Applying Artificial Intelligence & Reinforcement Learning Methods Towards Improving Execution Outcomes

Diana Kafkes*, Josep Puig Ruiz*, Drew Rooks*, Douglas Hamilton*, and Michael O'Rourke*

Nasdaq Stock Exchange

October 10, 2022

1 Introduction

Capital markets are dynamic and always evolving. This constant evolution provides opportunities to enhance timer based solutions to improve execution outcomes and further mitigate adverse selection and volatility. This paper proposes the introduction of a dynamic model-based timer that can be applied to the Nasdaq Midpoint Extended Life Order (M-ELO).

The current M-ELO uses a holding period— a brief waiting timer applied to the queue on both sides— statically set at 10ms (10^{-3} s). This holding period achieves relatively favorable markout when trading institutional loads with respect to the continuous book, achieving the intended purpose of matching like-minded long-term investors. However, this occurs at some expense of fill rate given the differences in the dynamics of individual securities. Our research shows that the interplay between fill rate and markout is more nuanced than a simple trade-off, as previous literature indicates. Here we demonstrate that achieving both higher fill rate and lower markout

*diana.kafkes@nasdaq.com, josep.ruiz@nasdaq.com, drew.rooks@nasdaq.com, douglas.hamilton@nasdaq.com, michael.orourke@nasdaq.com

is possible through leveraging Artificial Intelligence (AI) towards achieving dynamically-improved timer execution.

Our Dynamic M-ELO system[1] leverages an AI control scheme known as reinforcement learning to evaluate the duration of the holding period timer based on local market conditions. Our research shows that applying a dynamic timer to MELO achieves an increase in fill rate of 20.3% and a simultaneous decrease in markout of 11.4% compared to the current static holding period. We believe that these results are indicative of the many improvements AI-enhanced mechanisms can bring to capital markets. Here we detail the development and results of our proposed timer-update model and advance its adoption as a step towards the future of dynamic market order types.

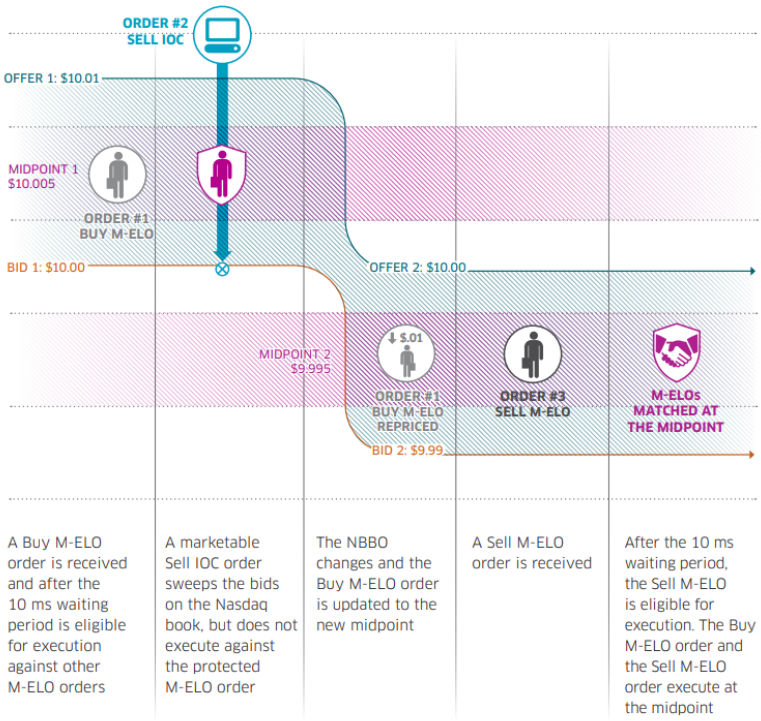


Figure 1: An example of a M-ELO trade.

[1]Our Dynamic M-ELO system is patent pending; we have filed several patent applications covering various aspects of it.

2 Assessing the State of the Current Static Holding Period

First, we present some details on how the current holding period used with M-ELO[2] functions in order to provide context for the improvement gains our system is able to make on the original.

The current static holding period market is designed to match like-minded longer-term investors on a broker-neutral exchange. When a M-ELO order is entered, it is held for a period of time before it becomes executable (Figure 1). This period of time depends both on the static holding period timer and a M-ELO order eligibility condition.

2.1 M-ELO Eligibility to Trade

Incoming M-ELO orders in the current implementation are assigned a static waiting timer of 10ms. This timer was updated from its earlier value of 500ms in May 2020.

This timer is only activated by a specific midpoint-crossing eligibility condition: the NBBO (National Best Bid and Offer) of the security or fund must have favorably crossed the price of the entered order. For buy orders, this means that the orders become eligible if the midpoint of the NBBO for that security is either at or lower than the bid price of the M-ELO order. Similarly, for sell orders, the midpoint of the NBBO must be either at or higher than the ask price of the M-ELO order in order to be eligible. As soon as an order becomes eligible by this criteria, it begins its waiting period of 10ms.

It is important to note that the waiting timer is not set off immediately when an order is entered, and to distinguish the two distinct waiting periods an order faces in the M-ELO queue: one before it is deemed eligible based on a midpoint-crossing condition and one that depends on the holding period timer assigned, in this case 10ms.

[2]Please note that the proposed system can be applied to both Nasdaq Midpoint Extended Life Order (M-ELO) and Nasdaq Midpoint Extended Life Order Plus Continuous Book (M-ELO+CB). For the sake of brevity in notation, and unless otherwise noted, we will henceforth refer to both M-ELO and M-ELO+CB as simply "M-ELO".

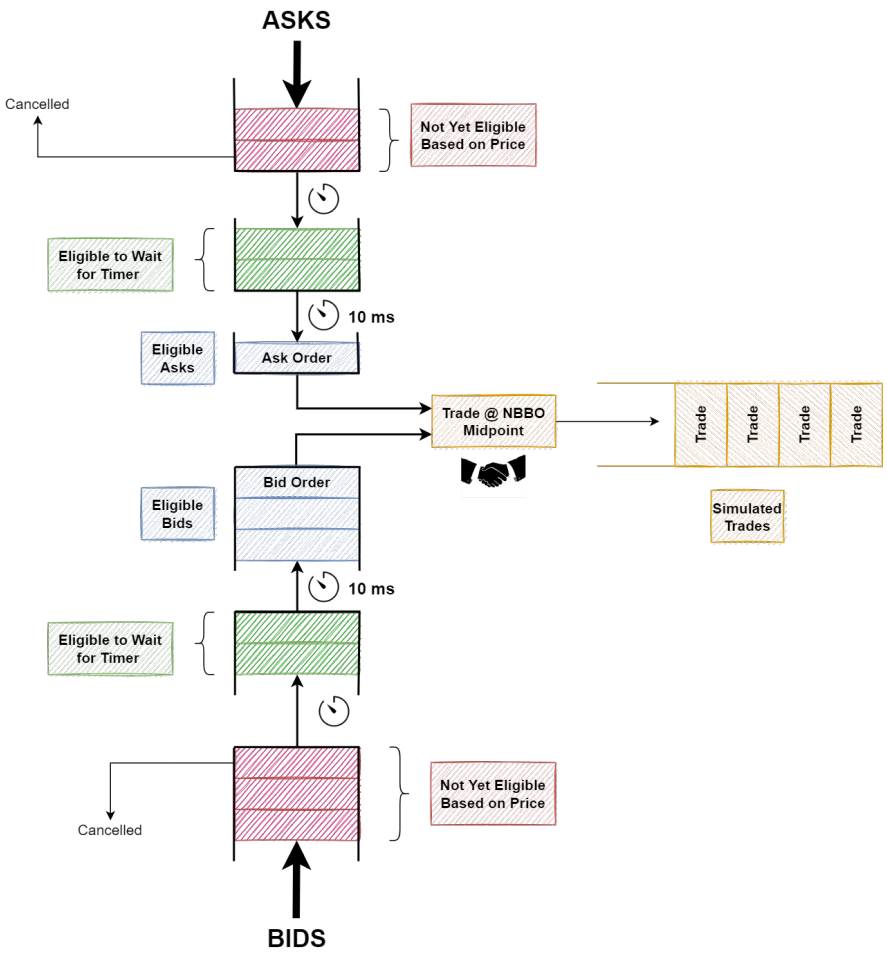


Figure 2: How Orders Become Eligible To Trade on M-ELO

2.2 Fill Rate-Markout Trade-off

While all orders are treated equally by the assignment of the same timer in the simplest notion of the word, certain trade-offs arise from the static timer's interactions with non-static market conditions that affect orders unequally.

