INTAGS: Interactive Agent-Guided Simulation

Song Wei*
Georgia Institute of Technology
song.wei@gatech.edu

Andrea Coletta **Svitlana Vyetrenko** **Tucker Balch**
J.P. Morgan AI Research
{andrea.coletta, svitlana.s.vyetrenko, tucker.balch}@jpmchase.com

Abstract

The development of realistic agent-based simulator (ABS) remains a challenging task, mainly due to the sequential and dynamic nature of such a multi-agent system (MAS). To fill this gap, this work proposes a metric to distinguish between real and synthetic multi-agent systems; The metric evaluation depends on the live interaction between the *experimental (Exp) autonomous agent* and *background (BG) agent(s)*, explicitly accounting for the systems' sequential and dynamic nature. Specifically, we propose to characterize the system/environment by studying the effect of a sequence of BG agents' responses to the environment state evolution, and we take such effects' differences as MAS distance metric; The effect estimation is cast as a causal inference problem since the environment evolution is confounded with the previous environment state. Importantly, we propose the Interactive Agent-Guided Simulation (INTAGS) framework to build a realistic simulator by optimizing over this novel metric. To adapt to any environment with interactive sequential decision making agents, INTAGS formulates the simulator as a stochastic policy in reinforcement learning. Moreover, INTAGS utilizes the policy gradient update to bypass differentiating the proposed metric such that it can support non-differentiable operations of multi-agent environments. Through extensive experiments, we demonstrate the effectiveness of INTAGS on an equity stock market simulation example.

1 Introduction

In various applications involving multi-agent system (MAS), training or testing an autonomous agent (*experimental agent*) that analyzes the current state of the system and responds promptly is a crucial task. Such a task oftentimes relies on live interaction with the environment, be it real or synthetic, to account for the responses of other interactive agents in the environment (*background agents*) to the experimental agent's action. In particular, in the financial domain, algorithmic trading (AT) should have access to a live environment with other market participants to construct and test trading strategies [Pardo, 2011, Balch et al., 2019]. The live interaction overcomes the limitation of traditional offline methods using historical replays, which fails to capture the dynamic nature of the environment and its reactivity [Coletta et al., 2023a]. However, interaction with the real environment is typically expensive, unsafe, and rarely feasible for research purposes, especially in the automotive or financial domain. As a result, the most prevalent approach to develop an experimental agent is through the interaction with a generative model, i.e., *agent-based simulator (ABS)*, aiming to emulate the real environment, i.e., background agents' behaviors.

*This work was done while Song Wei was interning at J.P. Morgan AI Research.

NeurIPS 2023 Workshop on Synthetic Data for Empowering ML Research.

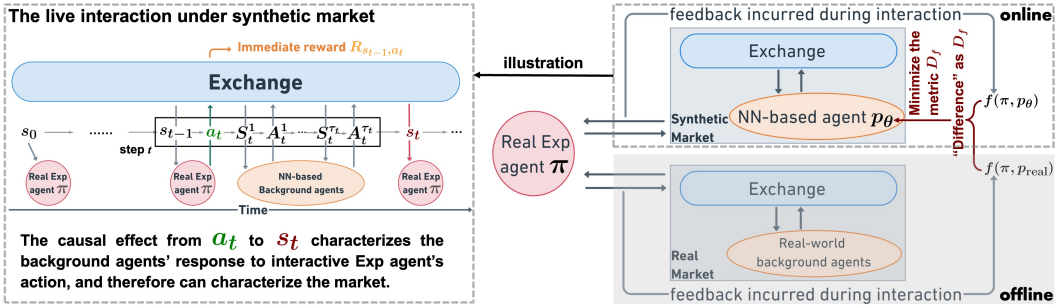

Figure 1: **Illustration in the market simulation application, where the environment consists of background (BG) agents/traders and the exchange. We show how the experimental (Exp) agent (following policy π) interact with the BG agents (following policy p_θ in simulation, or $p_{\texttt{real}}$ in reality) in the environment (left) and the INTAGS training framework (right). As in real MAS, within the t-th step, the BG agents act τ_t times, between consecutive actions of the Exp agent.**

Classic ABS are mostly parametric, and based on explicit rules that the agents need to follow. For instance, under the context of traffic simulation, there is parametric ABS built upon strict rules/laws that the drivers (i.e., agents) must obey [Quinlan et al., 2010]. However, such rules do not account for irrational or irregular human actions, and they cannot be easily summarized in complex environments, such as financial markets. This complexity motivates the need for advanced machine learning techniques to train non-parametric ABS which can capture those rules from real-world data. Focusing on the financial domain, conditional Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] are among the most popular approach for market simulation (see Appendix A for related work). Specifically, the state-of-the-art (SOTA) approach is a conditional Wasserstein GAN (cWGAN) trained on market replay, which was recently proposed by Li et al. [2020], Coletta et al. [2021, 2022]. To the best of our understanding, the success of cWGAN as SOTA in market simulation comes from its adaption to this specific application: By conditioning on the market input, cWGAN captures historical dependence and attempts to adapt to the sequential nature of trading activities (or rather, the interactions within the MAS). Additionally, cWGAN uses Wasserstein distance as the discriminator (or metric) to allow synthetic data that is unseen in the historical real market, potentially improving its generalization ability to account for complex market dynamics.

However, the aforementioned adaptions are still not able to completely capture the whole market dynamics. *First, cWGAN only considers sequential trading activities/interactions locally*. Specifically, cWGAN cuts the sequence of state-action pairs of the background agents into independent pieces. That is to say, cWGAN considers the local dependencies (i.e., the pairs (S, A); please refer to Section 2 for rigorous definitions of notations) of the whole sequential structure as shown in the left panel of Figure 1. Thus, cWGAN-based ABS is merely a generative model that completely ignores the dependency/progression of those sequential pairs. *The second pitfall of cWGAN is its poor generalization ability, which comes from the offline training on replays*: although cWGAN is able to generate (output) unseen states it may not respond realistically when an external agent acts aggressively, creating unseen (input) market states [Coletta et al., 2023a]. During training, cWGAN is conditioned mostly on historical states, with unknown performance when conditioned on states with different data distributions. The cWGAN's response (which shapes the simulated market) to those unseen states may become unrealistic. Therefore, it is crucial to introduce external experimental agents and consider the live/online interaction between experimental and background agents during the training. To the knowledge of the author, even though there are many recent efforts on GAN-based market simulators (to be reviewed in Appendix A), they did not go beyond either the classic local metric or offline training on replays, and a satisfying quantitative metric that adapts to the sequential, complex, and dynamic natures of the trading market is largely missing [Bouchaud et al., 2018, Vyetrenko et al., 2020]. As a result, it remains a challenging task to train ABS that generates realistic market data.

In this work, we address the aforementioned issues by proposing a distance metric between real and synthetic multi-agent systems that accounts for the sequential and dynamic nature of the MAS. Based on this novel metric, we develop an online training framework, named Interactive Agent-Guided Simulation (INTAGS). INTAGS only requires offline interactions/replays of the real-world experimental agent with the real environment, and thus will not interfere with the real-world production system; The framework is online because the objective function, or rather the proposed distance metric, is

obtained through the online/live interaction of the real-world experimental agent with the synthetic environment. Specifically, we propose to characterize the underlying environment by the causal effect [Rubin, 1974] from the experimental agent's action to environment state evolution, since such an effect is a result of a sequence of background agents' (which uniquely characterize the environment) responses to the experimental agent's action; We take the difference between the causal effects under real and synthetic environments as our MAS distance metric. Inspired by SeqGAN [Yu et al., 2017], we formulate the generator network as a stochastic policy to account for the sequential nature of the MAS. Moreover, by leveraging the Policy Gradient Theorem [Sutton et al., 1999], INTAGS minimizes our proposed metric without differentiating the metric, since its evaluation typically involves non-differentiable operations, such as order deletion in our market simulation application. We conduct extensive experiments to study the market simulation application, showing our INTAGS can generate much more realistic market data compared with the cWGAN-based simulator. This improvement could, in turn, facilitate the use of sample inefficient reinforcement learning (RL) approaches for trading strategy construction, reducing "time-period bias" and the unrealistic reactivity of the existing cWGAN simulator. Although our experiments specifically focus on simulating financial markets, our proposed INTAGS can be applied in all environments with interacting agents, including language generation [Bai et al., 2022] and traffic simulation [Suo et al., 2021].

2 Problem Set-up

We formulate the agents' interactions as a Markov Decision Process, in which the experimental (Exp) agent interacts with the environment (Env), which is characterized by the background (BG) agent(s), to adapt its strategy based on the dynamic feedback from the Env. The Exp agent can be rule-based, or trained by online reinforcement learning, enabling the Exp agent to make informed decisions sequentially in response to changing Env conditions.

Experimental agent. Denote the state space by $\tilde{\mathcal{S}}$, action space by $\tilde{\mathcal{A}}$, and a reward function by R. At each discrete time step $t \in \{1, \ldots, T\}$, where T is a finite time horizon, the Exp agent observes the state $s_{t-1} \in \tilde{\mathcal{S}}$, and takes action $a_t \in \tilde{\mathcal{A}}$ according to its policy π; this action can incur an immediate reward $R(s_{t-1}, a_t)$ and push the environment to the next state s_t as the BG agents respond to a_t.

Background agent. To distinguish the (synthetic) BG agent from the Exp agent, we denote the input to the BG agent as $S \in \mathcal{S}$, and output action as $A \in \mathcal{A}$; Here, $A = G_\theta(z|S),\ z \sim N(\mathbf{0}, \mathbf{1})$, where we parameterize G_θ as a NN with network parameter θ; Indeed, the BG agent follows the underlying stochastic policy:

$$A \sim p_\theta(\cdot|S). \tag{1}$$

The goal is to train a NN to emulate the real BG agents, ensuring that the simulated environment with a NN-based BG agent closely matches reality. See Figure 1 for an illustration of the interaction between the Exp agent and BG agent(s).

Terminologies. The NN-based BG agent can be viewed as "one agent representing all real BG agents in the world"; Following idea of the world model [Schmidhuber, 2015] and world agent [Coletta et al., 2021], we name this NN-based BG agent by *world BG agent* and the simulated/synthetic interactive environment with world BG agent by *world Env*. The interactive environment with real BG agents is referred to as *real Env*. In what follows, we will use the terms "simulator/generator", "world BG agent (policy)", and"world Env" interchangeably; Similarly, we use the terms "reality", "real BG agents", and "real Env" interchangeably.

Data. We use an ordered list $\mathcal{T}$ to denote the collection of states and actions when the Exp agent π interacts with the Env, and we name the collected data during (or after) this interaction as *rollout*. Specifically, take the world Env as an example (which is illustrated in Figure 1), the *partial rollout* that ends at time step $t \in \{1, \ldots, T\}$ is:

$$\mathcal{T}_{\pi,\theta}^{1:t} = \{s_0, \ldots, s_{t-1}, a_t, (S_t^1, A_t^1), \ldots, (S_t^{\tau_t}, A_t^{\tau_t})\}.$$

In particular, the prefix that determines the last sequential action $A_t^{\tau_t}$ of the BG agent is

$$\tilde{\mathcal{T}}_{\pi,\theta}^{1:t} = \left\{s_0, \ldots, s_{t-1}, a_t, (S_t^1, A_t^1), \ldots, (S_t^{\tau_t - 1}, A_t^{\tau_t - 1}), S_t^{\tau_t}\right\}.$$

3 Proposed Metric

In this section, we formally define the unique characteristic of the Env, through a *feedback* f, after the interaction of a fixed Exp agent π with the corresponding Env. Thus, the generator training can be done by minimizing the chosen feedback's difference between the world Env and the real Env.

