MARL Algortihms for Algorithmic Trading

Mert Cemri - Jingxing Wang
CS-285 Final Project Report

Extended abstract

Algorithmic trading strategies which utilize machine learning strategies have been increasingly popular recently, as they perform more favorably than traditional trading strategies in the dynamic and complex financial market environments. In our project, financial trading is formulated as a cooperative multi-agent RL (MARL) problem where multiple agents iteratively and synchronously interact with each other and the environment (stock market) and try to find an optimal (trading) policy. We worked with the GBP/USD currency data, which can be classified as a two-sided single asset stock market.

First, we implemented a suitable gym environment for the currency market (by building on an existing work for RL in finance for single agents), doing substantial changes in that environment to allow for multi-agent dynamics in the environment and fixing many bugs in the original environment. Then, we implemented the baseline algorithms to see how single-agent algorithms such as DDPG, A2C and PPO perform on this dataset. After some sanity checks and seeing that these agents meaningfully interact with the environment, we implemented the baseline multi-agent algorithms that are proposed in the literature to our environment, such as MAPPO, MADDPG and MAA2C. Note that it was not straightforward to come up with implementations of these algorithms to our environment, as algorithms like MAPPO were traditionally applied for games such as Starcraft. After establishing the baselines, we proposed a new algorithm with three crucial novelties that extend the literature, and then compared our proposed algorithm with the baseline methods at the end. Now, let us briefly discuss the new methods we proposed next.

The idea of the multi-agent RL algorithm we used in this project uses a hierarchical scheme, with different agents having different responsibilities (here, responsibility means the frequency of trading they focus on). In particular, let us consider the case we have 5 agents. The first agent does the highest frequency of trading and trades every day (so has period 1, P=1). The second agent has period 2 (P=2), and it trades once in every two days. The third agent has P=4, the fourth has P=8, and the fifth has P=16 (it trades once in every 16 days). Moreover, each agent sees the previous actions and rewards of the other agents with higher frequency. For instance, the third agent with P=4 sees the actions and rewards of P=1 and P=2 agents in the last 4 days. The theory behind such a structure rests on the Fractal Market Hypothesis (FMH) and Elliot Wave Theory which suggest that the major waves in stock prices are heavily influenced by the smaller waves; and therefore a careful aggregation of the information in high frequency trading can be very helpful for lower frequency trading. After establishing such a hierarchical structure, we propose 3 novel methods that significantly increase the performance of this model. First, we propose a clever method to choose which agent in this hierarchy should trade during inference time. Particularly, we come up with a learning-based strategy (using K-Means) to determine the empirical volatility of the stock market, and then if the volatility of the market is high, we use higher frequency agents to trade, and if the volatility of the market is low, we use the lower frequency agents to trade. As a second novelty, we ask the question, is there a more efficient way to transfer the knowledge of high-frequency trading to lower frequency agents. To that end, we initialize the neural network weights of the lower frequency agents with the trained weights of thehigher frequency agents. In addition, we show that we can boost the performance of our hierarchical MARL algorithm even further by applying a data-augmentation trick for the lower-frequency agents. In our experiments section, we show that all these three ideas we proposed are very beneficial for the algorithms return on the currency data. Finally, let us remind our two research hypotheses we had proposed. The first one was the question of scalability in MARL (does the overall performance increase as the number of agents increase). The second one was the suitability of a hierarchical multi-agent PPO algorithm for currency data (as the existing literature used DQN an DDPG algorithms for mulit-agent hierarhcichal trading agents). In our experiments, we showed using both our proposed hierarchical algorithm and the classical MAPPO algorithm that scalability hypothesis is false, and as the agent number increases, the overall performance does not necessarily increase (at some point noise becomes too high). Also, we show that the second hypothesis is true, as we use PPO algorithm for the individual agents in our hiearchichal model. Note that this model is very different from the MAPPO algorithm in the literature (and much more suitable to financial RL as we show).[1]

[1]The code and the trained models can be found here: `https://bit.ly/cs285_final_project`

1 Introduction, Related Work, and Our Contributions

In the landscape of algorithmic trading, a hierarchical multi-agent reinforcement learning (MARL) system is an effective use the collective intelligence of multiple agents (like a group of traders), we can formulate a multi-agent RL (MARL) framework where different agents are experts on different aspects of trading, but by interacting with each other they can learn from each other and collaboratively achieve financial returns which are superior to single-agent frameworks. In particular, letting some agents do high frequency trading (HFT) and some agents do low frequency trading (LFT) and make them communicate their findings with each other fosters a symbiotic relationship where insights from rapid HFT executions inform the strategic decisions of slower-paced LFT agents.