2.2.1 Defining Fill Rate and Markout

In order to address these inefficiencies, we first must define two metrics that give us some notion for the underlying liquidity and execution quality of the market: fill rate and markout, respectively [**5**, **6**, **7**]. We will refer to these

metrics throughout this White Paper in order to evaluate the performance of the current M-ELO market as well as our proposed adjustment to it that achieves the best of both worlds: increased fill rates and decreased markout. We define fill rate for a period of time and markout by trade as follows:

$$\text{FR}_{\text{period}} = \frac{\text{Shares Traded}}{\text{Total Incoming Shares}} \tag{1}$$

$$\text{MO}_{\delta,\text{bps,trade}} = \begin{cases} MO_{\delta,\text{bps,trade Buy}} = |\frac{-100*100(M_t - M_{t+\delta})}{M_{t+\delta}}| \\ MO_{\delta,\text{bps,trade Sell}} = |\frac{100*100(M_t - M_{t+\delta})}{M_{t+\delta}}| \end{cases} \tag{2}$$

where M_t is the midpoint of the NBBO at time t and δ_t denotes a specific time horizon.

Note here that the markout is always positive, i.e. it is defined as the absolute value of the price (dis)improvement for each side and counts as a penalty for any price movement within the time horizon. The negative sign inside the absolute value operator for the "Buy" markout component is not strictly needed, but we've decided to leave it in place to highlight the fact that generally price improvement for one side of a trade is met with price disimprovement for the other side of the same trade. Regardless, the goal is to minimize price deviation in *any* direction, hence the absolute value operator.

It's also worth noting that, while we considered many different time horizons in our research, here we will focus on discussing the specific case with $\delta_t = 1$ second (1s), i.e. the markout expressed in basis points (bps) one second after trading. Different time horizons yield similar results.

Lastly, note also that we are using a simplified aggregate shares formula for fill rate instead of fulfillment $\times$ hit rate. Using this alternative definition also yields similar results.

2.2.2 Static Timer vs. Non-Static Market Conditions

We propose here a solution that leverages AI to provide a bespoke intraday symbol-specific recommendation for the holding period at any given time, based on the specific conditions for that symbol. We find that by allowing for the holding period to change for each specific security and every 30 second interval we can further enhance the favorable outcomes that M-ELO participants already see.

We present a few simple examples to show how M-ELO currently operates and why it is effective. After that, we will demonstrate as well the additional value of having a changing holding period under certain situations.

We present first two examples that showcase the current behavior of M-ELO, and why it works well in providing favorable execution for like-minded parties, through Figures 3a and 3b. Two M-ELO orders, A and B, are placed, wait for the holding period of 10ms to elapse, and then become eligible to trade at the midpoint of the NBBO.

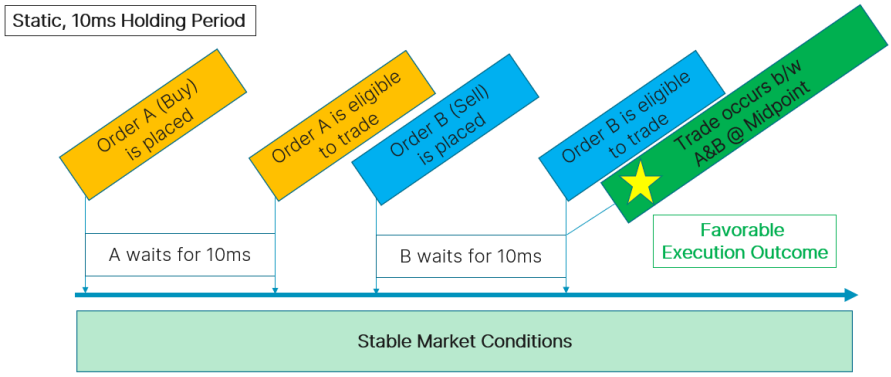
- Figure 3a depicts a situation in which the market conditions are very stable throughout. Under such circumstances, M-ELO allows for execution to occur, matching the two like-minded parties, and the outcome is satisfactory to both sides.

- Similarly, Figure 3b shows a situation in which market conditions get momentarily unstable, and price fluctuates significantly. Since the order that arrived later needs to wait as well for the holding period to elapse, the execution does not occur during this unstable moment, but rather occurs at a slightly later time, once the new price level is established. As a result, the later execution is favorable to both sides.

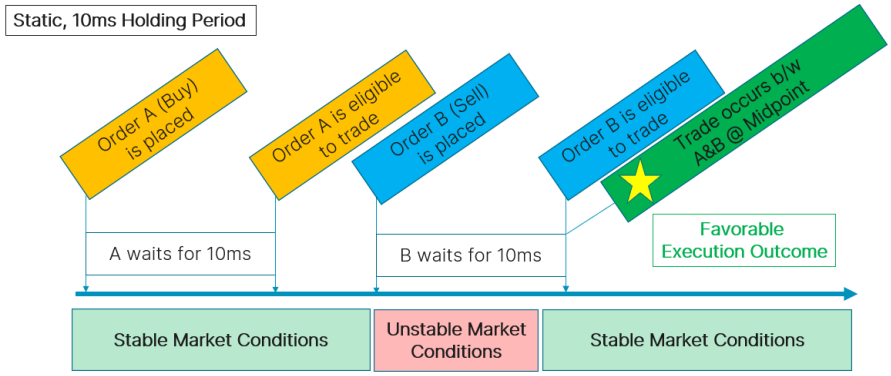
Now, we highlight how a changing holding period can further increase the performance of M-ELO. Order A is placed, waits for 10ms, and is now ready to trade, but there is no liquidity on the other side yet, so it rests. Moments later, a sell M-ELO order, Order B, is placed too. The conditions at this point in time are very stable for this symbol: if Order B was to trade immediately with Order A, the execution outcome would be favorable for both sides.

However, Order B needs to wait as well for the holding period to elapse. We explore two possible situations based on the duration of the holding period:

- With the static timer of 10ms, Order B needs to wait for 10ms. At some point, while waiting for the 10ms to elapse, market conditions worsen. Once the timer has elapsed, Order B becomes eligible to trade. Consequently, a trade occurs between Orders A & B at the midpoint of the NBBO. Due to the recent instability, the execution outcome is less favorable. This situation is depicted in Figure 4a.



(a) M-ELO in normal market conditions.

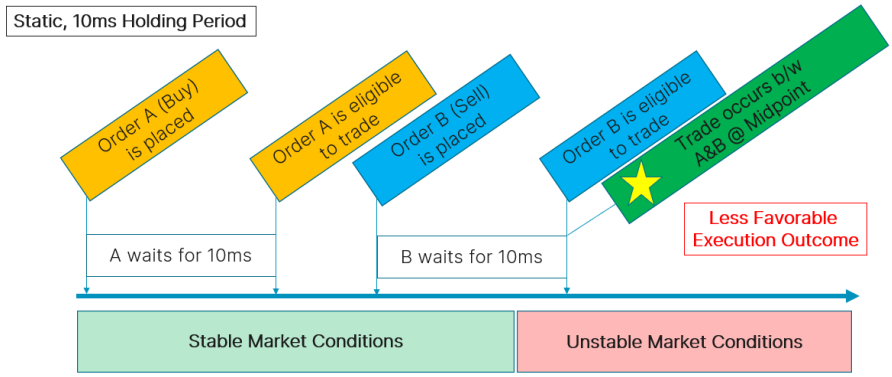


(b) M-ELO in unstable market conditions.

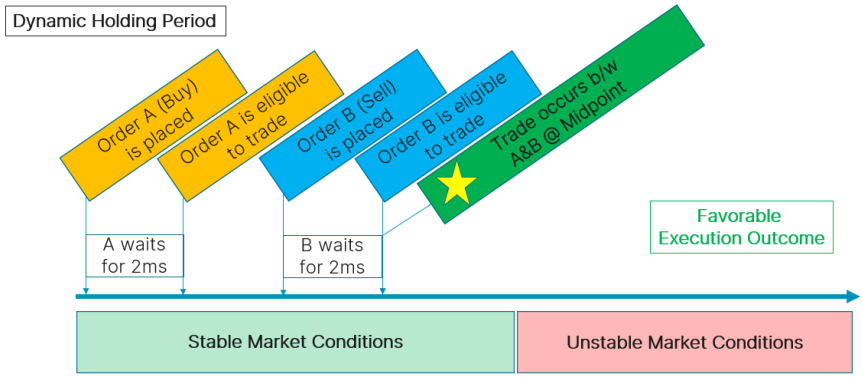
Figure 3: Example showing how current static M-ELO operates and how it effectively provides favorable execution outcomes that would have otherwise occurred in both normal and unstable market conditions.