3.1 Formulation

For Exp agent with policy π, the feedback can be obtained through its interaction with the Env, be it real or synthetic. Given the complete rollout $\mathcal{T}_{\pi,\theta}^{1:T}$ under the world Env, the feedback is denoted by $f(\mathcal{T}_{\pi,\theta}^{1:T})$, which can be understood as a realization of random variable $f(\pi, p_\theta)$ shown in Figure 1, where the randomness comes from the stochastic policy p_θ (1). Similarly, we use subscript `real` to denote the feedback under real Env, i.e., $f(\mathcal{T}_{\pi,\texttt{real}}^{1:T})$, which can be viewed as a realization of random variable $f(\pi, p_{\texttt{real}})$.

Given f, our proposed market distance metric, which is highlighted in red in Figure 1 and will be used to train the world BG agent, is defined as:

$$D_f(\theta) = \hat{d}\Big(\big\{f(\mathcal{T}_{i,\pi,\theta}^{1:T}),\ i = 1, \ldots, N\big\}, \big\{f(\mathcal{T}_{j,\pi,\texttt{real}}^{1:T}),\ j = 1, \ldots, N'\big\}\Big), \tag{2}$$

where $\mathcal{T}_{i,\pi,\theta}^{1:T}$'s and $\mathcal{T}_{j,\pi,\texttt{real}}^{1:T}$'s are the complete rollouts under the world Env and real Env, respectively. Here, $\hat{d}$ represents an empirical estimate of a distance metric between probability distributions. A popular example is Maximum Mean Discrepancy (MMD) [Gretton et al., 2012]. Please see further details, including a graphical illustration of the evaluation of D_f (see Figure 5) and the expression of an unbiased estimate of MMD in Appendix B.2.

3.2 Numerical evidence

We use numerical evidence to select feedback f and empirical probability distance metric $\hat{d}$. The experiments are conducted under the stock market simulation setting and please find complete details of the configurations in Appendix D.1. We start with the most straightforward candidate feedback — the end-of-rollout cumulative reward of the Exp agent, denoted by `EpisodeReward` — and plot its empirical distribution under world and real markets in the first row of Figure 2. We can observe that: for a sufficiently large number of rollouts ($N = 100$, the last column in the first row of Figure 2), both visual evidence and quantitative metric MMD support `EpisodeReward`'s effectiveness in differentiating markets. However, taking $N = 100$ incurs unreasonably high computational cost — as shown in Algorithm 1, N MC rollouts (step 7) are needed $T \times b$ times for each iteration in generator training (see further discussion on complexity at the end of this paper). Given the constraint on N, with a limited number of rollouts, it is difficult to differentiate two markets (as shown in the first two columns in the first row of Figure 2), rendering `EpisodeReward` undesirable for training generator.

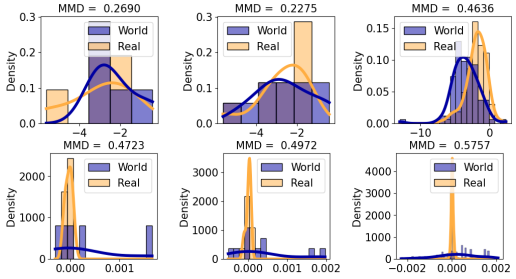

Figure 2: **The distribution of feedback (top: `Episode Reward`; bottom: `Mkt2NextReturn`) under multiple rollouts (from left to right: $5, 10, 100$) under real (orange) and world (blue) markets. The ideal feedback should have different empirical distributions under the real and world markets with few rollouts. Thus, `Mkt2NextReturn` is better, which can differentiate markets with 5 rollouts (bottom left panel).**

Based on the above analysis, we propose two "axioms" that f must satisfy: **(Ax1)** Separability — f must be able to differentiate markets in the sense that f's under different markets should be very different; **(Ax2)** Fast convergence — f should have its distribution quickly "converge" to the final state so that only a few rollouts are needed during the generator training. Indeed, `EpisodeReward` ignores the market dynamics encoded in the time series data collected during the interaction (just as the classic local metric does), and this might explain why the "convergence" w.r.t. number of rollouts is not fast enough, resulting in inefficient data usage and information loss. Since the main difference between the real and world markets comes from the BG agents, the feedback should depend on the BG agents. As illustrated in Figure 1, the effect from AT agent action a_t to next state s_t reflects how BG agents respond to a_t, and thus the resulting feedback can characterize the BG agents (and thus the market). To be more precise, we consider the action "placing market order" since market orders are more frequently seen than limit orders; The stylized fact we consider as the next-step state is market return, which is better (in the sense of the aforementioned 2 axioms) than other candidates including spread, imbalance, etc.; For brevity, we name this feedback as `Mkt2NextReturn`.

Causal effect as f. The estimation of `Mkt2NextReturn` can be cast as a causal/treatment effect estimation problem [Rubin, 1974], as s_{t-1} acts as a common cause (or rather, confounder) of both

action a_t (i.e., treatment) and next state s_t (i.e., observed outcome). To adjust for confounding, we apply an inverse probability weighted estimator for `Mkt2NextReturn`. In the second row of Figure 2, we report the empirical distribution of estimated `Mkt2NextReturn` against increasing rollout number, from which we can observe that with only 5 rollouts the empirical distributions are very different from each other (also verified by MMD) and they are fairly close to the "convergence state" at 100 rollouts. Thus, we choose `Mkt2NextReturn` as f that can differentiate markets with a small number of rollouts.

To train generator by minimizing D_f (2), $\hat{d}$ should also be properly chosen such that D_f meets (Ax1) and (Ax2). To ensure that the good separation shown in the second row of Figure 2 is not a result of certain random seeds, we perform bootstrap to quantify the uncertainty: For rollout number $N \in \{2, 3, 5, 7, 10, 20, 30, 40, 50\}$, we select N rollouts from 200 rollouts and repeat this procedure 50 times to obtain 50 MMDs; We plot the mean and 5% - 95% envelope, i.e., 90% bootstrap confidence interval (CI), trajectory of those MMDs against N when the two markets are different (or identical) in Figure 3. Additionally, we report result for `EpisodeReward` as f for comparison. We can observe that MMDs under the same market (red) are significantly smaller than that under different markets (blue) when N exceeds 5, verifying D_f's effectiveness when choosing `Mkt2NextReturn` as f and MMD as $\hat{d}$ to differentiate markets. On the contrary, such a pattern can only be observed for large enough N (around 30) when we consider `EpisodeReward` as f.

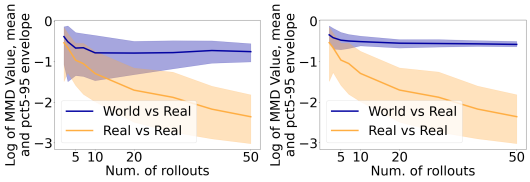

Figure 3: **The mean and 90% CI trajectory of our metric: $\hat{d}$ is MMD estimator, and f is `EpisodeReward` (left) or `Mkt2NextReturn` (right). The ideal metric should be large (or small) under different (or same) markets with few rollouts, reaffirming our finding from Figure 2 that `Mkt2NextReturn` is a better option.**

4 Generator Training

Formulation. Here, we introduce INTAGS that trains the world BG agent by minimizing interactive agent-based distance metric D_f (2). However, the major challenge is that the non-differentiable D_f renders commonly seen empirical methods, e.g., back-propagation, for gradient descent (GD) infeasible. To handle this issue, one popular approach is proposed in SeqGAN work, which reformulated the generator as a stochastic RL policy and performed Policy Gradient update [Sutton et al., 1999]. Fortunately, INTAGS's problem formulation aligns with SeqGAN's in the sense that the world BG agent generates actions sequentially, and the interactive agent-based metric evaluation occurs only at the end of the interaction[2]. Thus, the simulator training is formulated as:

$$\theta = \arg\min_{\theta} \mathcal{L}(\theta) = \sum_{A \in \mathcal{A}} p_\theta(A|S_1^{\tau_1}) Q_f(\tilde{\mathcal{T}}_{\pi,\theta}^{1:1}, A),$$

where the input to world BG agent $S_1^{\tau_1}$ is the last element in the rollout $\tilde{\mathcal{T}}_\theta^{1:1}$, and Q_f is the state-action value function, i.e., the cumulative expected cost conditioned on start state $\tilde{\mathcal{T}}_\theta^{1:1}$ and action A following the stochastic policy p_θ (1). The value function Q_f is chosen to be our metric D_f (2) with complete rollouts, assuming all intermediate costs are zeros. To be precise, at the intermediate stage of the interaction (say at t-th step), Q_f can be estimated through N Monte Carlo (MC) simulations to finish the interaction, referred to as *MC rollouts*:

$$Q_f(\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}, A) = \hat{d}\left(f(\boldsymbol{\mathcal{T}}_{\pi,\theta}^{\mathrm{MC}}), f(\boldsymbol{\mathcal{T}}_{\pi,\texttt{real}})\right),$$

where $f(\boldsymbol{\mathcal{T}}_{\pi,\theta}^{\mathrm{MC}})$ is the collection of feedbacks from N MC rollouts with prefix $\tilde{\mathcal{T}}_{\pi,\theta}^{1:t} \cup \{A\}$, i.e.,

$$f(\boldsymbol{\mathcal{T}}_{\pi,\theta}^{\mathrm{MC}}) = \left\{ f\big(\mathcal{T}_{i,\pi,\theta}^{1:T,\mathrm{MC}}\big) : \tilde{\mathcal{T}}_{\pi,\theta}^{1:t} \cup \{A\} \subset \mathcal{T}_{i,\pi,\theta}^{1:T,\mathrm{MC}}, \quad i = 1, \ldots, N \right\}, \tag{3}$$

and $f(\boldsymbol{\mathcal{T}}_{\pi,\texttt{real}})$ is the collection of feedbacks from N' complete rollouts under the real Env, i.e.,

$$f(\boldsymbol{\mathcal{T}}_{\pi,\texttt{real}}) = \left\{ f\big(\mathcal{T}_{j,\pi,\texttt{real}}^{1:T}\big),\ j = 1, \ldots, N' \right\}. \tag{4}$$

[2]Evaluating f and D_f using partial rollout is feasible, but will lead to insufficient data for the feedback estimation and metric evaluation, resulting in a less favorable metric that cannot differentiate markets.