Fractal market hypothesis [1] suggests that the behaviour of stock prices in financial markets has a fractal property, exhibiting a similar structure in different time intervals. On top of the fractal market hypothesis, Elliot wave theory [2] proposes that the prices exhibits repetitive behaviour at different frequency intervals. Essentially, this theory suggests that the markets are heavily influenced and act according to the collective psychology of traders in recurring trading cycles. Hence, a collection of high frequency subwaves form a corresponding major wave at a lower frequency setting. Moreover, this major wave itself can be grouped with other waves at its own frequency to form more major waves in even lower frequency (and higher period) timeframes. One noteworthy interpretation of Elliot wave theory and the fractal market hypothesis is that whether it is caused by the interactions of traders (agents) with different investment tasks and the aggregation of information among these collective traders. Indeed, as [3] suggests, in real life each trader is usually an expert of a specific timeframe type of trading, some focus on short-term profits with more risks while some others focus on long-term investments with less risk. In that sense, considering the insights from the fractal market hypothesis and the Elliot wave theory, it might be beneficial to model a group of agents where each agent has a specific task (focus on a different frequency of trading) but such that they aggregate the information of each other. In addition, the proposed flow of information in the literature and also the one employed in this project is from high frequency to low frequency.

Regarding the previous literature, [3] described a multi-agent deep reinforcement learning framework that performs currency trading. The state space involved in the MDP formulation refers to the observation of the market at time t, which is the open, high, low, close, and volume of the currencies ($s_t = (OHLCV)_t$). The actions are: long, wait, short, $a_t = \{-1, 0, 1\}$. The reward is the correctness of the action scaled by the wealth quantity. The experiment is done with three agents, each focusing on different trading timeframes: 5, 15, 60 minutes. As the MARL model, an extension of Deep Q-Networks (DQNs) to multi-agent framework is developed. The authors assumed immediate execution by the agents, zero transaction fee, full availability of order size, no leverage, and zero market impact. The work [4] conducts a similar experiment on stocks using a Multi-Agent DDPG (MADDPG)-like model, where agents are trained to have different trading preferences (risk mitigation or long-term return). Under similar assumptions, the author experimented the agents' performances on different stocks with in the span of 500 days. In our work, our test dataset has 1000 days, and we develop our algorithms with less assumptions (such as we allow nonzero transaction costs).

Particularly, our contributions to the existing framework can be classified as following:

- We formulated a hierarchical multi-agent PPO algorithm suitable for algorithmic trading (that is different from MAPPO [5] algorithm), and showed its effectiveness over the methods proposed in the literature (effectiveness of hierarchical multi-agent PPO was our first research hypothesis). This is explained in Section 2.1. Also, as a baseline method, we implemented the MAPPO (as well as MAA2C and MADDPG) algorithm for financial trading environment. As MAPPO algorithms are usually used for game settings such as Starcraft, this adaptation took more time than expected, however this was necessary as it is a good baseline algorithm to compare with our proposed methods.

- We investigated the issue of scalability as we analyzed how the performance of the MARL algorithm vary with an increasing number of agents in the group (this was our other research hypothesis). This is demonstrated in Section 4.1.

- We proposed a new algorithm to adaptively determine which agent to trade once we have a group of hierarchical agents focusing on different timeframes based on volatility of the stock prices. This is explained Section 2.2 and demonstrated in Section 4.3.

- We proposed a novel method to increase the performance of hierarchical multi-agent setting for the trading envrinments where we warm-start low frequency agents with the immediate high frequency agents to impart their knowledge in a more effective way. This is explained in Section 2.3 and demonstrated in Section 4.2.

- We suggest an efficient data augmentation method to increase the data-samples which the lower freqeuncy agents are trained on, which improves the overall performance of this MARL framework, which is explained in Section 2.4 and demonstrated in Section 4.2.
- Finally, we compare the methods we proposed with many baseline algorithms, including single-agent PPO [6], DDPG [7], A2C [8] and regular MARL algorithms MAPPO [5], MAA2C [9] and MADDPG [10] algorithms in Section 4.

2 Proposed Methods

2.1 Hierarchical Multi-agent PPO

The hierarchical framework that we propose is as follows: let us consider a network of 5 agents that form a directed graph as in Figure 1.

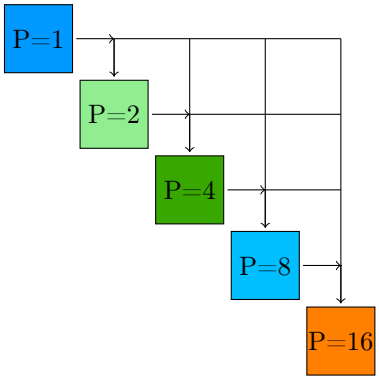

Figure 1: Hierarchical formation of agents, edges indicating the flow of information (actions and rewards in the previous states), and the node labels (P) indicates the trading timeframe of agents, i.e. P=2 agents trades once in every 2 days.

In that Figure, P denotes the trading period of the agents in terms of days, i.e. the highest frequency trading agent (P=1) trades every day, whereas the lowest frequency trading agent (P=16) trades once in every 16 days. The edges indicate the information flow. Here, the "information" shared between agents is the actions and rewards of the other (higher frequency) trading agents in the previous time steps. In particular, let us denote the action and reward of agent k at time step i with a_i^k, r_i^k. Also, let us denote the usual observation vector from stock market at time i with a vector $o_i \in \mathbb{R}^{18}$ (in our examples, the observations coming from the stock trading data are 18 dimensional vectors that include various information about the stock prices such as the Open, High, Low, Close, Volume information and various other financial indicators as we will explain in Section 3.2). Then, let us also denote the particular observation *seen* by agent k at time i with o_i^k. Then, for a network of 5 agents as in Figure 1, we can denote the final observation of the first three agents as:

$$o_i^1 = o_i, \quad o_i^2 = [o_i, a_{i-1}^1, r_{i-1}^1, a_i^1, r_i^1]$$

$$\begin{aligned} o_i^3 =& [o_i, a_{i-3}^1, r_{i-3}^1, a_{i-2}^1, r_{i-2}^1, a_{i-2}^2, r_{i-2}^2 \\ & , a_{i-1}^1, r_{i-1}^1, a_i^1, r_i^1, a_i^2, r_i^2] \end{aligned}$$

