (2021)
30. Yuan, Y., Wen, W., Yang, J.: Using data augmentation based reinforcement learning for daily stock trading. Electronics **9**(9), 1384 (2020)