The Policy Gradient Theorem gives a closed-form expression of the gradient of the objective function $\mathcal{L}$ with respect to (w.r.t.) parameter θ without differentiating interactive agent-based Q_f, i.e.,

$$\nabla_\theta \mathcal{L}(\theta) = \sum_{t=1}^{T} \mathbb{E}_{\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}} \left[\sum_{A_t \in \mathcal{A}} \nabla_\theta p_\theta \left(A_t | S_t^{\tau_t} \right) \cdot Q_f \left(\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}, A_t \right) \right],$$

where $S_t^{\tau_t}$ is the last element of $\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}$. Next, the likelihood-ratio technique [Glynn, 1990] is invoked to obtain an unbiased empirical estimate of the gradient, i.e.,

$$\nabla_\theta \mathcal{L}(\theta) \simeq \sum_{t=1}^{T} \mathbb{E}_{A_t \sim p_\theta(\cdot | S_t^{\tau_t})} \left[\nabla_\theta \log p_\theta \left(A_t | S_t^{\tau_t} \right) \cdot Q_f \left(\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}, A_t \right) \right],$$

where $\tilde{\mathcal{T}}_\theta^{1:1} \subset \cdots \subset \tilde{\mathcal{T}}_{\pi,\theta}^{1:T}$ all come from a complete rollout $\mathcal{T}_{\pi,\theta}^{1:T}$ under world Env p_θ. The expectation in the last line of the above equation can be approximated via sample average. See Appendix B for further details of the training of INTAGS, including practical implementation in Algorithm 1 and a graphical illustration of how to back-propagate to obtain the gradient in Figure 6.

Performance improvement. Lastly, we use numerical evidence to demonstrate the effectiveness of INTAGS. Specifically, we name the application of INTAGS under the context of market simulation as Algorithmic Trading-guided Market Simulator (ATMS). We evaluate the simulator performance with not only our novel metric but also visualizing certain stylized fact time series Bouchaud et al. [2018], Li et al. [2020]. To demonstrate the effectiveness of our ATMS, we visualize the stylized fact volume at first 10-levels from 10AM to 11AM over 12 independent trials in Figure 4. We can observe that, during the ATMS training, volume at first 10-levels is "increasingly similar" (in terms of magnitude and matching demand-supply) to reality while our AT-based metric is minimized.

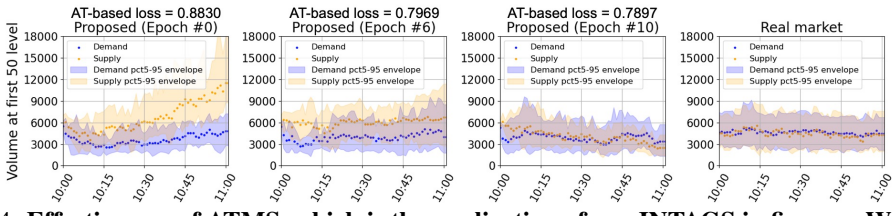

Figure 4: **Effectiveness of ATMS, which is the application of our INTAGS in finance. We report volume at first 10-levels. The ATMS can generate market data that is "increasingly" similar to reality while minimizing our AT-based metric.**

5 Conclusion and Discussion

This work proposes a MAS distance metric, on which we build INTAGS online training framework. The effectiveness is demonstrated by the market simulation application, reaffirming our claim that incorporating the agents' live interactions is of vital importance for simulating environments with interactive agents. Our experiments demonstrate the potential of INTAGS in real production; As illustrated in Figure 1, INTAGS only needs a fixed collection of real feedback $f(\boldsymbol{\mathcal{T}}_{\pi,\texttt{real}})$ (4), i.e., historical reply. Thus, the deployment of our INTAGS, such as ATMS in market simulation, does not interfere the real production. However, since INTAGS's deployment requires the Exp agent to take real-world strategy, which typically targets very complex tasks on a large time horizon T, one potential limitation is the computational complexity: Each update of parameter θ in Algorithm 1 requires $\mathcal{O}(NbT)$ interactions, resulting in $\mathcal{O}(NbT^2)$ steps. Luckily, the complexity can be reduced to $\mathcal{O}(NbT_0T)$ by approximating the gradient as:

$$\nabla_\theta \mathcal{L} \simeq \sum_{t=1}^{\boldsymbol{T_0}} \mathbb{E}_{x \sim p_\theta\left(\cdot | S_t^{\tau_t}\right)} \left[\nabla_\theta \log p_\theta \left(A_t | S_t^{\tau_t} \right) Q_f \left(\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}, A_t \right) \right].$$

This is because T_0 (and b) determines how many BG agent actions are sampled and penalized, and thus can be treated as batch-size in stochastic GD; This makes T_0 a tunable hyperparameter; By carefully tuning T_0 (and b; see the hyperparameter selection in Appendix D.1), we can finish the INTAGS training within a reasonable time frame (around 2 hours) using Amazon Web Services (r6i.24xlarge, 96 CPUs, 768 GiB memory). In practice, after proper selection of N[3], b, T_0, the complexity is merely linear w.r.t. the horizon T, which is determined by the real tasks.

[3] N controls how well the estimated metric Q_f is, and it has already been tuned by experiments — We want to find the smallest N to meet Ax2 and guarantee an effective metric (i.e., the estimated Q_f can differentiate simulated and real Envs).

Disclaimer

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

Robert Almgren and Neil Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3: 5–40, 2001.

Selim Amrouni, Aymeric Moulin, Jared Vann, Svitlana Vyetrenko, Tucker Balch, and Manuela Veloso. Abides-gym: gym environments for multi-agent discrete event simulation and application to financial markets. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

Tucker Hybinette Balch, Mahmoud Mahfouz, Joshua Lockhart, Maria Hybinette, and David Byrd. How to evaluate trading strategies: Single agent market replay or multiple agent interactive simulation? *arXiv preprint arXiv:1906.12010*, 2019.

Feryal Behbahani, Kyriacos Shiarlis, Xi Chen, Vitaly Kurin, Sudhanshu Kasewa, Ciprian Stirbu, Joao Gomes, Supratik Paul, Frans A Oliehoek, Joao Messias, et al. Learning from demonstration in the wild. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 775–781. IEEE, 2019.

Raunak P Bhattacharyya, Derek J Phillips, Changliu Liu, Jayesh K Gupta, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 789–795. IEEE, 2019.

Jean-Philippe Bouchaud, Julius Bonart, Jonathan Donier, and Martin Gould. *Trades, quotes and prices: financial markets under the microscope*. Cambridge University Press, 2018.

David Byrd, Maria Hybinette, and Tucker Hybinette Balch. Abides: Towards high-fidelity multi-agent market simulation. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 11–22, 2020.

Andrea Coletta, Matteo Prata, Michele Conti, Emanuele Mercanti, Novella Bartolini, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. Towards realistic market simulations: a generative adversarial networks approach. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

Andrea Coletta, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. Learning to simulate realistic limit order book markets from data as a world agent. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 428–436, 2022.

Andrea Coletta, Joseph Jerome, Rahul Savani, and Svitlana Vyetrenko. Conditional generators for limit order book environments: Explainability, challenges, and robustness. *arXiv preprint arXiv:2306.12806*, 2023a.

Andrea Coletta, Svitlana Vyetrenko, and Tucker Balch. K-shap: Policy clustering algorithm for anonymous state-action pairs. *arXiv preprint arXiv:2302.11996*, 2023b.

Rama Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative finance*, 1(2):223, 2001.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990.

Alejandro Gomez-Alanis, Jose A Gonzalez-Lopez, and Antonio M Peinado. A kernel density estimation based loss function and its application to asv-spoofing detection. *IEEE Access*, 8: 108530–108543, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Martin D Gould, Mason A Porter, Stacy Williams, Mark McDonald, Daniel J Fenn, and Sam D Howison. Limit order books. *Quantitative Finance*, 13(11):1709–1742, 2013.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012.

Ben Hambly, Renyuan Xu, and Huining Yang. Recent advances in reinforcement learning in finance. *arXiv preprint arXiv:2112.04553*, 2021.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Daniel G Horvitz and Donovan J Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

Michaël Karpe, Jin Fang, Zhongyao Ma, and Chen Wang. Multi-agent reinforcement learning in a realistic limit order book market simulation. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–7, 2020.

Adriano Koshiyama, Nick Firoozye, and Philip Treleaven. Generative adversarial networks for financial trading strategies fine-tuning and combination. *Quantitative Finance*, 21(5):797–813, 2021.

Ashutosh Kumar, Arijit Biswas, and Subhajit Sanyal. ecommercegan: A generative adversarial network for e-commerce. *arXiv preprint arXiv:1801.03244*, 2018.

Chia-Hsuan Kuo, Chiao-Ting Chen, Sin-Jing Lin, and Szu-Hao Huang. Improving generalization in reinforcement learning–based trading by using a generative adversarial market model. *IEEE Access*, 9:50738–50754, 2021.

Junyi Li, Xintong Wang, Yaoyang Lin, Arunesh Sinha, and Michael Wellman. Generating realistic stock market order streams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 727–734, 2020.

Charles M Macal and Michael J North. Tutorial on agent-based modeling and simulation. In *Proceedings of the Winter Simulation Conference, 2005.*, pages 14–pp. IEEE, 2005.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd international conference on Machine learning*, pages 673–680, 2006.

Robert Pardo. *The evaluation and optimization of trading strategies*, volume 314. John Wiley & Sons, 2011.

Yagna Patel. Optimizing market making using multi-agent reinforcement learning. *arXiv preprint arXiv:1812.10252*, 2018.

Michael Quinlan, Tsz-Chiu Au, Jesse Zhu, Nicolae Stiurca, and Peter Stone. Bringing simulation to life: A mixed reality autonomous intersection. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6083–6088. IEEE, 2010.

Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. In *International Conference on Learning Representations*, 2019.

Muhammad Sarmad, Hyunjoo Jenny Lee, and Young Min Kim. Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5898–5907, 2019.

Jürgen Schmidhuber. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *arXiv preprint arXiv:1511.09249*, 2015.

Dino Sejdinovic, Bharath Sriperumbudur, Arthur Gretton, and Kenji Fukumizu. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The annals of statistics*, pages 2263–2291, 2013.

Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4902–4909, 2019.

Zijian Shi and John Cartlidge. Neural stochastic agent-based limit order book simulation: A hybrid methodology. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 2481–2483, 2023.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28, 2015.

Victor Storchan, Svitlana Vyetrenko, and Tucker Balch. Learning who is in the market from time series: market participant discovery through adversarial calibration of multi-agent simulators. *arXiv preprint arXiv:2108.00664*, 2021.

Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Yuan Tian, Qin Wang, Zhiwu Huang, Wen Li, Dengxin Dai, Minghao Yang, Jun Wang, and Olga Fink. Off-policy reinforcement learning for efficient and effective gan architecture search. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16*, pages 175–192. Springer, 2020.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

Svitlana Vyetrenko, David Byrd, Nick Petosa, Mahmoud Mahfouz, Danial Dervovic, Manuela Veloso, and Tucker Balch. Get real: Realism metrics for robust limit order book market simulations. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

Magnus Wiese, Robert Knobloch, Ralf Korn, and Peter Kretschmer. Quant gans: deep generation of financial time series. *Quantitative Finance*, 20(9):1419–1440, 2020.

Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. *Advances in neural information processing systems*, 30, 2017.