So the highest frequency trading agent, P=1 is oblivious to the information of other agents. P=2 agent sees the action and reward information of agent P=1, the third agent (P=4) sees the information coming from the first two agents (P=1 and P=2), and the final agent (P=16) sees the information coming from every other agent.

After formulating such a network of hierarchical agents, let us discuss how we train these agents. First, we train the highest frequency agent (P=1) using the PPO algorithm [6] on the training set (the details regarding the training set and test set as well as the environment characteristics are explained in Section 3). Then, we collect the actions and rewards of the first agent for every day on the training set, and append it to our dataset. Then, we train the second agent (P=2) on this dataset, such that it trades every two days and sees information

of the first agent. Then, we collect the actions and rewards of the second agent, append them to our dataset, and train the third agent using PPO and continue until all agents are trained.

2.2 Adaptive Selection of Trading Agents

Consider we have a hierarchical multi-agent setting with five agents as decribed in Section 2.1. The first agent with the highest frequency trades every day. The second agent trades once in every two days, and sees the actions and rewards of the first agent on the previous day and on that day. The third agent trades once in every 4 days, and sees the actions and rewards of the agents 1 and 2. The fourth agent trades once in every 8 days, seeing the actions and rewards of the first 3 agents, and the fifth agent trades once in every 16 days with the lowest frequency, seeing the actions and rewards of every other agent. Now, the natural question to ask is, that in such a multi-agent setup, who should take the actions and how should the profits of the system be calculated. One straightforward answer to this question is that every agent should take an action, and participated in the "trading" process, and at the end of the time horizon for the test dataset, the final balance of these agents should be averaged to calculated the average return of this team of agents. One other plausible answer to this question is that, since the least frequency agent (the one that trades in every 16 days) sees the information flow of every other agent and the information in general flows from high frequency to low frequency in this hierarchical setup, only the last agent should make decisions.

However, we figured out that there can be a better way to approach to this problem. The idea is that, can we develop an algorithm to adaptively select which agent to use. We can empirically quantify the volatility of the market. Then, for the times where the market is very volatile, when the effects of short term investments can be significant, we make the high frequency agents trade. However, when the volatility of the market is too low, we make the low frequency agents trade. To implement this algorithm, what we do is on our training dataset, we collect the Close values of the assets into an array and group them in groups of 16 (which is the lowest common multiple of the periods of our 5 agents in the MARL network). Then, we calculate the percentage change between consecutive days, and then compute the standard deviation of these percentage changes in daily close values of the assets. This gives us an empirical estimate of the volatility of the market. The, using the K-Means algorithm, we cluster these volatilities into five bins, which make up our thresholds for these five agents, Then, on the test set, for every 16 days, we calculate the volatility of the market, and based on the proximity to the 5 cluster means given by thge K-Means algorithm, we select which agent to trade. The, the balance of all the agents are equated. The experiments pertaining to this idea are discussed in Section 4.3.

2.3 Warm-starting Hierarchical Agents

One issue that we came across with training the hierarchical scheme introduced in Section 2.1 is this: even though we might achieve performance better than single agent PPO using a hierarchical multi-agent PPO formulation where we avergae the balances and let every agent trade or use an adaptive selection method as described above, there is no strict increase in the performance of the agents when we move from high frequency trading agents to low frequency trading agents. For instance, let us say that we trained the first agent that trades every day on 4096 days. Then, we collect its actions and rewards for these days, append it to our dataset of indicators (because they are going to be the part of the observation space fot the next lower frequency agents), and train the second agent that trades every two days. Note that to train this agent, we have 2048 days of data, since it trades every two days. Of course in return for this decrease in the training data samples, the second agent has a higher dimensional observation space than the first agent (as it sees the actions and rewards of the first agent for the previous day and the current day). However, as we move to the very low freuency trading agents such as agent with period 16, the number of trading samples decrease significantly, for instance only 256 days in this example. Empirically, we have seen that such a decrease in the number of training data samples cause the agents to not be sufficiently trained to learn how to act, to the point that agent 5 (with period 16) was usually losing money. Given that agent 1 (the most high frequent agent) was usually making positive profits (if that agent did poorly as well, we would not have considered this next algorithm), we came up with the following idea: when training the lower frequency trading agents, can we initialize the weights of the neural network parameters of the models with the trained models of the immediate higher frequency trading agent? Concretely, consider a hierarchical MARL framework of 3 agents, the first agent trades every day, the second agent trades every 2 days, the third agent trades once in every 4 days. Then, we first train the first agent, storing its actions and rewards for each training data sample it has been trained on. Then, we initialize the agent 2 with the parameters of agent

2, and then again using some RL algorithm (we used PPO), we train agent 2. Notice that when we initialize the weights of this agent with the weights of agent 1, we sort of imparted the knowledge of high frequency trading to this agent. However over time with regular RL training, this agent learns how to trade for its own timeframe, however now the policy is "built upon" the strategy of the previous agent. To train agent 3, we similarly initialize its weights from agent 2, giving it the actions and reward information of both agent 1 and 2. We refer to this way of training the agents as "warm-starting hierarchical agents".