- In contrast, assume a shorter dynamic timer, for example 2ms, and a stable market. The shorter dynamic allows A & B to trade sooner, which is a more favorable outcome for both parties during a stable period in the market. It is indeed a current market reality that execution algorithms brokers provide to asset managers and other institutions often need to rebalance liquidity across venues which may cause orders that could potentially match to miss each other due to a timer that is longer than it needs to be to provide the right level of protection. Fewer missed executions increases the efficacy of those broker algorithms and

therefore improves investor experience in M-ELO. Indeed, there are situations in which a changing holding period provides desirable outcomes, highlighting the need for the proposed solution. This situation is depicted in Figure 4b.



(a) Static timer might result in less favorable outcomes under specific circumstances.



(b) Shorter timer might result in more favorable outcomes under specific circumstances.

Figure 4: Example where a static timer might result in a less favorable execution outcome, showcasing the value of dynamically reducing the timer under specific circumstances.

Lastly, for the rare situation in which market conditions significantly deteriorate and a longer holding period might be desirable to shield against

volatility, the stability protection mechanism will be momentarily activated, increasing the holding period to a higher value for a short amount of time. This mechanism, which is described in more depth in Section 4.3, is depicted in Figure 5.

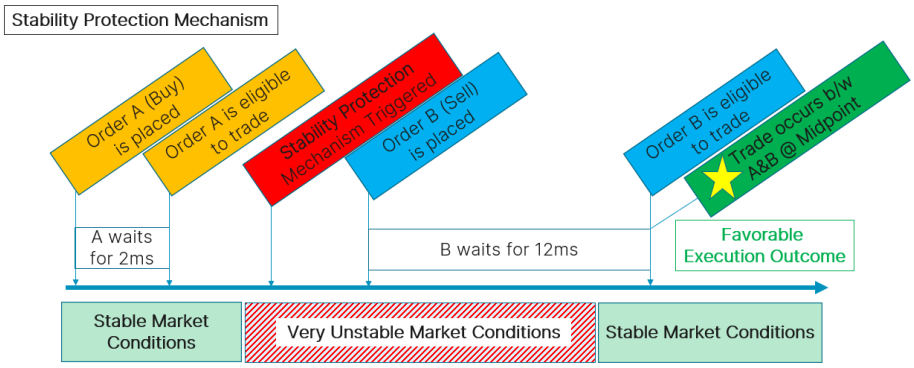


Figure 5: The stability protection mechanism acts as a safeguard against undesired execution outcomes driven by periods of high, unusual volatility. More details on this can be found in Section 4.3

3 Building an AI-Based Timer-Control System

Dynamic M-ELO is an AI-based control system that learns from watching market behavior and can dynamically adjust the holding period timer to improve client outcomes with respect to both liquidity and execution quality. The model evaluates changes to the timer between the range of .25 to 2.5ms by .25ms increments every 30 seconds (780 times per day) and includes stability protection enhancement (discussed at length in Section 4.3) that briefly overrides the timer to 12ms in the case of high volatility.

Here we discuss the methodology involved in the creation of Dynamic M-ELO, including an overview of reinforcement learning, how we built and validated a surrogate environment for our training process, and insight into our model development.

3.1 Reinforcement Learning

We framed our dynamic timer control solution as a reinforcement learning problem. We did this because we needed a way for our model to interact with and receive feedback from the trading system based on each timer chosen. Other machine learning approaches would have involved simulating a large or even intractably-large—depending on the frequency of timer updates chosen— number of timer-paths across our chosen learning period. This made reinforcement learning the clear choice.

Reinforcement learning is an AI paradigm in which a model ("agent") is trained to take the most optimal actions in an environment. This is achieved by the model taking random actions at first and receiving feedback— both positive and negative— in the form of a reward that is then used to tailor the model's future approach **[3]**. Overtime, the agent gradually switches from exploring its environment through random actions to exploiting what it has learned through these earlier actions **[3]**. The training of the model in this way involves a loop (Figure 6) that borrows heavily from the psychology of how we learn to walk or ride a bike— with all the falls, cuts, and scrapes included on the way to eventually achieving balance.

In a generic single iteration known as an episode: the agent takes an action and receives feedback from the environment in the form of both a reward and the next state from which it can take another action. Each time around the loop generates a (state, action, reward, next state) "experience" which is stored in what is known as the agent's memory buffer and accessed

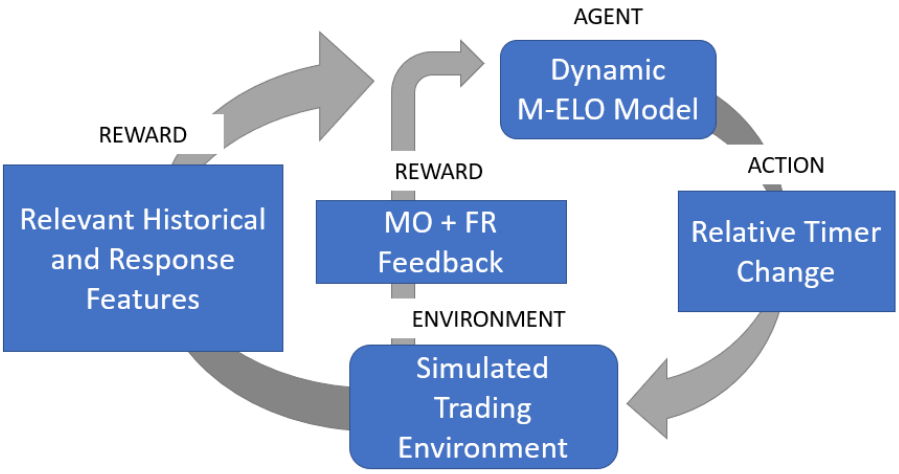


Figure 6: The Reinforcement Learning framework used to train Dynamic M-ELO.

for training [**3**].

Likewise, in our Dynamic M-ELO training process, we have a model (agent) select a relative timer change (action) that then impacts a simulated trading environment which provides feedback in the form of the next state and reward. In this case, the next state provided is the relevant historical and response features returned from the environment and the reward is a share-weighted linear combination of fill rate and markout, the proxies for liquidity and execution quality we discussed before in Equations 1 and 2:

$$\frac{\lambda}{\sum q_i} \sum_{i=1}^{T} q_i (\mathrm{MO}_{\delta,\mathrm{agent,bps}} - \mathrm{MO}_{\delta,\mathrm{synthetic@10,bps}}) + (1-\lambda)(\mathrm{FR}_{\mathrm{agent}} - \mathrm{FR}_{\mathrm{sim@10}}) \tag{3}$$

where q_i is the number of shares traded in trade i, T is the total number of trades in a period, and λ is a weighting design factor. This can be interpreted as the marginal advantage the Dynamic M-ELO agent is able to achieve with respect to the 10ms M-ELO as represented in our simulated environment. For a comprehensive explanation of how the simulated environment at 10ms

corresponds with historical M-ELO data, please see Section 5.1.

3.2 Developing a Simulation of M-ELO with a Non-Static Holding Period

In order to train our agent, we first had to build a simulated M-ELO trading environment. Without it, the Dynamic M-ELO agent would not have had a safe place to interact with and learn from without causing undue damage. This is often done in reinforcement learning to ensure that the actual environment is not disturbed. Within this simulated trading environment, the agent is able to change the duration of the holding period timer and receive positive and negative feedback relating how the environment's internal state reacts to this timer change.

The simulation built mostly consists of the M-ELO order eligibility logic detailed in Section 2.1 wrapped around a matching engine that ingests historical and slightly modified data. This matching engine accepts orders, cancellations, and replacements and has the basic capability of matching buy/sell orders of the same symbol at the same price.

The reason why slightly modified data is needed as an input to the matching engine is that the simulation is built to show what the trading environment *would have been* like at timers other than 10ms. This timer change importantly affects user's cancellation behavior.

3.2.1 Simulating Cancellations

We defined a way to statistically model appropriate cancellation behavior at timers other than 10ms based on historical past behavior. For each order without historical cancellation after the 10ms holding period, we sample from a binomial distribution to figure out whether or not the order would have a chance of cancelling. This binomial distribution was fit using the likelihood of that user cancelling for each specific symbol at a given time of day. If the user in question did not historically cancel that symbol at that time of day, we used that user's overall probability of cancelling for that symbol in general.