Shuai Xiao, Hongteng Xu, Junchi Yan, Mehrdad Farajtabar, Xiaokang Yang, Le Song, and Hongyuan Zha. Learning conditional generative models for temporal point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Kang Zhang, Guoqiang Zhong, Junyu Dong, Shengke Wang, and Yong Wang. Stock market prediction based on generative adversarial network. *Procedia computer science*, 147:400–406, 2019.

Appendix of "INTAGS: Interactive Agent-Guided Simulation"

A Extended Literature Survey

Here, we extend our literature survey to more relevant topics.

Stock market data simulation. One particular application of interest is Limit Order Book (LOB) stock market simulation, a field initially studied via Interactive Agent-Based Simulation [Macal and North, 2005] that models and simulates interactions among background agents, i.e., market participants. Presently, the SOTA parametric approach is Agent-Based Interactive Discrete Event Simulation (ABIDES) [Byrd et al., 2020, Amrouni et al., 2021]. As mentioned earlier, the adaption to the unique characteristics of the underlying application is crucial to the success of generative models. However, oftentimes it would be too difficult to explicitly describe those rules, especially for trading activities, leading to the application of advanced machine learning techniques to summarize those rules from data. Under the context of LOB market simulation, precisely specifying the type (momentum agents, value agents, noise agents, etc.) and quantity of background agents to emulate real multi-agent systems is rather challenging, let alone the fact that agents' identities are missing for the MAS calibration [Coletta et al., 2023b]. Thus, there is an increasing amount of work replacing parametric models with easy-to-tune neural networks (NNs) to effectively capture the trading market's complex dynamics, offering a more flexible and scalable framework for training a generative model to capture the complex market dynamics. Notably, Stock-GAN [Li et al., 2020] utilized cWGAN to generate limit orders; Later, Coletta et al. [2021, 2022] extended Stock-GAN and introduced the concept of a *world agent*, which can not only place limit and market orders but also cancel and replace orders.

The aforementioned GAN-based approaches explored various NN architectures, trying to adapt to the corresponding applications, but none of them went beyond the classic local loss that directly penalizes the action of background agents (i.e., the output of the generator) for given input market state — one exception is Shi et al. [2019], where they studied the customer/agent policy, i.e., how to place BUY orders in the e-commerce market, with RL, and leveraged adversarial imitation learning [Ho and Ermon, 2016, Torabi et al., 2018] that replaced the critic in GAN with the agent reward. On one hand, as the generative model tries to capture the policy of the BG agents, it is natural to use RL or imitation learning (IL). In the same spirit of our adaptation of SeqGAN, Shi et al. [2019] used IL for generator training to account for the sequential nature of the market. However, to the best of our knowledge, Shi et al. [2019] seems to be the only attempt along the direction of IL for market simulation. On the other hand, as we will see later, using the reward directly as the loss cannot capture the causal relationship that can discriminate between real and synthetic markets. Additionally, Shi et al. [2019] has limitations as it can only place one type of order.

Generative models in finance. Training experimental agents usually involves interaction with the real market, which is often impractical. To create a suitable market for the experimental agent, generative models representing all background agents are needed and have gained prominence in finance, with GAN being a particularly popular choice. To the knowledge of the author, this line of research traces back to Kumar et al. [2018], Shi et al. [2019], who adapted GAN to generate BUY orders in e-commerce markets. Notably, GAN has been utilized to model more complex stock markets and simulate various types of stock market data, including transaction event time [Xiao et al., 2017, 2018], price [Zhang et al., 2019, Wiese et al., 2020, Koshiyama et al., 2021], and even orders [Li et al., 2020, Coletta et al., 2021, 2022].

Generative models for RL. Our proposed INTAGS is closely related to RL: On one hand, the generator is formulated as a stochastic RL policy and trained via policy gradient. On the other hand, the downstream task — the Exp agent — can be done by RL. In literature, reinforcement learning and generative models have been traditionally considered separate fields until some recent developments reveal their promising connections. One notable application involves leveraging generative models in RL training, particularly relevant in the context of optimal execution tasks in financial markets. Indeed, it is a popular approach to study financial tasks with an interactive simulator: For example, Karpe et al. [2020] trained RL agent to perform optimal execution tasks by interacting with parametric market simulator ABIDES [Byrd et al., 2020, Amrouni et al., 2021] such that the training process captures real-world dynamics, enabling agents to grasp the consequences of their actions on the

responses of other market participants; Kuo et al. [2021], Koshiyama et al. [2021] trained agents to perform downstream financial tasks with the help of a conditional GAN-based market simulator. One notable work studying the connection between GAN and RL is Ho and Ermon [2016], which proposed to mimic the expert policy from instances by inverse RL (IRL) followed by RL; They theoretically proved that explicitly learning of the cost function in the IRL could be bypassed, which enabled end-to-end learning of the policy from the expert policy instances. Moreover, the resulting imitation learning problem takes a GAN formulation, bridging RL and GAN from a very novel perspective. Other contributions in this direction include Finn et al. [2016], Fu et al. [2017], who formulated IRL as GAN problem.

RL for Generative models. As pointed out by Yu et al. [2017], Shi et al. [2019] as well as our work, the generator can be formulated as the (RL) agent policy, and therefore imitation learning can be used for market simulation, which opens up more possibilities in this area (i.e., market simulation). However, it is difficult to train adversarial imitation learning models in practice since careful reward augmentation [Bhattacharyya et al., 2019] and curriculum design [Behbahani et al., 2019] are needed in practice, and this might explain why GAN is still the most popular approach for market simulation. We refer readers to Torabi et al. [2019] for a recent survey on imitation learning. On the other hand, as most GAN-based simulators, Ho and Ermon [2016] also used the classic loss function and this is the key difference from our work: the proposed interactive agent-based environment distance metric and the derived training framework — INTAGS — are the main contribution of our work. Another seemingly closely related work in this direction is Ruiz et al. [2019], who studied the classification problem in the presence of limited real data via a classifier trained on generative models. They claimed to leverage RL to help with the GAN training. However, as pointed out by the reviewer, even though the feedback from evaluating the classifier on the real data can help improve the generative model, there is no clear state and action space definition nor sequential decision-making, rendering the claim of reformulating the GAN training as RL less convincing. Other works that leveraged RL to help with the GAN training include: Sarmad et al. [2019], who proposed to use RL to control the input to generators in the GAN, and Tian et al. [2020], who proposed to use RL to help search for the optimal GAN architecture. However, non of those works leverage sequential agents' interactions to develop a metric for the training of a generative model, which is the key difference between our work and those existing works.

B Additional Details of INTAGS

B.1 Causal effect estimation for feedback

One of our main contributions is the novel interactive agent-based MAS metric, which compares the discrepancy of the feedback (empirical) distributions under different environments. The feedback is defined over a longer sequential state-action-next state chain, compared to the previous state-action pair used in classic cWGAN. By considering a longer chain, our metric considers and introduces a much more complex sequential dependency.

To understand what is the complex sequential dependency, let us first recall one proposed feedback in the market simulation application: the `Mkt2NextReturn`; That is, the feedback is the causal effect from Exp AT agent placing market (BUY) order to the next market return. However, at time step t, the market evolution s_t is not only the result of Exp AT agent's action a_t and the BG agents' responses A_t^j's, but also *correlated* to previous market return s_{t-1} — this is the additional sequential dependency introduced by our feedback; in the classic setting, the BG agents' state-action pairs (S_t^j, A_t^j)'s have very clear and simple dependency that action A_t^j only depends on the input market state S_t^j.

Next, let us elaborate on why introducing such additional sequential dependency poses a challenge in feedback estimation. Again let us consider feedback `Mkt2NextReturn` in the market simulation; To estimate f from collected sequential observations, one can *select the time steps where* $a_t = \text{placing market order}$ and take the average of corresponding s_t's as the estimator. To be precise, the naive estimator of feedback f, i.e., the causal effect from Exp agent action to the next state, say j-th

element of the state vector, is defined as

$$f_{\text{naive}} = \frac{1}{\#\{t : a_t = \text{placing market order}\}} \sum_{t=1}^{T} s_t(j) \mathbf{1}_{\{a_t = \text{placing market order}\}},$$

where $\#$ denotes the cardinality of a set, $s_t(j)$ is the j-th element of the state vector (e.g., return), and $\mathbf{1}$ is the indicator function.

The problem of the above naive estimator is the *selection bias*, i.e., the selected sub-population is not representative enough of the whole population, making the average effect estimator above a biased one; To understand this, let us consider a very simple example, where we want to study the effect of carrying a lighter to developing lung cancer: If the effect from smoking, which acts as a common cause to both carrying a lighter and developing lung cancer, is not considered, a false causal conclusion (which is indeed just correlation) that carrying a lighter will result in lung cancer will be made. Under the potential outcome framework by Rubin [1974], the causal effect estimation from the treatment variable to the outcome variable needs to take into account the effect of pre-treatment covariates, called potential confounders.

There are two solutions: **(Apch1)** One naive approach is to manually break the correlation between s_{t-1} and a_t; In the market simulation, we can use zero-intelligent Exp AT agent that aggressively or randomly place orders; However, as we mentioned in our discussion (see Section 5), the deployment of INTAGS needs the Exp agent to be the real one in the production system and therefore it is unlikely that such an Exp agent is zero-intelligent. **(Apch2)** One very popular approach to estimate causal effect from observational data is to leverage inverse probability weighted (IPW) estimator [Horvitz and Thompson, 1952] to adjust for the potential confounders; The high-level idea is to adjust the weight of each selected sample such that the re-weighted sub-population is the same with the whole population in theory, addressing the issue of selection bias. To be precise, we denote the propensity score, which is the probability of "receiving treatment", as

$$e(s) = \mathbb{P}(a = \text{placing market order} \mid \text{state} = s) = \pi(s, a),$$

where π is the policy of the Exp agent (or AT agent in our market simulation application). The IPW estimator of feedback f is defined as:

$$f_{\text{IPW}} = \frac{1}{\#\{t : a_t = \text{placing market order}\}} \sum_{t=1}^{T} \frac{s_t(j) \mathbf{1}_{\{a_t = \text{placing market order}\}}}{\hat{e}(s_{t-1})},$$

where $\hat{e}$ denotes the estimated propensity score (typically through logistic regression). In practice, to ensure estimates $\hat{e}$'s are not overly small, they are typically clipped using a certain threshold (denoted by PSthres), i.e., $\hat{e} \leftarrow \max\{\hat{e}, \text{PSthres}\}$.

B.2 Value function evaluation

The value function Q_f estimation relies on N' feedbacks from the real Env and N feedbacks obtained by performing N complete rollouts under the world Env. As illustrated in Figure 5, the interaction

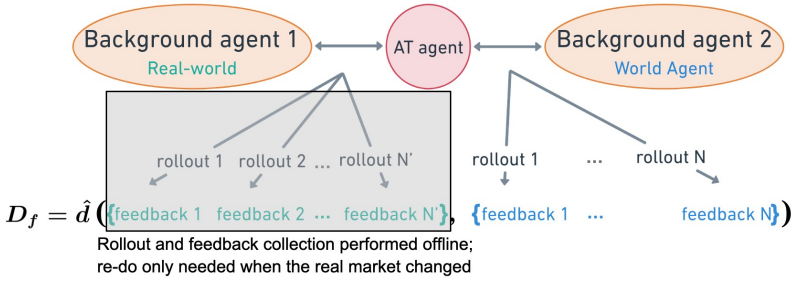

Figure 5: Illustration of the calculation of our proposed interactive agent-based distance metric.

under real Env to obtain the feedback only needs to be performed offline; During the training of the world BG agent policy/generator network in Algorithm 1, the interaction under world Env and the network parameter update are performed iteratively.