2.4 Data Augmentation

Even though warm starting the agents in this framework caused some increase in performance, this following data augmentation trick allowed us to have the real performance boost in our agents, and really empower the multi-agent strategy over the single-agent ones. Recall that one issue with hierarchical training was that, as the trading frequency got lower (and hence the period gets larger) the low frequency agents have fractions of the data the higher trading agents see, making it hard to train those agents. In particular, consider the case with 3 agents. Then, for the first 8 days, let a_i^k denote the action of agent k at day i. Then, we can write the actions as:

$$a_1^1, a_2^1, a_2^2, a_3^1, a_4^1, a_4^2, a_4^3, a_5^1, a_6^1, a_6^2, a_7^1, a_8^1, a_8^2, a_8^3$$

Notice that agent 2 only trades for even days, and agent 3 trades for days that are multiples of 4. However, to augment the training procedure, we can simply shift the days the first agent trades, so that we have more data for the lower frequency trading agents, i.e.:

$$a_1^1, a_2^1, a_2^2, a_3^1, a_4^1, a_4^2, a_4^3, a_5^1, a_6^1, a_6^2, a_7^1, a_8^1, a_8^2, a_8^3$$

$$a_2^1, a_3^1, a_3^2, a_4^1, a_5^1, a_5^2, a_5^3, a_6^1, a_7^1, a_7^2, a_8^1, a_9^1, a_9^2, a_9^3$$

Hence, we make agent 2 trade on odd days as well. In this way, we can double the training data of agent 2, and by shifting the days two times more, we can quadruple the training data of agent 3 (with period 4). The experiments pertaining to this idea and the warm-starting idea in the section above are discussed in Section 4.2.

3 Experimental Setting

3.1 Data Collection

In our experiments, we used GBP/USD currency data. We obtained data from Yahoo Finance [2]. We collected daily data (so the smallest time increment between data points is 1 day), as Yahoo Finance was putting some limitations on hourly or minutely data. We trained the models with the data from 2004-01-01 to 2020-01-01, and tested the models with the data from 2020-01-02 to 2023-10-01 (date is formatted as year-month-day).

3.2 Trading Environment

We used the foreign currency trading environment proposed by finrl [11], [12] as a baseline. In our environment, at each time step the action is processed for a transaction to be processed to simulate real world settings. This transaction is processed in the next time step for the reward to be calculated.

3.2.1 Actions

The action space is a real number between 0 and 3, i.e. the action of agent k at time i is $0 \leq a_i^k < 3$. The integer part of the action corresponds to the action: 0 is buy, 1 is sell, and 2 is hold. The role the decimal part of the action plays will be described in section 3.2.3.

[2] https://finance.yahoo.com/

3.2.2 Observations

The observation space for an independent agent (not considering the information coming from other agents) is 18 dimensional. It includes regular financial data: Open, High, Low, Close, Minute, Hour, Day. It also includes technical indicators: macd, boll_ub, boll_lb, rsi_30, dx_30, close_30_sma, close_60_sma. The last four dimensions depend on the variance of the data in comparison with the agent's balance; they are: agent's balance, max_drawdown_pct, current_holding, current_drawdown. On top of these regular financial statistics, there can be additional information agents share with each other, as explained in Section 2.1.

3.2.3 Reward Model

If the action is between 0 and 2, the following steps are taken. Otherwise, the reward is zero.

At each step after the actions are declared, the following quantities are recorded: the close price of that step, stop loss max, and profit taken. Stop loss max (SL_{MAX})and profit taken max (PT_{MAX})are set constant while initializing the environment (at 300 and 2500 in our case). Profit taken (PT) is calculated in the following way (assuming that a_i is the action taken by same generic agent k at time i, we dropped the agent index k for generality):

$$PT_i = (a_i - \lfloor a_i \rfloor) \times PT_{\text{MAX}} + \text{SL}_{\text{MAX}}$$

In the next step, the Low and High price indicators for the stock are compared with the Close price indicator for the stock of the previous step. Denote them as L_{i+1}, H_{i+1}, and C_i. The reward is found according to the following algorithm 1 (let r_i be the reward of the i^{th} step):

Algorithm 1 Calculate the reward

if $\lfloor a_i \rfloor = 0$ **then**
  if $L_{i+1} < C_i - \frac{SL_{\text{MAX}}}{100000}$ **then**
    $r_i = -SL_{\text{MAX}}$
  else if $H_{i+1} > C_i + \frac{PT_i}{100000}$ **then**
    $r_i = PT_i$
  else
    $r_i = 0$
  end if
else
  if $H_{i+1} > C_i + \frac{SL_{\text{MAX}}}{100000}$ **then**
    $r_i = -SL_{\text{MAX}}$
  else if $L_{i+1} < C_i - \frac{PT_i}{100000}$ **then**
    $r_i = PT_i$
  else
    $r_i = 0$
  end if
end if

Please refer to this literature [13] for an analysis and explanation of how these financial quantities such as max drawdown are relevant for financial trading.