If the results of the binomial draw were favorable for cancellation, we then sampled the time interval between when the order was placed and when the cancellation would occur from an exponential distribution. This exponential distribution's parameters were fit from past user historical cancellation

behavior for that symbol at that time of day. Similarly to the likelihood of cancelling calculation, missing values were filled in first by the user for that symbol, and then for that user in general.

3.2.2 Validating our Simulation

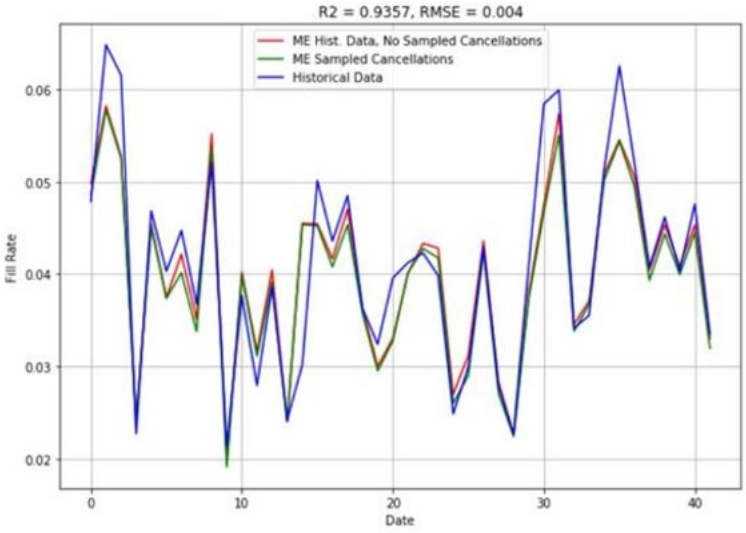


Figure 7: Validation of simulation against historical with and without simulated cancellation behavior. Results presented are a subset.

We ran our simulation forward at the 10ms timer to validate it. We received high RMSE values (averaging above .95, and all greater than .88) when plotting our resulting trades for specific symbols in comparison with the historical. The overall trend is that our simulation tends to undershoot the historical trades by a small amount. Performing this study at 10ms verified that we could replicate historical M-ELO activity with high fidelity and gave us high confidence that we could use this simulation to provide a reasonable estimate for the trading environment at different timers. Therefore, we could use the simulation as a training ground for our agent.

4 Model Development

Our timer-set agent is a deep neural network with multiple layers and over 35,000 parameters. It was trained following the DDQN (Double Deep Q-Network) training paradigm, where the agent's state-action value (Q-value) is approximated using a deep neural network [**2**].

4.1 Understanding DDQN

The Q-value provides some notion of the long-term value of a discrete sum of the agent's discounted actions [**1**]. To make this more concrete, the agent's actions will update the state it is in, sometimes in favorable ways and sometimes in nonfavorable ways. The favorable updates will increase the Q-value, and the nonfavorable actions will decrease the Q-value. These incremental updates are summed together for a discrete number of times around the RL Loop (Figure 6) with a nearsighted discount factor that makes Q-value updates that happened earlier in the chain decrease in impact as time goes by. Through many iterations of the training loop, the Q-value can be optimized.

The DDQN algorithm is a variant of the DQN algorithm, and involves initializing two identical models— a main and target network— and updating the target model's weights less frequently than the main model's [**2**]. While the main model goes around the loop and collects batches of (state, action, reward, next state) experiences in its memory buffer, the target model's parameter updates happen less frequently than the main model's for the sake of stability. This means that the target model's parameters cannot become over-adapted to one set of experiences, rendering it more stable than the main network which adapts itself to each set of sampled experiences. This lag is advantageous as the main model's parameters are periodically updated to be a weighted average of the main model and the more stable target model, ensuring it retains a middle ground of adaptation.

4.2 The Training Process

We trained our agent on data derived from our simulation operating between January 1, 2022 and April 1, 2022 (Q1)— a time period which we assessed to be representative in both duration and market behavior, including a range of relevant market volatility conditions. From this period of time, we used 380 symbols that represent a subset of the 6257 symbols that are actively

traded with M-ELO. This subset covers 67% of the current M-ELO volume and includes both tick- and nontick-constrained stocks.

Each trading day that the agent experiences consists of 380 (number of symbols) symbol-days. Each symbol-day is discretized into 30 second periods that our agent steps through, meaning there are (number of trading days) $*$ (number of tickers) $*$ ($780 \times \frac{\text{30 sec periods}}{\text{day}}$) episodes— times around the RL Loop— in our training process.

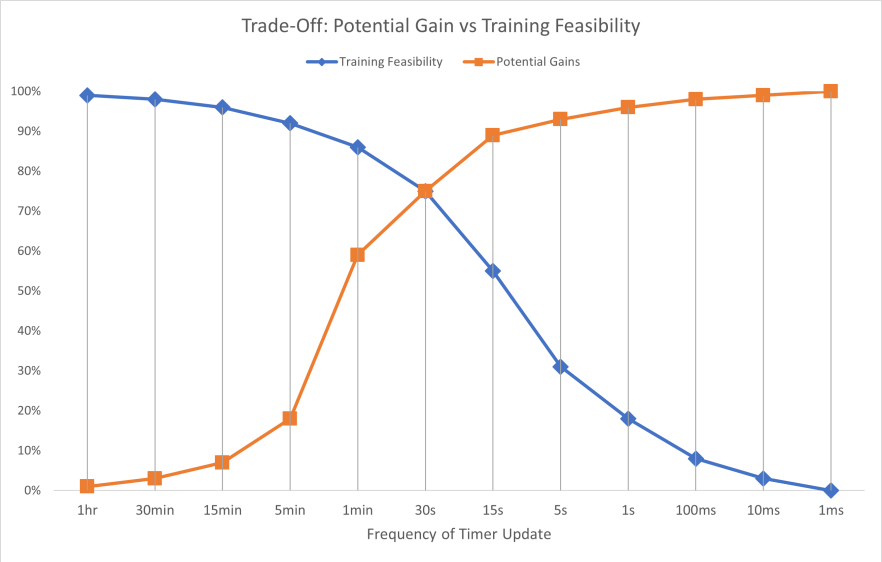
We restrict our agent to take 5 possible actions each 30 second period: it is able to change the timer by [-.5, -.25, 0, +.25, +.5] ms. It does this by evaluating the past 30 seconds of market conditions which are output from our simulation. This includes historically-based information such as features derived from the one- and five-day moving averaged M-ELO historical orders and trades, NBBO historical moments, the continuous book; as well as timer-dependent information such as the resultant fill rate, markout, and metrics derived from agent-made trades and simulated M-ELO activity.

There are 142 features in total, and they make up the state and next state part of the (state, action, reward, next state) experiences that the agent samples at the end of each calendar day to train from. For more technical details regarding the training process, including the pseudo-code, please see Appendix 7.

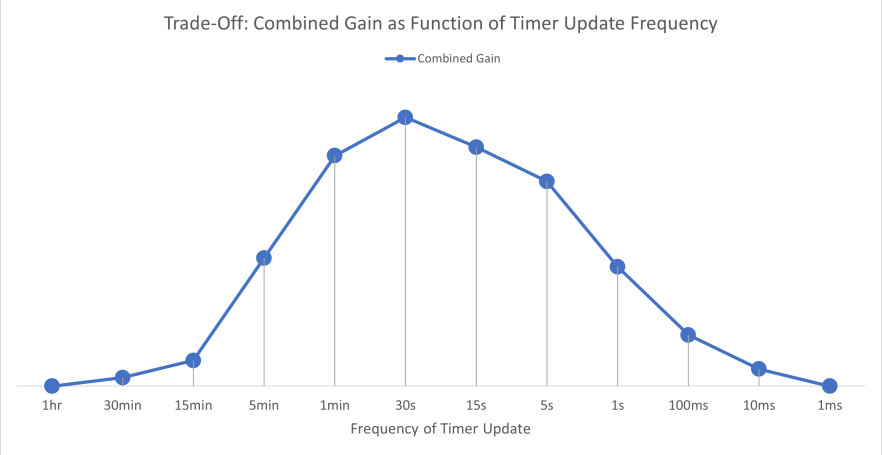
4.2.1 Motivating Allowable Timer Values

The optimal range of timers was chosen through substantial experimentation. First, we researched the lowest value the timer could take while retaining similar markout to those achieved by the original M-ELO during stable periods. We then found a timer that could effectively mitigate markout across periods of mild instability. For extremely unstable periods (which will be properly defined in Section 4.3), we chose 12ms as our stability protection timer since we found it to be the lowest value that achieves substantial gains over the current M-ELO during high volatility events.