In this work, the probability distribution distance metrics we consider are MMD, ED, and EMD. Other discrepancy metrics such as Kullback–Leibler divergence and Jensen–Shannon divergence are less favorable as we can observe the supports of two empirical probability distributions are different in Figures 2, 10 and 9 (especially when the number of rollouts is small).

Given two sets of samples $\{f_1, \ldots, f_N\}$ and $\{g_1, \ldots, g_{N'}\}$, an unbiased estimator of MMD can be obtained through the following U-statistic:

$$\hat{d}\big(\{f_1, \ldots, f_N\}, \{g_1, \ldots, g_{N'}\}\big) = \frac{\sum_{i \neq j} k(f_i, f_j)}{N(N-1)} + \frac{\sum_{i \neq j} k(g_i, g_j)}{N'(N'-1)} - \frac{2\sum_{i,j} k(f_i, g_j)}{NN'},$$

where $k(\cdot, \cdot)$ is the user-specified kernel function. Popular choices include Gaussian kernel with bandwidth parameter $\sigma > 0$, $k(f_1, g_1) = \exp\{\|f_1 - g_1\|_2^2/(2\sigma^2)\}$, where $\|f_1 - g_1\|_2$ represents the Euclidean distance.

In addition, the empirical estimate of Energy Distance is as follows:

$$\hat{d}_{\mathrm{ED}}(\{f_1, \ldots, f_N\}, \{g_1, \ldots, g_{N'}\}) = \frac{\sum_{i \neq j} \|f_i - f_j\|_2^2}{N(N-1)} + \frac{\sum_{i \neq j} \|g_i - g_j\|_2^2}{N'(N'-1)} - \frac{2\sum_{i,j} \|f_i - g_j\|_2^2}{NN'},$$

It is worthwhile noting that ED is a special case of MMD with linear kernel; Indeed, their equivalence has already been established [Sejdinovic et al., 2013]. Lastly, the estimation of Earth Mover's Distance is done by a popular built-in implementation in `wasserstein_distance` in `scipy.stats`; we omit further details as EMD is a less favorable $\hat{d}$ choice according to our empirical evidence.

B.3 Density estimation

Here, since the generator network G_θ is a deterministic transformation, we apply kernel density estimation (KDE) to obtain the induced density function p_θ; We choose KDE since it enjoys a closed-form and differentiable solution. In our experiment, we use a Gaussian RBF kernel with a bandwidth parameter selected by the median heuristic (which is referred to as adaptive bandwidth). Note that using KDE in the objective function is not novel in literature, and there exists work that proposed to include the bandwidth as a learn-able parameter during the optimization [Gomez-Alanis et al., 2020], which is referred to as optimized bandwidth. Alternatively, one can choose a fixed bandwidth parameter in KDE; Here, since the bandwidth choice is not a major contribution in our work, we leave the performance of those bandwidth choices (i.e., adaptive, fixed or optimized; this work takes the adaptive one) as future development and the user can freely choose among them during the implementation.

B.4 Training algorithm

In this part, we present the algorithm for training INTAGS. Before that, we give the derivation of the gradient as follows:

$$\begin{aligned}
\nabla_\theta \mathcal{L}(\theta) &\simeq \sum_{t=1}^{T} \sum_{A_t \in \mathcal{A}} \nabla_\theta p_\theta\left(A_t | S_t^{\tau_t}\right) \cdot Q_f\left(\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}, A_t\right) \\
&= \sum_{t=1}^{T} \sum_{A_t \in \mathcal{A}} p_\theta\left(A_t | S_t^{\tau_t}\right) \nabla_\theta \log p_\theta\left(A_t | S_t^{\tau_t}\right) \cdot Q_f\left(\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}, A_t\right) \\
&= \sum_{t=1}^{T} \mathbb{E}_{A_t \sim p_\theta(\cdot | S_t^{\tau_t})}\left[\nabla_\theta \log p_\theta\left(A_t | S_t^{\tau_t}\right) \cdot Q_f\left(\tilde{\mathcal{T}}_{\pi,\theta}^{1:t}, A_t\right)\right].
\end{aligned}$$

Moreover, we use graphical illustration to show how to perform Algorithm 1 in practice. In particular, we illustrate the forward pass and back-propagation to obtain the gradient in Figure 6.

Algorithm 1 Training of Interactive Agent-guided Simulation

Input: Exp agent π, feedback $f(\boldsymbol{\mathcal{T}}_{\pi,\texttt{real}})$ (4), empirical probability distance $\hat{d}$, learning rate r, time horizon T, batch size b, rollout number N.

while convergence not achieved **do**
  Perform a complete rollout following p_θ: $\mathcal{T}_{\pi,\theta}^{1:T}$.
  for $t = 1$ **to** T **do**
    Given prefix $\tilde{\mathcal{T}}_\theta^{1:t} = \{\ldots,\ S_t^{\tau_t}\} \subset \mathcal{T}_{\pi,\theta}^{1:T}$, sample: $A_{t,1}, \ldots, A_{t,b} \sim p_\theta(\cdot|S_t^{\tau_t})$.
    for $k = 1$ **to** b **do**
      Finish N MC rollouts with prefix $\tilde{\mathcal{T}}_\theta^{1:t} \cup \{A_{t,k}\}$ following p_θ: $\boldsymbol{\mathcal{T}}_{\pi,\theta}^{\mathrm{MC}}$ (3).
      Compute: $Q_f^{t,k} = \hat{d}\left(f(\boldsymbol{\mathcal{T}}_{\pi,\theta}^{\mathrm{MC}}), f(\boldsymbol{\mathcal{T}}_{\pi,\texttt{real}})\right)$.
    end for
  end for
  $\theta \leftarrow \theta - r\sum_{t=1}^{T}\frac{1}{b}\sum_{k=1}^{b} Q_f^{t,k}\nabla_\theta \log p_\theta\left(A_{t,k}|S_t^{\tau_t}\right)$.
end while
return θ

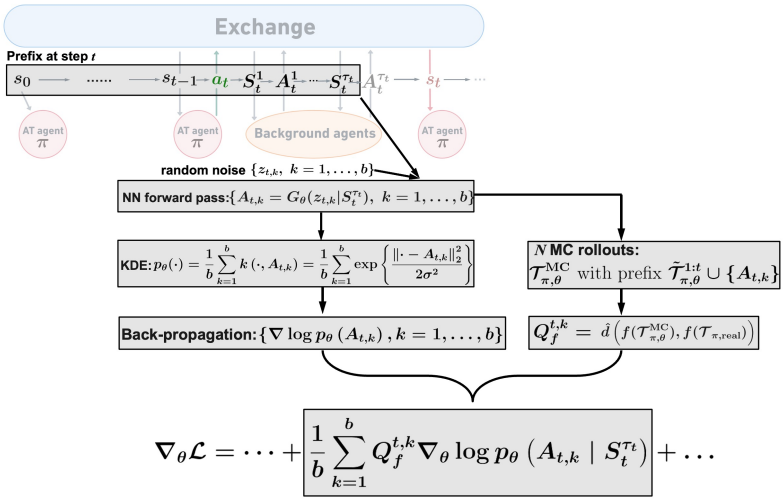

Figure 6: Illustration of Algorithm 1 — the forward-backward pass and and the gradient evaluation.

C Additional Background Knowledge of Market Simulation Experiment

In financial markets, traders buy (bid) and sell (ask) financial assets based on the available market information, aiming to make a profit. The buy and sell trades/orders are processed through a matching engine, also known as an *exchange*, that operates under specific rules (typically "first in first out"). The most common orders are market orders and limit orders: Market orders are executed immediately at the prevailing bid or ask price, whereas limit orders set an upper (or lower) price threshold for a buy (or sell) order, and thus may not be (immediately) executed until there is a matching counterparty. As illustrated in Figure 7, the LOB [Gould et al., 2013] organizes and displays outstanding (i.e., unexecuted) limit orders, providing insights into market conditions and liquidity; Its statistical properties, called *stylized facts* [Cont, 2001, Vyetrenko et al., 2020], such as mid-price and spread, can help traders with informed decision-making. One fundamental problem in financial modeling is algorithmic trading (AT), where an algorithmic agent replaces human traders such that it can analyze large volumes of market data, identify patterns, and execute orders at high speeds. Meanwhile, the market responds to the Exp agent's actions in the way that other participants, i.e., BG agents, adjust their strategies and place orders accordingly. This Exp agent can be designed/trained to perform a variety of tasks, e.g., optimal execution, etc.

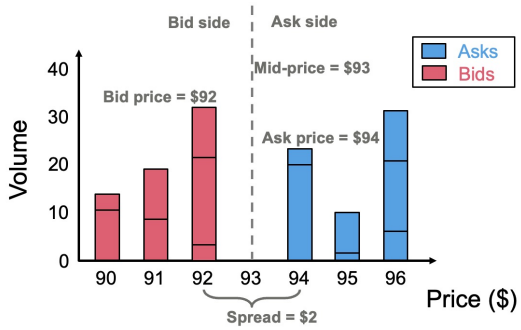

Figure 7: **The LOB structure for a specific assert.**

C.1 Limit order book

LOB dynamics. We begin with introducing the dynamics of LOB, as shown in Figure 8; We illustrate this dynamics of the LOB with three cases:

- Case 1: A buy market order with volume 25 arrives, leading to 25 shares of order executed on ask side based on price and order arrive time. The execution price will be the ask price, and the resulting LOB market will have mid-price and spread shifted to $\$93.5$ and $\$3$.
- Case 2: A buy limit order with price $\$98$ and volume 25 arrives. As $\$98$ exceeds the current ask price (i.e., $\$94$), the incoming will be executed at the ask price. The resulting LOB market will be exactly the same with case 1.
- Case 3: A but limit order with price $\$93$ and volume 10 arrives. Since there will be no matching counterparty on the ask side, this order will be placed in LOB, resulting in LOB market mid-price and spread shift to $\$93.5$ and $\$1$.

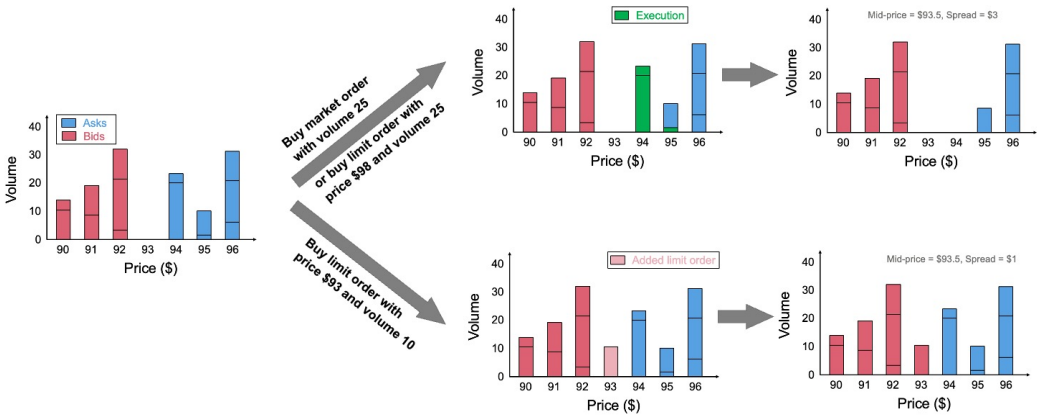

Figure 8: Illustration of the dynamics of LOB.