3.3 Baselines

These are the models against which we compare our proposed methods. Note that there was no proper implementation of MAPPO algorithm to the algorithmic trading dataset (as this algorithm was used mainly for multiagent game settings such as Starcraft or League of Legends), so we developed an implementation of this algorithm to this environment.

3.3.1 Single-agent algorithms from stablebaselines 3

The single-agent algorithms from Stablebaseline 3 include PPO [6], DDPG [7], and A2C [8] implemented in the traditional way. The performance of these agents are used as baselines for the MAPPO algorithms so they will

not be displayed here. It is worth noting that DDPG trades very conservatively, and PPO outperforms A2C in most cases.

3.3.2 MAPPO Algorithm

We developed one MAPPO agent as our baseline. This implementation also explores scalability, which will be described in section 4.1. The optimal number of independent agents in our experimental setting is around 3, since it performs the best in both the training and the test set.

The first MAPPO algorithm includes a supervising agent and several independent agents. The independent agents have an observation space of 18 dimensions. The supervising agent sees all the actions and rewards of the independent agents from the previous step. It has an observation space of (18+number of independent agents$\times$2) dimensions. Let N be the number of independent agents.

Algorithm 2 The MAPPO algorithm

Require: Two environments with 18 dimensions
for i = 1 to N: **do**
1: reset environment 1 to prepare for training
2: train independent agent i
3: test agent i on time period to obtain actions and rewards
4: append the actions and rewards columns to the environment 2
end for
Train supervision agent on environment 2

The performance of MAPPO as shown in figure 2 is better than single agent PPO. This is true not only for the test set shown in the figure, but also on average. On test sets, MAPPO has a higher variance, but on the training set it almost always outperforms PPO. We think this is true because the supervising agent has more exploration over the environment because it sees the actions and rewards of the independent agents.

3.3.3 MAA2C Algorithm

The MAA2C algorithm is trained in a similar manner. We trained N independent A2C agents. We obtained their rewards and actions as part of the supervision agent's environment.

MAA2C reaches negative returns on average. In figure 2, MAA2C outperforms A2C for most of the test set. On average, however, there is no clear sign that MAA2C outperforms A2C.

3.3.4 MADDPG Algorithm

The MADDPG algorithm uses the same framework. Without carefully tuning the parameters, DDPG trades conservatively and almost always performs the action of holding.

The MADDPG and single agent DDPG models, as one can see in figure 2, still performs much more conservatively than PPO or A2C models. They hold more frequently. In general, PPO agents perform better than DDPG agents, which perform better than A2C agents.

4 Experiments

4.1 Scalability of MARL in Algorithmic Finance

Consider the MAPPO algorithm one explained in 3.3.2. The performance of the supervision agent changes as the number of independent agents change, which is summarized in Figure 3. While having more independent agents enables more exploration for the supervision agent, having too many independent agents makes the supervision agent put less weight on their actions and rewards. In addition, another analysis on scalability is provided in Figure 4 using the hierarchical framework, where we show that higher period agents do not necessarily do better than lower frequency agents.

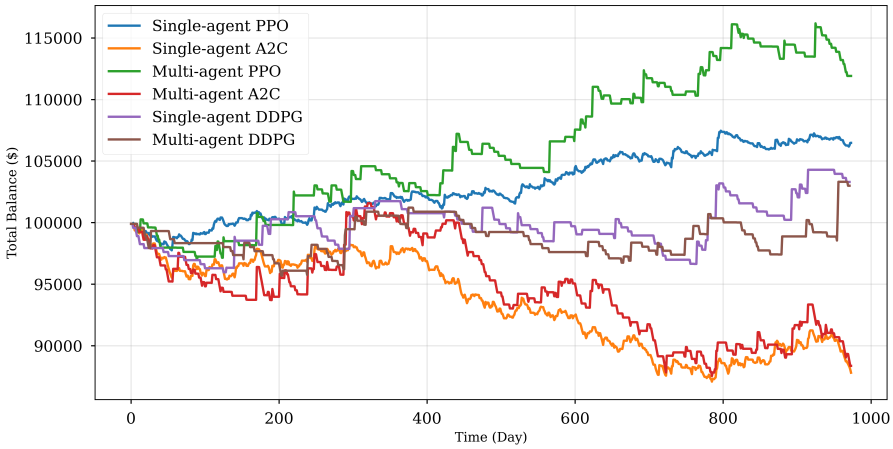

Figure 2: Test set comparison of MAPPO, MAA2C, MADDPG, all trained on two independent agents, labeled as multi agents, and the average performance of ten different PPO, A2C, and DDPG agents labeled as single agents.