Likewise, the 30 second update cadence was chosen by considering the difficulty of learning at different frequencies and our system's latency requirements. We assert that the theoretical best-possible timer would update with every incoming order. However, training a reinforcement learning agent on every order would be a difficult problem to generalize, resulting in a brittle model. Additionally, despite recent advances in fast electronics, this would be difficult to implement in real-time where nanosecond-latency is required. We



(a) Trade-off between training feasibility and potential upside gains, as a function of the frequency of timer update.



(b) Optimal frequency of timer update, based on the trade-off above.

Figure 8: Choosing the frequency of timer update.

investigated achievable update cadences and found that the point of marginal returns was between 15 and 30 second updates, and decided to go with 30 seconds to give our agent the best chance of learning generalizable actions (Figure 8).

4.2.2 Model Maintenance in Production

We can expect M-ELO environment market dynamics, like any other capital market dynamics, to evolve over time. These sorts of changes account for what is known in statistics as distribution shift. In general, models are not guaranteed to perform well when tested on data that has shifted from the training set.

Given this issue, we implement a form of transfer learning known as retraining to combat the natural performance degradation associated with distribution shift in deep learning. This retraining leverages the idea that training and testing the model on more recent market conditions will yield better results. After all, market conditions of a given day tend to be generally more similar to the the day/week before than they are to previous months.

Retraining involves freezing the weights in the first few layers of our network after a certain point, and allowing the other weights to train on more recent data. Here we commence retraining upon reaching an indicated retrain start date in the simulated environment.

We implement two retraining independent schedules: one for daily retraining and another one for weekly retraining. Both retraining schedules involve two sequential stages. In the first stage, all 35,000+ parameters of the model are trained until the specific retrain start date, in this case January 20, 2022. Then, in the second stage, some of the initial layers of the model are frozen and the end ones are allowed to update based on the most recent data, generating many additional models that are tied to being tested for specific dates. This process is depicted in Figure 9

In the case of the daily retraining, the unfrozen parameters are updated at the end of each subsequent day and saved to be tested on the next day. Likewise, in the case of the weekly retraining, the training process updates the unfrozen parameters throughout days which constitute the subsequent trading week. This model is saved and then tested on the next week. These schedules proceed iteratively until all days or weeks being tested are covered by retrained models.

Beyond the freezing of parameters to allow part of the model to adapt to

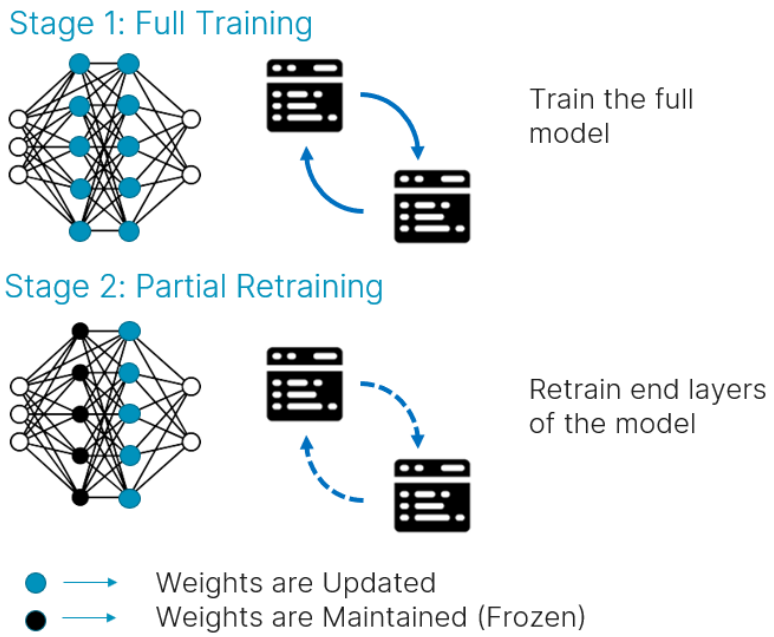


Figure 9: Visual representation of the proposed training and retraining schedule, using a simplified neural network with three input neurons, two hidden layers, and two outputs.

more recent temporally-local conditions, we also recommend the full retraining of the model on a monthly to quarterly basis to ensure it keeps up with longer-scale larger distribution shift.

4.3 Stability Protection Mechanism

Finally, as a system safeguard against volatility, we added a stability-protection mechanism to our timer-set model. In early development, we noted a significant number of one-off adverse markout events during some 30 second periods. This motivated us to design such a mechanism reactive to changes in the NBBO price.

Often these volatility events happen at a much faster frequency than the available timer change cadence, so we developed a method capable of reacting to these events at a more appropriate scale on the order of seconds. We verified that this mechanism improves M-ELO execution quality and

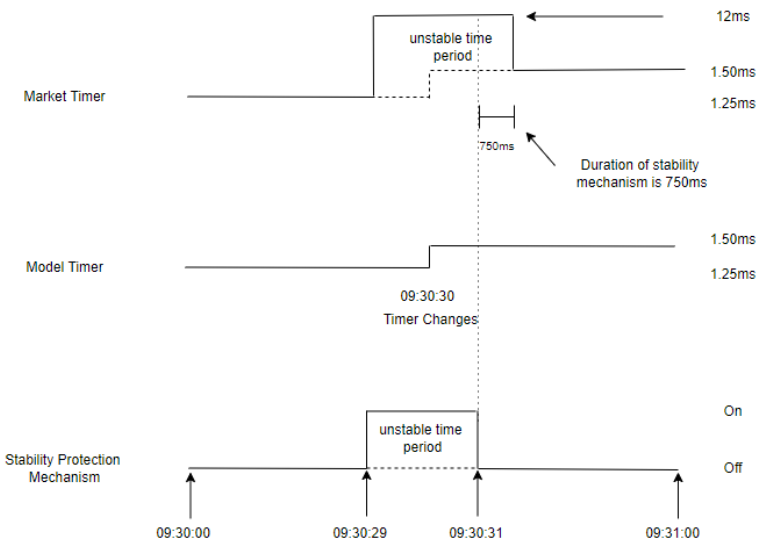


Figure 10: The market timer is set by combining the model timer and the signal from the stability protection mechanism.

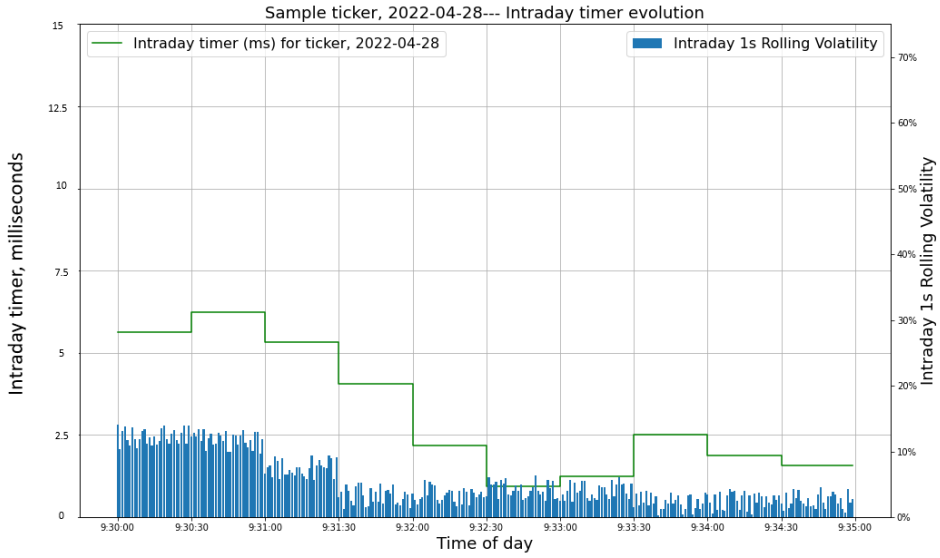
performance overall, and protects participants in unstable periods.