Stylized facts. Next, we define stylized facts used in this work following the notation in Coletta et al. [2022]. We denote $p_a^i(t), v_a^i(t), p_b^i(t), v_b^i(t)$ as the price and volume at i-th level of the LOB, at time t, for ask and bid respectively. For example, in the starting state of the LOB (left panel) shown in Figure 8, $p_a^1(t) = \$94$ and $p_b^1(t) = \$92$. The depth of a limit order with price threshold $p(t)$ is defined as

$$d(t) = \begin{cases} p_b^1(t) - p(t), & \text{If side = bid,} \\ p_a^1(t) + p(t), & \text{otherwise.} \end{cases}$$

The stylized facts used here are:

- Mid-price, average of ask price and bid price, i.e.,

$$m(t) = \frac{p_a^1(t) + p_b^1(t)}{2}.$$

- Return and price impact:

$$r(t_1, t_2) = \log \frac{m(t_1)}{m(t_2)}.$$

  Typically the return at time t is defined as $r(t, t-1)$; our "price impact" is defined as $r(t, 0)$, which measures the return w.r.t. to the beginning of our market scenario.

- Spread:

$$\delta(t) = p_a^1(t) - p_b^1(t).$$

- Volume imbalance, the demand and supply inequality within the first n-levels, i.e.,

$$I^n(t) = \frac{\sum_{j=1}^n v_b^j(t)}{\sum_{j=1}^n v_b^j(t) + v_a^j(t)}. \tag{5}$$

- Absolute volume within the first n-levels:

$$V^n(t) = \sum_{j=1}^n v_b^j(t) + v_a^j(t).$$

  In particular, the bid (or buy) and ask (or sell) volumes within the first n-levels are defined as

$$V_b^n(t) = \sum_{j=1}^n v_b^j(t), \quad V_a^n(t) = \sum_{j=1}^n v_a^j(t). \tag{6}$$

C.2 Optimal execution and reinforcement learning

One most famous framework for optimal execution (or optimized trade execution), where the goal is to sell a target share of stocks within a finite time horizon, would be the Almgren–Chriss Model [Almgren and Chriss, 2001], where the market dynamics takes a parametric form, leading to a "pre-planned strategy that does not depend on real-time market conditions" [Hambly et al., 2021]. To address this issue, Nevmyvaka et al. [2006] first applied RL in the optimal execution problem. However, their RL training used (historic) real data, in which there will be no responses from other traders. [Karpe et al., 2020] leveraged the parametric market simulator ABIDES [Byrd et al., 2020, Amrouni et al., 2021] to generate training data to address this issue. On the contrary, Patel [2018] did consider multiple agents in the RL. For recent advancements in this direction, we refer readers to a nice survey by Hambly et al. [2021] (see Section 4.2 therein).

Reinforcement learning for optimal execution. In the aforementioned optimal execution task, e.g., acquiring q_0 shares of one particular asset within T, the objective function of finding the optimal policy $\pi^\star$ via online RL problem is:

$$\pi^\star = \arg\max_\pi \mathbb{E}_\pi \left[\sum_{t=1}^T \gamma^{t-1} R(s_{t-1}, a_t) - \mathrm{P} q_T \right],$$

where $\gamma \in [0, 1]$ is the discount factor that determines the importance of future rewards, $\mathbb{E}_\pi$ denotes the expectation over trajectories generated by policy π, $q_T \geq 0$ denotes amount of unfulfilled shares to meet the target q_0 at the end of rollout, and $\mathrm{P} > 0$ is the penalty per unfulfilled share.

Rule-based policy, such as aggressively placing q_0 market orders at the first step, is feasible, but will greatly impact the market and result in sub-optimal profit; Indeed, interacting with the market by executing orders will change the market dynamics, and therefore executing (small) orders sequentially according to the currently observed state is a more sensible strategy.

Deep Q-Network. One popular approach to find the optimal policy for optimal execution is to optimize the action-value function, i.e., the Q-function, which represents the expected cumulative discounted reward starting from state s, taking action a, and following a particular policy. The optimal Q-function, defined as $\tilde{Q}^\star(s,a) = \max_\pi \mathbb{E}_\pi[\sum_{k=t}^{T} \gamma^{k-t} R(s_{k-1}, a_k) \,|\, s = s_t, a = a_t]$, obeys the Bellman equation:

$$\tilde{Q}^\star(s,a) = \mathbb{E}_\pi \left[R(s,a) + \gamma \max_{a' \in \tilde{\mathcal{A}}} \tilde{Q}^\star(s',a') \,|s,a \right],$$

where a' is the next-state after agent takes action a at state s. When the state and action spaces are discrete, one can estimate the optimal Q-function by iteratively updating the Q-values as follows:

$$\tilde{Q}_{i+1}(s,a) \leftarrow \tilde{Q}_i(s,a) + \alpha \left(R(s,a) + \gamma \max_{a' \in \tilde{\mathcal{A}}} \tilde{Q}_i(s',a') - \tilde{Q}_i(s,a) \right),$$

where α is the learning rate that controls the weight given to new information. According to Sutton and Barto [2018], this value iteration algorithm converges to the optimal solution, i.e., $\tilde{Q}_i(s,a) \to \tilde{Q}^\star(s,a)$ as $i \to \infty$. In practice, especially when dealing with high-dimensional or continuous state spaces, the Q-function is often approximated using a Deep Q-Network (DQN), i.e., $\tilde{Q}(s,a;\tilde{\theta}) \approx \tilde{Q}^\star(s,a)$, where a deep NN parameterized by parameter $\tilde{\theta}$ is used to represent the Q-function, enabling generalization to unseen states [Mnih et al., 2013]. The DQN is trained by minimizing the following objective function (typically via gradient-based method):

$$\tilde{\mathcal{L}}(\tilde{\theta}) = \mathbb{E}\left[\left(R(s,a) + \gamma \max_{a' \in \tilde{\mathcal{A}}} \tilde{Q}(s',a';\tilde{\theta}) - \tilde{Q}(s,a;\tilde{\theta}) \right)^2 \right].$$

C.3 Baseline Simulator: Conditional Wasserstein GAN

Despite the success of variational autoencoder (VAE) [Sohn et al., 2015] under many other contexts, the most popular machine learning approach for generating market data is still GAN. Conditional GAN (cGAN) [Mirza and Osindero, 2014] extends the standard GAN [Goodfellow et al., 2014] framework by incorporating conditional information. The generator network, which acts like a trading agent, G_θ takes random noise input z and a conditional variable S to produce synthetic orders A (i.e., random sample z is generated from a prior distribution, such as Gaussian distribution, and then transformed by G_θ to "match" the true distribution). The discriminator network D estimates the probability that an order is real by considering both the generated order $G_\theta(z|S)$ and the true order A along with the corresponding conditional variable S. The training of a cGAN involves optimizing a min-max objective function:

$$\min_\theta \max_D \; \mathbb{E}_{A|S\sim p_{\texttt{real}}}[\log D(A|S)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G_\theta(z|S)))],$$

where $p_{\texttt{real}}$ represents the real data distribution and p_z represents the prior noise distribution, and S represents the feature/input that cGAN will condition on.

Even though there have been modern machine learning techniques adapted to generative models under various specific domains, such as imitation learning for traffic simulation to test self-driving algorithms [Suo et al., 2021], such adaption for LOB stock market simulation is largely missing until Li et al. [2020], Coletta et al. [2021, 2022], who leveraged cWGANs to train a NN-based world BG agent that places realistic orders based on relevant market information. By utilizing a cWGAN, the world BG agent can learn from the simulated market data, improving its trading strategies in response to the dynamic market. In particular, Coletta et al. [2021] used Wasserstein distance as the discriminator, i.e., they adopted Wasserstein GAN [Arjovsky et al., 2017]:

$$\min_\theta \max_{w \in \mathcal{W}} \mathbb{E}_{A|S\sim p_{\texttt{real}}}\left[f_w(A|S)\right] - \mathbb{E}_{z\sim p_z}\left[f_w\left(G_\theta(z|S)\right)\right],$$

where f_w is the discriminator and the function G_θ is the generator. They used gradient descent w.r.t. w and θ iteratively to optimize the objective. Additionally, there exist methods to calibrate ABIDES with advanced machine learning techniques such as GAN [Storchan et al., 2021, Shi and Cartlidge, 2023], the mainstream market simulator is still based on GAN due to its superior performance; another main reason is that the agent identity information is missing is real market data, making it difficult to calibrate the parametric market simulator.

D Additional Experimental Details of Algorithmic Trading-guided Market Simulation

D.1 Training details

Configuration. As we do not have access to real trading data, we consider a state-of-art parametric simulator as the *real Env*, where it is easier to obtain the "ground truth"; The goal of our experiments is to provide a proof-of-concept and support the generalization to the real market. In particular, we take ABIDES [Byrd et al., 2020] with explicitly specified/parametric BG agents as "reality". Using ABIDES data we train our simulated *world BG agent*. The Exp agent is chosen as the AT agent that performs the optimal execution task, aiming to acquire a fixed share of a single asset within a fixed time horizon while minimizing the transaction cost. To improve simulator resilience under unseen and rare market states [Coletta et al., 2023a], the AT agent places aggressive orders. Its input includes private state (current time and remaining shares) and market state (such as spread). The world BG agent takes the market state from the past five steps as input [Coletta et al., 2022].

In our experiment, the configuration of the optimal execution task is: time horizon $T = 10$, parent order size $q_0 = 50$. For the AT agent performing this task, the private state at time step t for the AT agent is normalized elapsed time t/T, remaining shares to acquire normalized by total parent order size q_0, and their difference; The market state consists of imbalance ((5) with all levels and $n = 5$), spread, price impact, and direction feature. The action space $\tilde{\mathcal{A}} = \{0, 1, 2\}$, where 0 stands for executing fixed size Market Order, 1 stands for executing fixed size (10 shares) Limit Order, and 2 stands for action hold (i.e., no action). In our experiments, the external agent to places orders aggressively such that the world BG agent is able to encounter diverse market states. Additionally, this eases the implementation burden as rollout using this simple rule-based AT agent (instead of NN-based agent) can be performed in parallel easily, which leads to reduced computational cost. In particular, we choose an aggressive agent which continues to place limit orders until the parent order is fulfilled, which results in $T = 5$ in practice. The parametric BG agents' (treated as "reality" in the experiment) configuration is: one Exchange Agent, two Adaptive Market Maker Agents, 100 Value Agents, 25 Momentum Agents, and 5000 Noise Agents; Please see Byrd et al. [2020], Amrouni et al. [2021] for additional details on the parametric market simulator ABIDES[4].