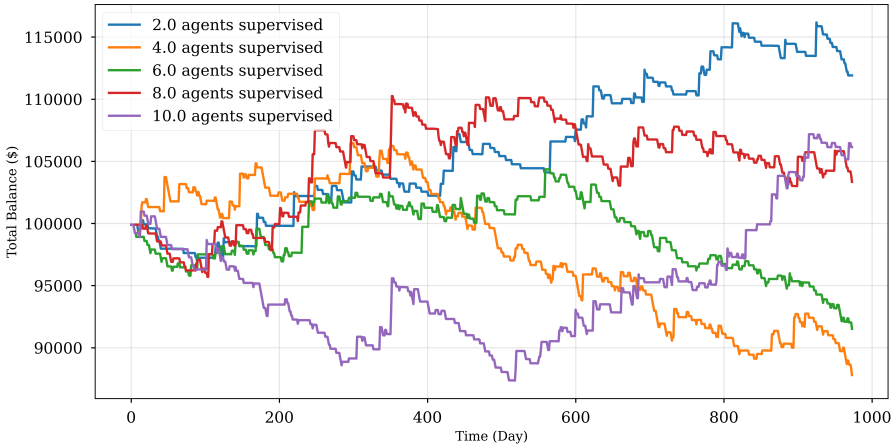

Figure 3: Test set performance of MAPPO algorithm one as the number of independent agents changes

4.2 Warm-starting Hierarchical Agents and Data Augmentation

Having discussed the experimental results for our research hypotheses (suitability of developing a multi-agent PPO to algorithmic finance and the issue of scalability in multi-agent RL), let us discuss our the results of our novel methods and how much of an improvement in performance they cause. First, we analyze the effect of the warm-starting the lower frequency agents with the trained weights of the agent whose frequency is immediately higher than that agent, as described in Section 2.3. The results are depicted in Figure 5. Notice that how much of an improvement in the performance of hierarchical MARL agents is achieved with this technique. Also, note that here, the overall balance of the MARL algorithm is calculated by letting all these individual agents in the MARL network trade and average out the balances. In Section 4.3, we will show a much better trading strategy for hierarchical MARL agents.

Now, one obvious question to be asked by looking at 5 is that, the single-agent PPO algorithm still seems not so bad from MARL algorithms proposed. The reason for this is, in these settings, the data with which the lower frequency agents is trained is significantly lower than the high frequency single agent PPO algorithm. However in Section 2.4, we proposed a data augmentation method to remedy this problem. Indeed, in Figure 6, we show that once we apply this data augmentation trick to our hierarchical agents (after using the aforementioned warm-starting method), there is a significant boost in performance. Notice how the blue curve in Figure 6 (corresponding to hierarchichal MARL PPO algorithm with warm-starting and data augmentation tricks applied) achieves better performance than single-agent PPO on any day. Now, before comparing our

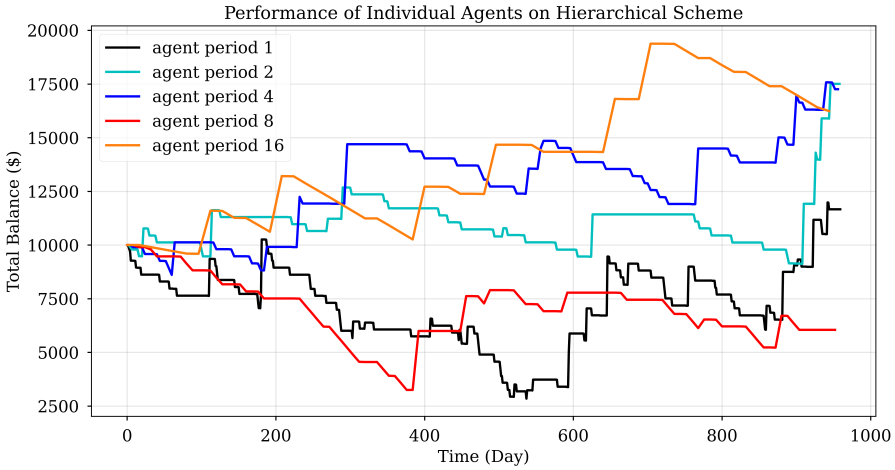

Figure 4: Performance of individual agents in the hierarchical model as described in Section 2.1 with data augmentation trick mentioned in 2.4 applied to ensure the total training days for these models are equal. Notice that the overall performance of the MARL framework does not necessarily scale with the number of increasing agents, as other plots in this section also indicate.

proposed model with other baselines, let us discuss how adaptive selection of agents can cause a further boost in performance in the next section.

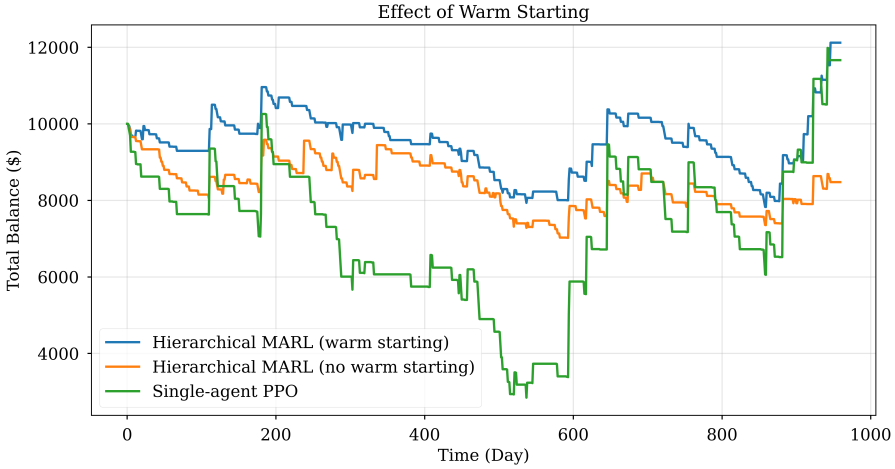

Figure 5: Effect of warm starting technique mentioned in Section 2.3 to the balance of the vanilla hierarchical ppo algorithm explained in Section 2.1 (no data augmentation). Warm-starting trick improves the performance of the MARL algorithm throughout the test data.