Periods which trigger our instability condition (see below) cause the timer to be set to 12ms for 750ms, thereby mitigating negative impact from unfavorable execution. It is important to note that this 750ms duration is independent of the model setting the timer every 30 second period. The market timer is determined by combining the model timer and the stability protection mechanism signal (Figures 11a and 11b). An example of how the stability protection mechanism can improve execution outcomes can be found in Figure 5, from Section 2.2.2

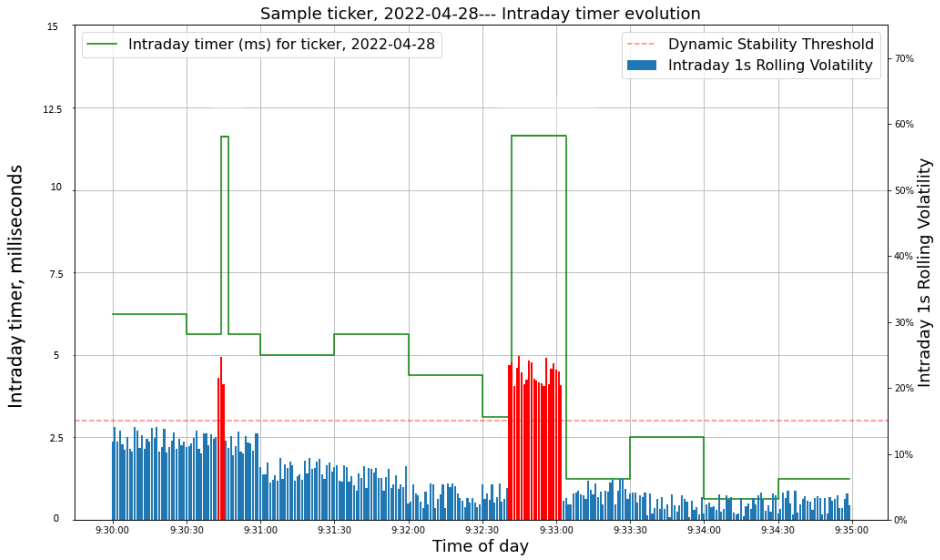
4.3.1 Setting a Stability Threshold

We deem the NBBO unstable if we detect unusually high activity based on the range of NBBO values recorded in the last three seconds. We do this by comparing the range to a threshold for a given ticker and 30 second time interval. The threshold is calculated based on a bisection-quantile method using the previous days' aggregated values (see pseudo-code in 7).

This calculation aims to achieve a certain percent of each day that is



(a) Timer movement given stable NBBO prices.



(b) Timer movement given unstable NBBO prices.

Figure 11: Demonstration of stability protection mechanism.

considered unstable. Currently, we aim to consider 1.0% of the day unstable, i.e. to have on average 1.0% of time in between the open at 9:30 AM and close at 4:00 PM be considered unstable. This can also be interpreted as flagging only the top 1.0% most unstable regions to trigger the 12ms condition. This level of protection was chosen through substantial experimentation.

While it is possible to envision a more sophisticated way of implementing a protection mechanism based on the NBBO midpoint changing, we have found this one to be sufficiently performant. Planned future further exploration of different thresholds includes moving to an intraday moving average or some sort of varying timeline-weighted threshold.

5 Behavior of Dynamic M-ELO on the Market

Our Dynamic M-ELO timer demonstrates an average combined volume-weighted improvement of 31.7% over the current static 10ms. This includes a 20.3% increase in fill rate and a 11.4% decrease in markout when compared to the current M-ELO and tested over a quarter-long period from January 1, 2022 to April 1, 2022. This corresponds with our retraining period: every date that is tested uses a model that was strictly trained on previous days' information. For more details, please revisit Section 4.2.2.

During this period, the average number of timer changes per day across all symbols was 140 out of a possible 780 (see Figure 16). This means the model decided to dynamically change the timer about 18% of the time, leaving it at the previous set value about three fifths of the time.

Here we explain how we measure performance gains over the current M-ELO timer, review our aggregate results, and present a firm-level analysis that confirms Dynamic M-ELO does not suffer from systematic-bias. Additionally, we thoroughly assess the interpretability of our model's decisions through a feature explainability study.

5.1 How are Gains Measured?

Our results represent an average combined volume-weighted improvement of 31.1% when compared to the static 10ms M-ELO *as represented in our simulation*. We assert the need to compare against the 10ms M-ELO simulation rather than the historical for two important reasons.

First, as mentioned in Section 3.2, our simulation M-ELO trading environment tends to slightly undershoot trades when compared to historical values, so in order to provide an homogeneous metric for comparison, we measure its performance relative to the simulated baseline. Erring on the side of prudence, we note that, as a direct consequence, this methodology provides a rather conservative estimate of performance, when compared to how it would be if using actual historical values as benchmark.

Second, the agent-set timer often results in orders trading that never could have traded at the static value of 10ms (e.g., by getting cancelled before the 10ms have elapsed, but after a hypothetical shorter holding period). This introduces a source of bias when comparing the aggregated markout the agent achieves in different time periods against the historical aggregated markout. If there are more trades, which is often the case, there is more aggregate

markout rendering this comparison unreasonable. Instead, we compare the agent-achieved markout with the synthetic markout from shifting the agent-achieved trades to have occurred at the static 10ms holding period.

Given our baseline as outlined above, the respective improvement gains are calculated as follows:

$$\text{FR Improvement (as a \%)} = \frac{\text{FR}_{agent} - \text{FR}_{sim@10}}{\text{FR}_{sim@10}} \tag{4}$$

$$\text{MO Improvement (as a \%)} = \frac{\text{MO}_{agent} - \text{MO}_{synthetic@10}}{\max(|\text{MO}_{synthetic@10}|, |\text{MO}_{agent}|)} \tag{5}$$

and added together to give the total improvement. Note that the markout improvement calculation differs from a simple percent. We revised this formula for metrics which can vary between positive and negative values. We found it to provide more robust and conservative comparisons.

Regardless, as can be inferred from Equations 4 and 5, the gains for both metrics are measured following the usual standard convention, i.e., a positive value corresponds with an improvement with respect to the baseline.

5.2 Aggregate Backtest Results

To provide a better idea of the distribution of markout, fill rate, and combined improvement across all symbols, we have broken down our aggregate +20.3% fill rate, +11.4% markout gain result into quartiles as shown in the box-and-whisker plot in Figure 12 and in the descriptive statistics in Table 1. The box-and-whisker plot clearly shows that the combined results are right-skewed, since there are more positive than negative outliers in both fill rate and markout improvement. This means that, beyond the net positive average across symbols, the positive combined improvement happens more frequently than negative loss per-symbol.

Additionally, we benchmarked our model's performance against both static and random timers. This was done to assure that the positive increase in fill rate and decrease in markout can be attributed to the model and not to merely lowering the timer or having it switch randomly between allowable values.

Likewise, for the static timers, we see that the Dynamic M-ELO model outperforms all but the lowest allowable static timer in terms of combined

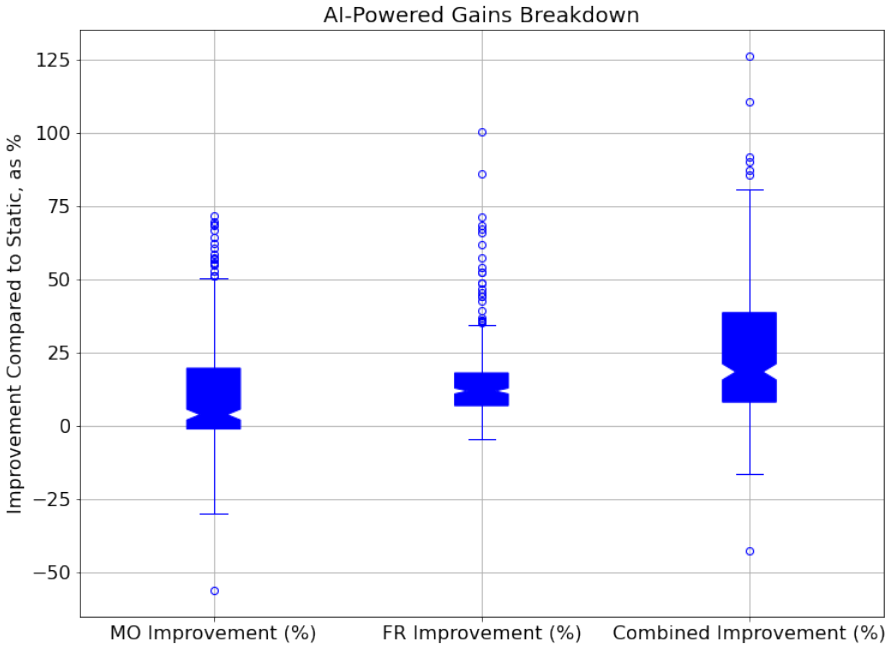


Figure 12: Distribution of fill rate, markout, and combined improvement across tickers. Note that markout and fill rate are treated as independent for their descriptive statistics, so the both column is not just a sum of the other two. For descriptive statistics, see Table 1.

gain, losing only to the .25ms timer by fill rate. However, given the fact that .25ms is the minimum allowed timer value, this actually provides the upperbound for the agent's possible fill rate gain. In terms of markout, Dynamic M-ELO is a clear winner, yielding a substantial net positive markout reduction compared to a static .25ms timer.