Training hyperparameters. As mentioned previously, the complete rollout number is chosen as $N = 5$ such that the feedback's empirical distribution under the world BG agent market is sufficiently different from that of the "real market". To ensure rollouts under the "real market" will not incur extreme feedbacks, we conduct $N' = 100$ offline rollouts to construct the feedback collection under world market. As T_0 and b together act like the batch-size, we consider grid search over $T_0 \in \{3, 4, 5\}$ and $b \in \{3, 5\}$. By this grid search, we choose the combination with the smallest (proposed) metric, which is

- $T_0 = 5$ and $b = 3$ for `Mkt2NextReturn`, which corresponds to the results in Figure 4 and the **second** column in Figures 14, 15, 16 and 17.
- $T_0 = 5$ and $b = 3$ for `Mkt2NextPriceImpact`, which corresponds to the results in the **forth** column in Figures 14, 15, 16 and 17.

To compare the results of `Mkt2NextReturn` with `EpisodeReward`, we use the same hyperparameter setting for `EpisodeReward`, i.e.,

- $T_0 = 5$ and $b = 3$ for `EpisodeReward`, which corresponds to the results in the **third** column in Figures 14, 15, 16 and 17. The corresponding AT-based metric is optimized from 0.7156 to 0.6176.

Additionally, to further demonstrate of the effectiveness of our AT-based metric and ATMS, we consider

- $T_0 = 2$ and $b = 5$ for `Mkt2NextReturn`, which corresponds to the results in Figure 13.

The initial learning rate r is chosen to be 10^{-9}, which decreases by half every 10 iteration. There are in total 100 iterations and based on the decrease of learning rate they are divided into 10 epochs

[4]Available at `https://github.com/jpmorganchase/abides-jpmc-public`

(each with 10 iterations). We use the cWGAN network parameter trained on AINV market reply data (which is different from the "real" market) as the warm-start/initialization when we train ATMS.

D.2 Additional results for other feedback candidates

In Figure 9, we present the empirical distribution of other feedback candidates. Interestingly, when utilizing various stylized facts, with the exception of price impact, as the next state, we observe a similar pattern to `EpisodeReward`. This similarity renders these alternatives unfavorable choices in our proposed ATMS and might suggest that those stylized facts are less representative of the market. It is important to mention that we observe a similar pattern of the results for `Mkt2NextPriceImpact` (shown in the last row of Figure 9) to that of our ideal candidate `Mkt2NextReturn` (shown in the second row of Figure 2 in the main text of the paper). Therefore, `Mkt2NextPriceImpact` is another candidate that can be readily used to train/improve the world BG agent.

Furthermore, we report the results of `Mkt2Reward` in Figure 10, which show similar patterns to `EpisodeReward`, providing strong evidence to support our claim that `Mkt2Reward` solely depends on the current LOB status and the exchange rule, and thus cannot represent the BG agents.

D.3 Additional results for other $\hat{d}$ candidates

In addition to the result for MMD depicted in Figure 3, we present the outcomes for Energy Distance and Earth Mover's Distance in Figure 11. In the first row where f is chosen as `EpisodeReward`, we can see the resulting market distance metric's behavior is reasonable in the sense that the metric is closer to zero when comparing two identical markets (indicated by the orange line, "Real vs Real"). However, the metric performs poorly to differentiate markets as we cannot see the pattern that blue line is significantly larger than the orange one with small rollout number (as shown in right panel in Figure 3).

From the second row in Figure 11, where `Mkt2NextReturn` is chosen as f, it is surprising to find that the metric's behavior appears unreasonable as the metric between different markets is smaller than that when two markets are identical. The above observations indicate that the corresponding metrics may not be suitable for capturing the dissimilarity (or similarity) between different markets accurately, and therefore ED and EMD are less favorable candidates for the subsequent simulator training.

For completeness and to provide further insights into their unsuitability (for subsequent training of ATMS), we also include the results for MMD, ED, and EMD when choosing `Mkt2Reward` as feedback f in Figure 12. These additional results serve as direct evidence highlighting the undesirable characteristics of these metrics in the context of our ATMS. The findings reinforce the significance of developing and employing a specialized distance metric, as we propose, to effectively evaluate the distance between financial markets, ensuring the reliability of our ATMS.

D.4 Additional evidence for the effectiveness of ATMS

To further establish the reliability of our proposed ATMS, we conduct additional experiments to investigate the impact of randomness and training hyperparameters on its effectiveness. In Figure 13, we present the results obtained using a different random seed and distinct training hyperparameters. Remarkably, we observe a similar pattern to the findings depicted in Figure 4. This consistency across different experimental setups strongly suggests that the effectiveness demonstrated in Figure 4 is not solely attributed to randomness but indeed indicative of the superior performance of our proposed ATMS. These findings reinforce the credibility and practical applicability of our ATMS, affirming its capability to consistently generate realistic market data regardless of varying random initialization and training conditions.

D.5 Comparison with cWGAN baseline

Next, we show that our ATMS outperforms existing cWGAN baseline [Coletta et al., 2022]; See Appendix C.3 for further details. Interestingly, we find that the stylized facts are not only similar to the target market but also show more balanced BUY and SELL volumes. While the cWGAN training can be biased towards one direction Coletta et al. [2023a], our novel approach seems to

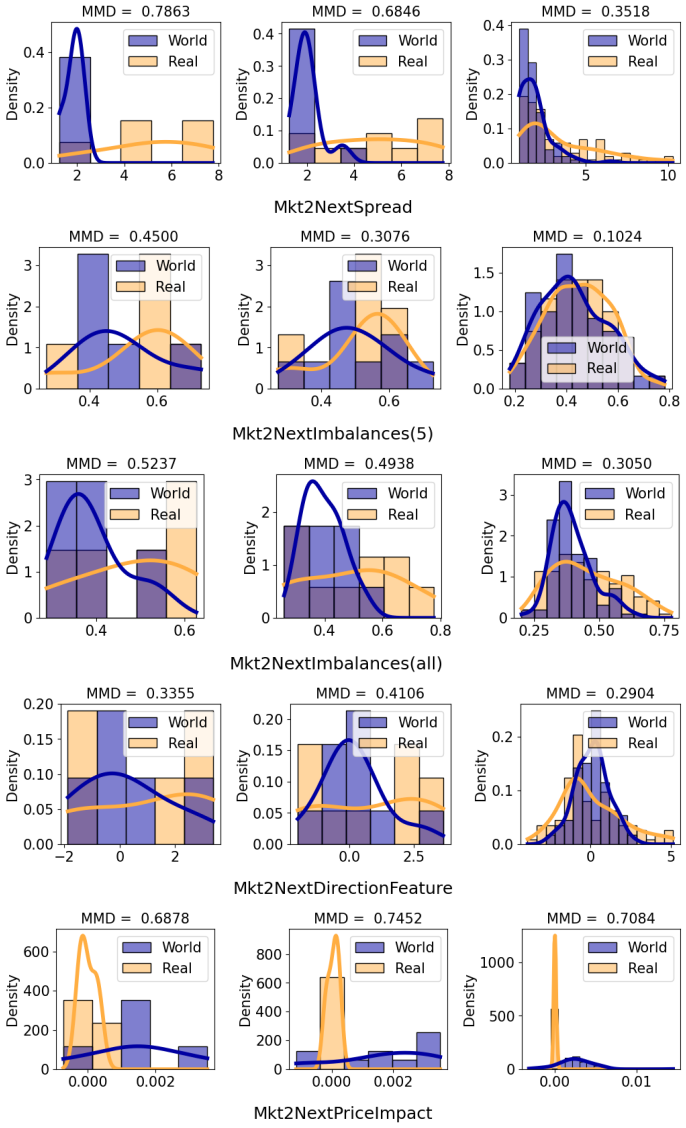

Figure 9: The empirical distribution of additional causal effect feedback candidates (from top to bottom, we report `Mkt2NextSpread`, `Mkt2NextImbalances(5)`, `Mkt2NextImbalances(All)`, `Mkt2NextDirection`, and `Mkt2NextPriceImpact`) when performing multiple (from left to right: $5, 10, 100$) rollouts using AT agent under real (orange) or world BG agent (blue) markets. Again, the idea feedback should be able to differentiate different markets with only a few rollouts, making the `Mkt2NextPriceImpact` the most ideal candidate in this figure.

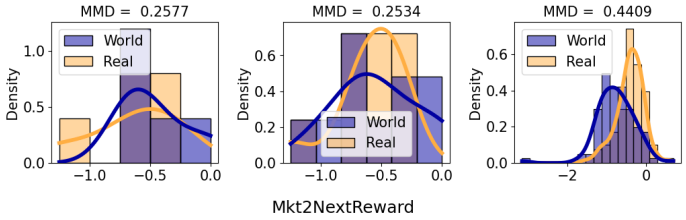

Figure 10: The distribution of `Mkt2Reward` when performing multiple (from left to right: $5, 10, 100$) rollouts using AT agent under real (orange) or world BG agent (blue) markets. Apparently, when there are only around 5 rollouts, `Mkt2Reward` as f cannot faithfully differentiate the markets, making it a poor candidate. This reaffirms our claim the the feedback must depend on the underlying BG agents.

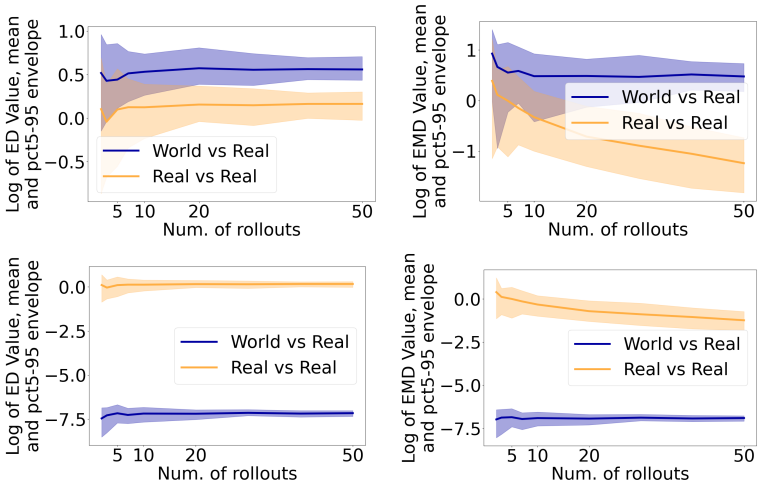

Figure 11: The mean and $5-95\%$ envelop trajectory (over 50 bootstrap trials) against increasing number of rollouts of our proposed market distance metric with feedbacks `EpisodeReward` (top) and `Mkt2NextReturn` (bottom), and $\hat{d}$ chosen to be ED (left) and EMD (right). The above results all indicate poor f and $\hat{d}$ combinations that cannot be used for INTAGS (or ATMS) training.