4.3 Adaptive Selection of Trading Agents

Now, let us briefly review what our experiments so far gained us until this point. We have a hierarchical multi-agent RL model where each individual agent is trained using PPO, using the information flow explained in Section 2.1, and the warm-starting trick and the data-augmentation methods described in Sections 2.3 and 2.4. We showed that such a multi-agent algorithm achieves much better result than single-agent PPO algorithm where all the agents in the network trade and at the end of trading their balances are averaged out.

However, as described in Section 2.2, we propose a better way to select which agents trade and to make a better use of the total balance of the agents. In particular, let us assume we have 5 agents that are trained as explained in the paragraph above. Then, we collect the Close prices of our stock for the days in the training set,

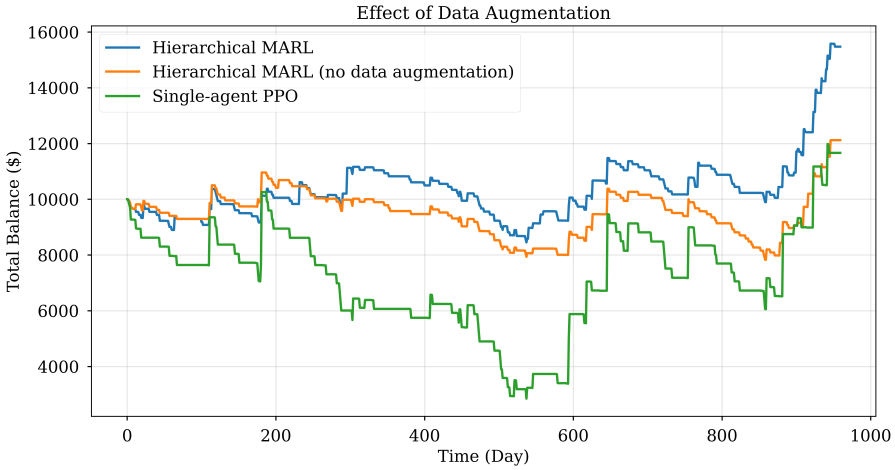

Figure 6: Effect of data augmentation technique mentioned in Section 2.4 to the balance of the vanilla hierarchical ppo algorithm explained in Section 2.1 (no data augmentation).

and then group this data by batches of 16, then calculate the standard variations of the percentage changes in these prices to estimate the volatility of the market. Then, we use K-Means algorithms to cluster this volatility data into 5 groups. Then, during inference, we look at the standard deviation of the daily change in Close price of the stock value for the last 16 days, and find out to which cluster is this volatility estimate close to. Then, if the volatility is classified to be in the cluster $k \in \{1, 2, 3, 4, 5\}$, we let agent k trade (notice that agent k trades once in every 2^{k-1} days as explained in Section 2.1. Then, we plot the result in Figure 7. The other trading strategy mentioned in that plot is simply using the lowest frequency trading agent. Notice that this agent does a really good job, especially considering that this is after the data-augmentation trick. Nonetheless, the best performance is obtained using the adaptive selection of agents trick. Indeed, this makes sense, because as hypothesized by the Fractal Market Hypothesis and Elliot Wave Theory, each agent is an expert of a specific time-frame, and we are making the best use of those collaborative agents by adaptively choosing who should trade given the market conditions. Finally, in Figure 8, we compare our method with many other baseline algorithms. After many novel modifications we proposed in this paper, our method achieves the best results by a large margin.

5 Group Members Contributions

Mert: Formulated and implemented the particular hierarchical multi-agent PPO algorithm in this project. Came up with and implemented the adaptive selection of agents method, also formulated and implemented the warm-starting of agents method, and implemented data-augmentation methods proposed in this project. Built the trading gym environment for hierarchical multi-agent RL algorithms.
Jesse: Implemented the single agent RL algorithms and formulated a suitable implementation of MAPPO, MAA2C and MADDPG for the currency trading environment as baseline algorithms. Set up the environment for the general MARL methods in literature, and contributed to the idea formation of adaptive selection of agents.

Acknowledgement

We thank to the teaching assistants of the course for helpful discussions regarding the proposed methods in our project, espcially the data augmentation and warm-starting of agents.