5.2.1 Firm-Level Analysis

In order to ensure that our results do not provide an unfair advantage to any specific firm, but rather distribute the gains equitably, we tried to identify patterns and trends that could potentially signify a systematic bias towards specific firms.

	MO	FR	Both
Avg	11.4%	20.3%	31.7%
25^{th} Percentile	-1.1%	6.8%	8.0%
50^{th} Percentile	3.8%	11.8%	18.4%
75^{th} Percentile	19.6%	18.0%	38.6%

Table 1: Distribution of fill rate, markout, and combined improvement across tickers. Note that markout and fill rate are treated as independent for their descriptive statistics, so the both column is not just a sum of the other two. See Figure 12 for a box and whisker plot representation.

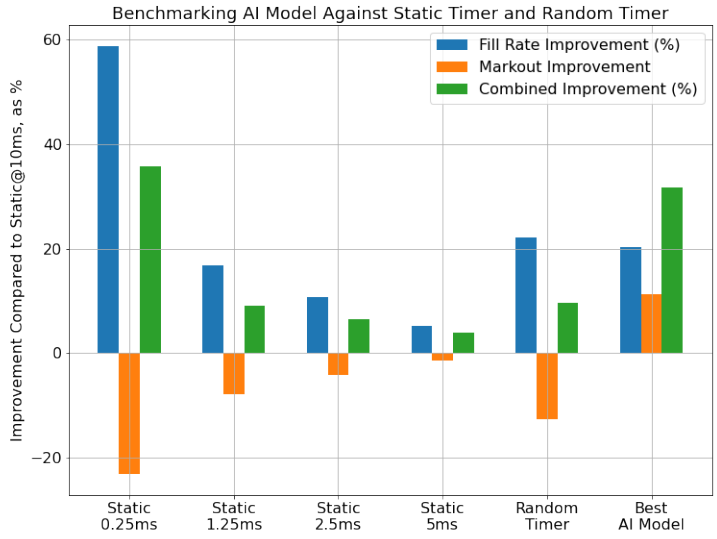


Figure 13: Dynamic M-ELO results compared with relevant static timers and random switching between allowable timer values. All the metrics in this chart are measured using the static version of M-ELO as baseline.

To this end, we performed a detailed analysis on the distribution of AI-driven gains to assess how they particularly impacted each firm that actively participates in M-ELO, regardless of volume. This yielded satisfactory re-

sults: no overall trend emerged, confirming that Dynamic M-ELO will not result in systematic-biased execution towards any one firm.

5.3 Explainability Study

Practical deployment of AI systems is not possible without substantial investigation into how they make their decisions. Indeed, garnering trust for the outcome of the system is wholly dependent on an understanding of how the model works. Here we address any concerns arising from explainability in two ways. First, we present a philosophical discussion of how AI decisions differ from human decisions, and how and why this can be beneficial. Then, we hone in on our model and provide an in-depth analysis of how specific inputs map onto our model's outputted timer change recommendations as well as how the complex interactions between inputs impact the timer duration value.

5.3.1 AI Decision Making

Humans make decisions *ex ante*, meaning they are based on our expectations or predictions of the future. Unfortunately, our predictions of the future are rarely correct and are often clouded by a myriad of obscure, confounding factors. We are limited in three main ways: information overload, biases, and decision fatigue. Information overload is the psychological concept that describes how we cannot leverage all available information due to our limited ability to process it; cognitive and emotional biases covers any deviations from strict rationality, which happen more often then we care to admit; and decision fatigue describes how our fallible brains tire after too much use.

AI systems, on the other hand, do not suffer from information overload or decision fatigue. As for cognitive biases— the likes of hindsight bias— and emotional biases— loss/regret aversion and overconfidence — they can be said to not fall prey to these specific examples as well. A broader discussion of how AI systems can be biased by their training sets, especially when their training sets involve demographic information, is out of the scope of this paper.

Furthermore, AI systems make decisions *a priori*, i.e., strictly based on what they have observed from the past. Throughout the training process, AI systems learn a knowledge base from which to make their decisions. This means that, when trained for one specific task, they can become knowledge

domain experts and execute on that task much more efficiently than humans ever could aim to.

Our Dynamic M-ELO model is one such system. It builds its knowledge base to control the duration of the hold period timer, and thereby becomes the equivalent of a subject matter expert timer setting system that can be tasked to deal with the complexity of incoming information and the cadence of decision making without fatigue— unlike a human operator. To the extent that our model is temporally biased to its retraining set, this is by design: our model performs better by adapting to local conditions. The data it ingests is reliable— coming straight from the Nasdaq core internal messaging system.

5.3.2 Feature Sensitivity and Feature Interaction Study

Here we make a case for the explainability of our model's decisions given this reliable data. As stated in Section 3.2, our model ingests 142 features that capture market dynamic information about the continuous book and recent M-ELO activity, as well as other information associated with how the timer impacts the simulation environment. We performed both a localized feature sensitivity analysis and a global feature interaction study to motivate our model's timer-set decisions.

Our feature sensitivity analysis involved varying each feature's value and seeing how changing it (while holding all other features constant) directly impacted the trained model's timer duration decision. The results of this study indicate that, out of the 142 features, 27 have been found to be directly correlated with higher (slower) timers, while 25 have been found to be inversely correlated. For the sake of interpretability, we share some of these below. For a more comprehensive analysis of our feature explainability study, please see Table in Section 7.1 (Appendix 7.1).

Features which, as they increase, tend to contribute toward slowing (increasing) the holding period include: increases in the standard deviation of NBBO prices, number of unique firms placing sell orders on M-ELO, and the volume-weighted average NBBO spread. These factors are all associated with higher volatility of the underlying, and Dynamic M-ELO rightly reacts by increasing the timer to try to combat the possibility of momentarily high markout trades.

On the other hand, features which, as they increase, tend to contribute to speeding up (reducing) the holding period include an increase in the median and maximum number of shares per trade and the number of resting M-ELO

(a) (b)

Figure 14: Example 30 second time periods resulting in action 1.0 (a), mapping onto a timer decrease of .25ms, and 4.0 (b), mapping onto a timer increase of .5ms.

bids left. These factors are associated with high fill rate, relatively stable periods, and Dynamic M-ELO reacts by decreasing the timer to try to allow as little friction as possible when trading.

Furthermore, we can analyze the interactions between these features and how they contribute to the model's decisions. For example, an increase in the standard deviation of M-ELO buy-side signed markout and a decrease in the median and maximum number of shares per trade was found to be strongly correlated with an increase in timer. Likewise, a decrease in the average NBBO midpoint and an increase in the NBBO price standard deviation, corresponding with a rapid negative change in price, correlate with an increase in timer.

More complicated interactions can be seen in the Shapley value study below, which uses principles from game theory to calculate the marginal contribution of each feature to a model's output [**4**]. Here we present the ordered top seven influential features that, when combined with all of the other 142 features for a given 30 second time period, give our model's output. It is important to note however, that combining the sum total of the relative impact (Shapley) values yields the maximum Q-value from our model's output and not the actual action itself. This is because the model outputs the Q-values of each action for a given 30 second period, which are argmaxxed to give the model's prescribed relative timer change action. Two examples can be seen in Figure 14b.

6 Conclusion

We have proposed a dynamic-timer modification to the Nasdaq Midpoint Extended Life Order (M-ELO) holding period market that, based on our simulation, achieves an increase in fill rate of 20.3% and a decrease in markout of 11.4% compared to the current static 10ms M-ELO timer.

Our Dynamic Timing system leverages AI to evaluate and determine the duration of the M-ELO holding period in relation to everchanging local market conditions. In doing so, it is able to achieve combinations of fill rate and markout results that were previously unreachable by static timers (see Figure 15). We assert that Dynamic M-ELO has shifted the fill rate-markout Pareto frontier towards more favorable trading execution for all parties involved, representing a meaningful innovation.

These overwhelmingly positive results suggest further opportunity to improve market quality with conditionally-attuned products and controls. In an evolving world with more access to data and computational resources than ever before, innovations like Dynamic M-ELO are not only possible and effective, but also represent opportunities to move towards more dynamic market solutions.