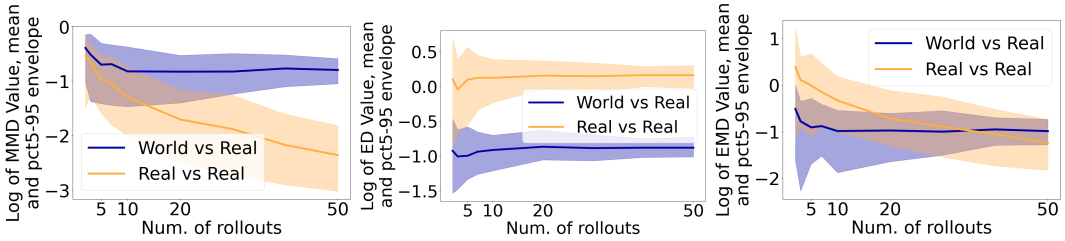

Figure 12: The mean and $5-95\%$ envelop trajectory (over 50 bootstrap trials) against increasing number of rollouts of our proposed market distance metric with feedback `Mkt2Reward`, and $\hat{d}$ chosen to be MMD (left), ED (middle) and EMD (right). Similarly, the above results all indicate poor f and $\hat{d}$ combinations that cannot be used for INTAGS (or ATMS) training.

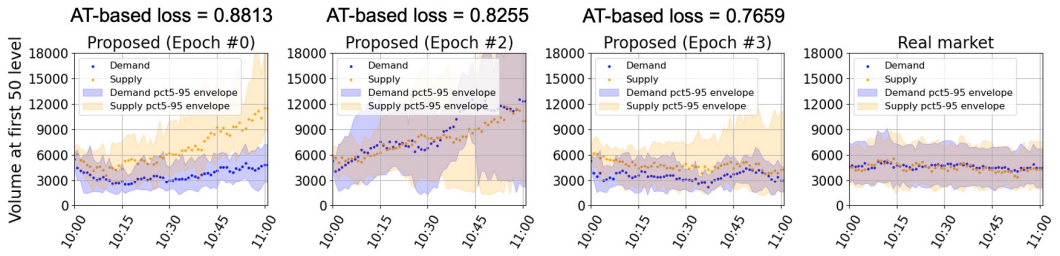

Figure 13: Effectiveness evaluation of our proposed ATMS with different random seeds and training hyperparameters. The consistent pattern across experiments reaffirms the robustness of ATMS in generating realistic market data.

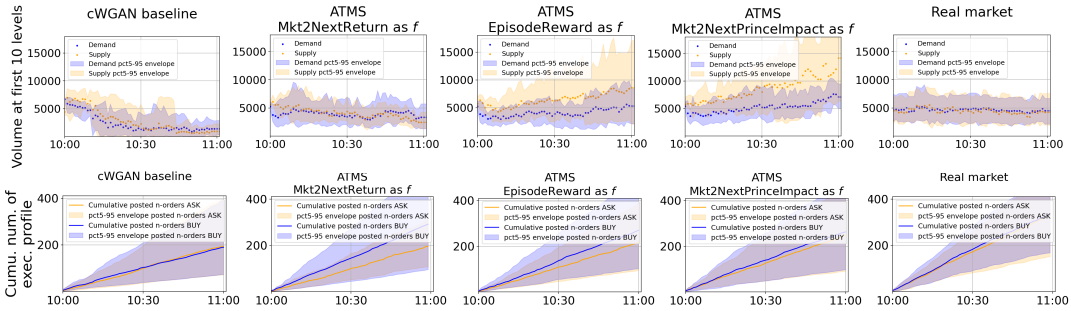

Figure 14: **Comparison of stylized fact time series between our ATMS and cWGAN baseline. The top panel in the second column is exactly the third panel in Figure 4, where ATMS optimizes the resemblance of `Mkt2NextReward` to reality and generates the most realistic volume at first 10-levels in terms of supply-demand match and the magnitude. Meanwhile, optimizing the resemblance of `Mkt2NextPriceImpact` (the fourth column) leads to improved match of cumulative number of executed profile to reality.**

provide a more fair metric and avoid such biases. Thus we could mitigate the weakness shown in Coletta et al. [2023a], where a fairly simple, yet effective, strategy can exploit such biases, and turn the BUY/SELL unbalance into a profit. Indeed, as illustrated in Figure 1, unlike classic local loss that directly penalizes the world BG agent's actions $A_t^{(j)}$'s for given inputs $S_t^{(j)}$'s, our proposed metric penalizes the AT agent input s_t for given previous step's AT agent action a_t. As a result, by considering that AT agent input (or state) s contains numerous stylized facts characterizing the market, ATMS that minimizes our metric can produce stylized facts much closer to reality compared to the cWGAN baseline.

Specifically, we report the results of our proposed ATMS with different feedback choices in Figure 14. It is interesting to observe that using different f's leads to improved similarity to reality for different stylized facts: From the first row of Figure 14, we can see ATMS with `Mkt2NextReturn` as f (second column in the figure) yields volume at first 10-levels most similar to reality in the sense that the supply and demand match with each other just like the real market, and their magnitudes are the most more similar to reality; Indeed, using `Mkt2NextReturn` as f when training ATMS can lead to the best match of volume at first n-levels in the LOB ($n \in \{1, 5\}$) to reality as evidenced in Figure 15. Although ATMS with `Mkt2NextPriceImpact` as f yields very different volume at first 10-levels from reality, it does achieve improved similarity compared to the initialization (see the first panel in Figure 4) in terms of matching supply-demand. From the second row of Figure 14, we can observe that ATMS with `Mkt2NextPriceImpact` as f (fourth column in the figure) can yield the best cumulative number of executed profile: The improvement compared to `Mkt2NextReturn` is evident, whose BUY is significantly larger than SELL; Compared to cWGAN baseline, it correctly captures not only the magnitudes of SELL and BUY but also the pattern that BUY is slightly larger than SELL. In addition, the third column in Figure 14 serves as "direct evidence" that `EpisodeReward` is not appropriate for training ATMS, reaffirming our previous claims. For completeness, we also report results for other stylized facts (e.g., spread) in Figures 16

and 17, from which we can observe that there is still a gap between simulated and real markets, and neither our ATMS nor cWGAN can generate market data with those stylized facts similar to reality. All of the above results show that our ATMS with properly chosen feedback (or rather, the application of our INTAGS to trading markets) can generate much more realistic data compared to the cWGAN baseline.

Lastly, we report results of volume at first n-levels in the LOB for $n \in \{1, 5\}$ in Figure 15, from which we can observe: our proposed approach is behaving similarly to cWGAN baseline for $n = 1$ case, and they both do not fully capture the real market dynamics; For $n = 5$ case, the patterns are fairly similar to that of $n = 10$ case shown in Figure 14, reaffirming our claims above.

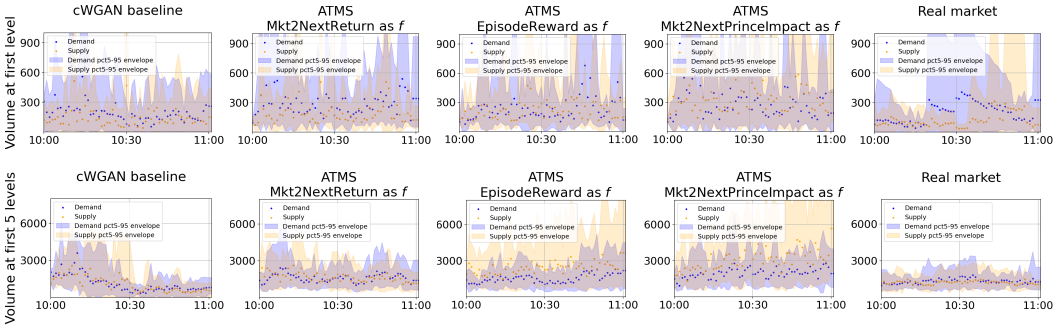

Figure 15: Comparison of volume at first n-levels in the LOB for $n \in \{1, 5\}$ among different simulated markets (specified on top of each panel). We can observe that neither our ATMS nor the cWGAN baseline can correctly capture the volume at the first level whereas our ATMS with `Mkt2NextReturn` can correctly capture the volume at the first 5-levels (just as shown in Figure 14).

Additionally, to demonstrate that our proposed ATMS does not lead to performance degradation by exhibiting significantly different behavior from the real market in terms of other stylized facts, we present results for those additional stylized facts in Figures 16 and 17, from which we can observe that it is evident that the behaviors of both our ATMS and cWGAN baseline are pretty different from reality. Nevertheless, we can observe from Figure 16 that our ATMS have better variance matching to reality compared to the cWGAN baseline for depth and spread, even though all of them do not correctly capture the magnitude.

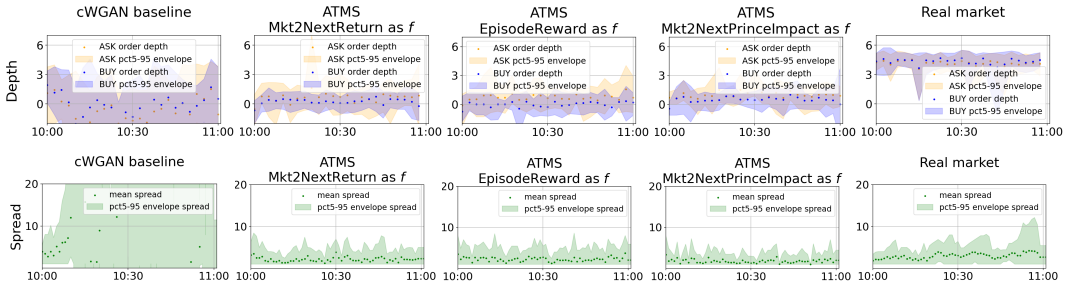

Figure 16: Comparison of additional stylized facts (top: depth; bottom: spread) among different simulated markets (specified on top of each panel). We can observe that our ATMS with various feedback choices can better capture the variance of those stylized facts in real market compared to the cWGAN baseline.

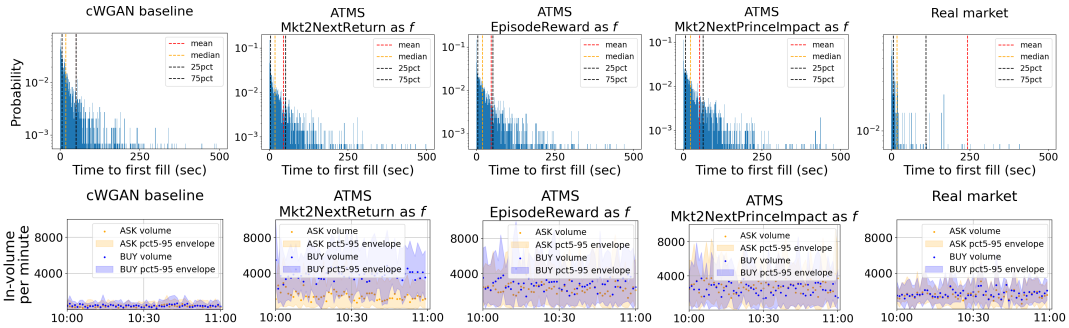

Figure 17: Comparison of other stylized facts (top: time to first fill; bottom: in-volume per minute) among different simulated markets (specified on top of each panel). We can observe that neither our ATMS nor the cWGAN baseline can correctly capture those stylized facts in real market, suggesting future work to further improve the market simulator.