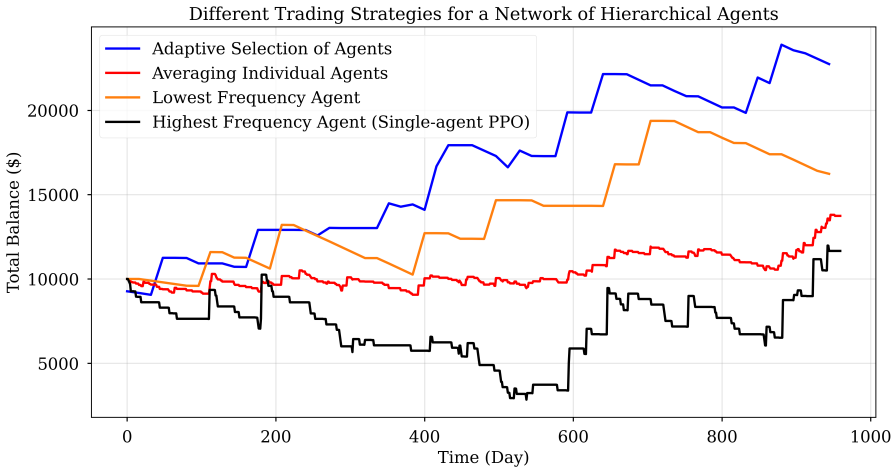

Figure 7: Comparison of different trading strategies for a hierarchical network of 5 agents as in Figure 1. The highest frequency agent trades every day, and it corresponds to the single-agent PPO algorithm. The lowest frequency agent trades once in every 16 days and sees the actions and rewards of every other agents in the previous 16 days. Averaging case (red plot) means that all 5 agents trade, and their balances are averaged out. Adaptive scheme (blue curve) shows the superior performance, and it is done according to the algorithm explained in Section 2.2. Note that for hierarchical framework in this experiment, both the warm-starting and data augmentation tricks which have been introduced before have been applied.

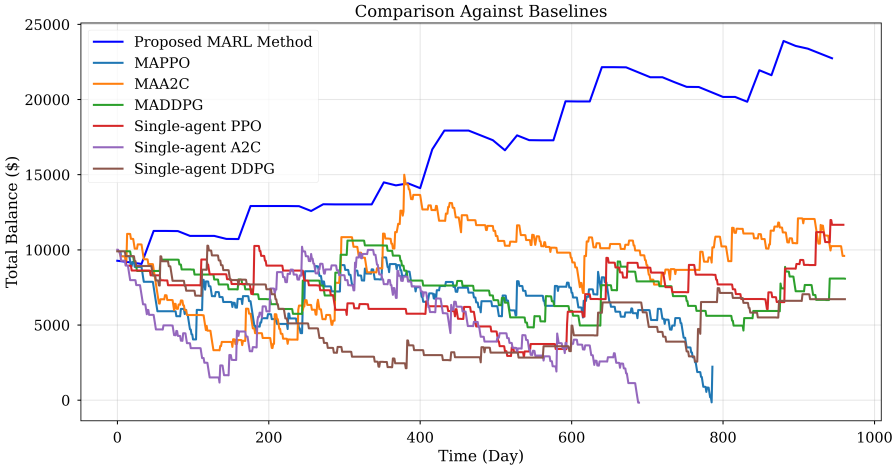

Figure 8: Comparison against baselines. Here, the proposed method refers to the case where we use the adaptive selection algorithm explained in Section 2.2, together with the warm-starting technique as explained in Section 2.3 with the data augmentation method mention in Section 2.4 on the hierarchical algorithm explained in Section 2.1. The MARL algorithms here (including ours, MAPPO, MAA2C and MADDPG) has 5 agents.

References

[1] A. Weron and R. Weron, "Fractal market hypothesis and two power-laws," *Chaos, Solitons & Fractals*, vol. 11, no. 1-3, pp. 289–296, 2000.

[2] A. J. Frost, R. R. Prechter, J. C. Charles, and W. E. White, "Elliott wave principle: key to stock market profits," *(No Title)*, 1990.

[3] A. Shavandi and M. Khedmati, "A multi-agent deep reinforcement learning framework for algorithmic trading in financial markets," *Expert Systems with Applications*, vol. 208, p. 118124, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422013082

[4] Y. Huang, C. Zhou, K. Cui, and X. Lu, "A multi-agent reinforcement learning framework for optimizing financial trading strategies based on timesnet," *Expert Systems with Applications*, vol. 237, p. 121502, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423020043

[5] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, 2022.

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[9] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2019.

[10] Z. Yan and Y. Xu, "A multi-agent deep reinforcement learning method for cooperative load frequency control of a multi-area power system," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4599–4608, 2020.

[11] X.-Y. Liu, J. Rui, J. Gao, L. Yang, H. Yang, Z. Wang, C. D. Wang, and G. Jian, "FinRL-Meta: Data-driven deep reinforcementlearning in quantitative finance," *Data-Centric AI Workshop, NeurIPS*, 2021.

[12] X.-Y. Liu, Z. Xia, J. Rui, J. Gao, H. Yang, M. Zhu, C. D. Wang, Z. Wang, and J. Guo, "FinRL-Meta: Market environments and benchmarks for data-driven financial reinforcement learning," *NeurIPS*, 2022.

[13] M. Magdon-Ismail and A. F. Atiya, "Maximum drawdown," *Risk Magazine*, vol. 17, no. 10, pp. 99–102, 2004.