7 Appendix

7.1 Further Analysis

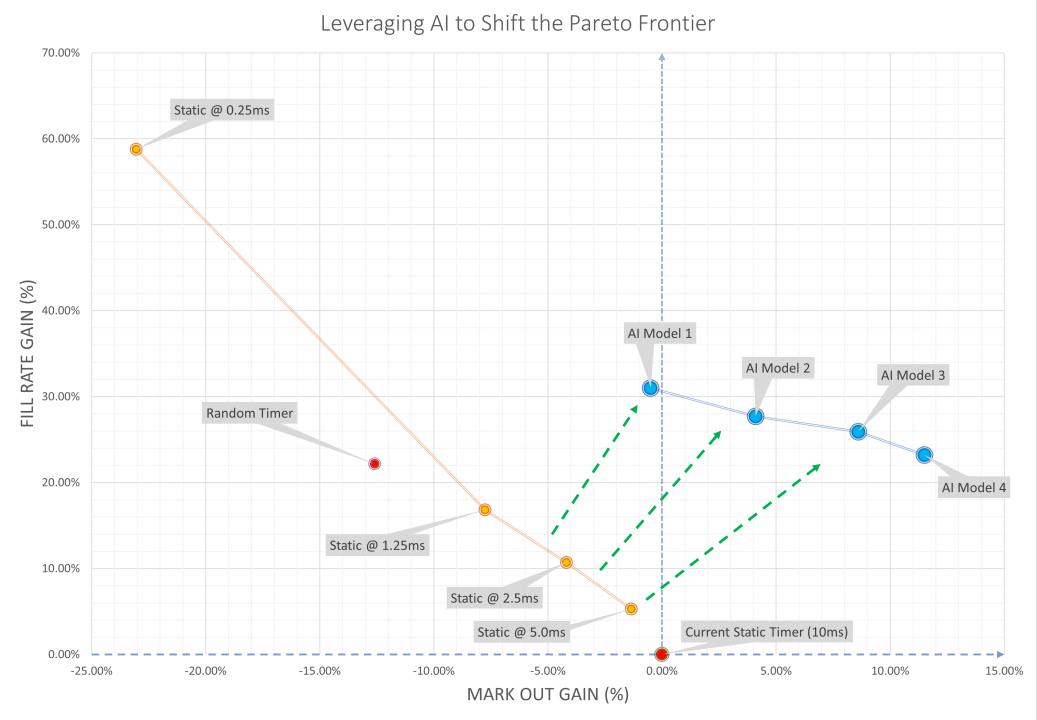


Figure 15: Demonstrated ability of AI to shift Pareto frontier of allowable fill rate and markout values. In particular, the AI models, indicated in blue, significantly outperform an hypothetical static version of M-ELO with timers lower than 10ms, indicated in yellow. The green dashed arrows signify the simultaneous gains driven by the AI system.

Table 2: More comprehensive overview of local feature explainability study.

Relationship with Holding Timer	Feature
Strongly Directly Correlated	NBBO price stdev
	# firms placing M-ELO sell orders
	Volume-weighted avg NBBO spread
Somewhat Directly Correlated	Max NBBO spread
	NBBO price level skewness & kurtosis
	# of resting M-ELO asks
	M-ELO sell trade qty stdev
	# unique firms on M-ELO
	M-ELO buy signed markout, stdev and max
	Short-term kurtosis of NBBO trade qty
Somewhat Inversely Correlated	# of resting M-ELO bids
	Shares/trade average
	Short-term NBBO trade qty skewness
	NBBO midpoint average
	# unique firms placing M-ELO buys
	Proportion of buys in incoming M-ELO orders
Strongly Inversely Correlated	# of shares/trade median and max
	Timer (tendency towards lower timers when possible)

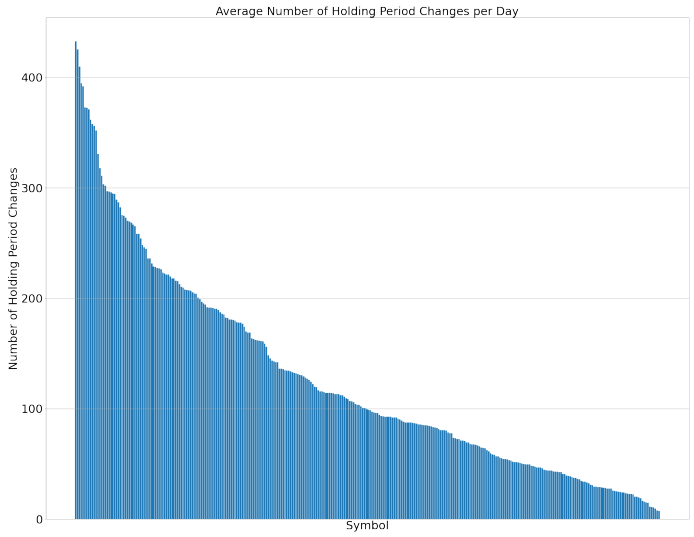


Figure 16: Average timer changes per day by symbol. The maximum number of changes per day is 780, corresponding to the amount of 30 second periods in a trading day.

7.2 Algorithms

7.2.1 D3QN: Modified DDQN Algorithm

Algorithm 1 D3QN: Dynamic M-ELO-DDQN

for date **d** in {date, end date} **do**
———————— Begin Agent Experience Collection ————————
  for symbol **s** in {symbols} **do**
    for iteration **i** in $780 \times \frac{\text{30 sec periods}}{\text{1 day}}$ **do**
      Gather and store (state, action, reward, next state) experiences
    end for
  end for
———————— End Agent Experience Collection ————————
———————— Begin Agent Training Loop ————————
  for m in number of symbols **do**
    Sample a batch of (state, action, reward, next state) experiences
    for experience **e** in {batch} **do**
      Generate target Q-values
    end for
    Calculate loss L
    Update main model parameters via gradient descent
  end for
  Update target weights to be evolving weighted linear combination of main model and target model weights
———————— End Agent Training Loop ————————
end for

The training process we implemented for Dynamic M-ELO actually differs from the vanilla DDQN algorithm in several ways [**2**]. First, instead of adding all episodes to the agent's memory, we only add specific episodes which have a non-zero reward, i.e. that return a non-zero fill rate and/or markout value. Additionally, instead of using a single memory buffer, we use a multi-buffer, separating experiences by each component of the reward function to ensure balanced training. Finally, instead of sampling from the buffer at the end of each episode only once, we sample from the buffer a number of times corresponding to the product of the number of ticker times and the number of steps in each training iteration (at the end of every symbol-day).

For further details, please see annotated pseudo-code above, which is broken up into two distinct phases— agent experience collection and agent training loop— for ease of understanding.

7.2.2 Stability Coverage Algorithm

Algorithm 2 Quantile-Bisection Stability Coverage Method

for symbol **s** in $\{\text{symbols}\}$ **do**
    Decide on optimal coverage c^*
    Initialize naive price range lower bound l and upper bound u
    Compute midpoint m between l and u
    Calculate coverage c for m
    while (l, u) do not offer optimal coverage c^* **do**
        if $c < c^*$ **then**
            Replace the value of u with m
        else
            Replace the value of l with m
        end if
        Update m to be the midpoint of the new l and u
        Update c to be the coverage of the new midpoint m
    end while
    Return m as optimal threshold
end for

References

[1] Mnih, V., Kavukcuoglu, K., Silver, D. et al. *Human-level control through deep reinforcement learning*. Nature 518, 529-533 (2015). 10.1038/nature14236.

[2] Hasselt, H. *Deep Reinforcement Learning with Q-Learning*. Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16) 23, 2094-2100 (2016). 10.5555/3016100.3016191

[3] Sutton, R.S. and Barto, A.G. *Reinforcement Learning: An Introduction*. MIT Press, 2018. ISBN: 9780262193986.

[4] Lundberg, S.M. and Lee, S. *A Unified Approach to Interpreting Model Predictions*. Proceedings of the 31st Neural Information Processing Systems Conferece (NeurIPS-17).

[5] Lehalle, C. and Laruelle, S. *Market Microstructure in Practice*. World Scientific Publishing Co. Pte. Ltd., 2018. ISBN: 9789813231122.

[6] Hasbrouck, J. *Empirical Market Microstructure*. Oxford University Press, 2007. ISBN: 9780195301649.

[7] O'Hara, M. *Market Microstructure Theory*. Blackwell Publishing, 1995. ISBN: 9781557864437.