Strategic Participation on Tokenized Platforms: Balancing Investment and Laboring Intensities

Online Supplement

Tianyi Li[1], Xiaoquan (Michael) Zhang *Department of Decisions, Operations and Technology, CUHK*

OS-A: The *Li and Zhang* (2024) platform development model

We introduce the *Li and Zhang* (2024) model for tokenized platforms' development trajectories. At each time step t, the platform's utility $u(t)$ for an average participant consists of three terms with coefficients $\alpha_{1/2/3}$, corresponding to the three roles a participant simultaneously plays on the platform: user, investor, laborer.

$$u(\eta, t) = \alpha_1 u_{user}(\eta) + \alpha_2 u_{investor}(\eta) + \alpha_3 u_{laborer}(\eta). \tag{1}$$

$\eta = \eta(t)$ is participants' population at time t. Three utility terms are modeled respectively:

- u_{user} considers an optimal number of connections (friends) for an individual participant on the platform, η_{ind}^{opt}. Participant's real connection number η_{ind} is linked to the platform's adopter population through $\eta = \eta_{ind}^{l}$ with exponent l, i.e., the degree of separation (**?**). Platform's usage utility increases with the square of the connection number before reaching peak utility, and then gradually decreases with further connections, until ultimately reduced by fraction ϕ. Two logistic functions are used to specify u_{user}. In all, $u_{user} = \frac{h(\eta_{ind}^{opt})^2}{1+q_1 e^{-\eta_{ind}/\eta_0^+}} - \frac{\phi h(\eta_{ind}^{opt})^2}{1+q_2 e^{-(\eta_{ind}-\eta_{ind}^{opt})/\eta_0^-}}$.

- $u_{investor}$ considers that each piece of token investment $P(t)$ invested at t is held t_p periods, so at each time step, the participant collects investment returns from past t_p periods. Price is determined as the inverse ratio of the platform population at the token-investing period to the platform population at the current period, to the power of β. An opportunity cost is considered, via a background period interest rate ϵ. In all, $u_{investor} = \sum_{k=1}^{t_p} P(t-k)[(\frac{\eta(t)}{\eta(t-k)})^{\beta} - (1+\epsilon)^k]$.

- $u_{laborer}$ considers that laboring capacity $L(t)$ generates immediate return during each period (as opposed to delayed return in $u_{investor}$). Unit laboring reward w_m, initially distributed among a small community of core laborers η_{min}, is inversely proportional to platform population $\eta(t)$, and slowly decreases over time, modeled by an exponential decay with constant t_0. A unit laboring cost w_0 is considered with each unit of laboring capacity. In all, $u_{laborer} = L(t)(w_m \frac{\eta_{min}}{\eta(t)} e^{-t/t_0} - w_0)$.

$\zeta(t)$ is the number of non-adopters at time t. The sum of $\eta(t)$ and $\zeta(t)$, constant over time, equals the entire population N. The transition between η and ζ follows the SIS dynamics (e.g., **?**): $\frac{d\zeta}{dt} = -\gamma^+ \frac{\zeta\eta}{N} + \gamma^- \eta$, $\frac{d\eta}{dt} = \gamma^+ \frac{\zeta\eta}{N} - \gamma^- \eta$. Platform adoption rate γ^+ depends on utility increase during the current period, while platform abandon rate γ^- depends on utility reduction compared to the previous level τ periods before. The two effects have sensitivities K^+ and K^-, respectively. Thus we have $\gamma^+ = \left[\log(\frac{MAX(u_t - u_{t-1}, 0)}{\Delta u_0} + 1)\right]^{K^+} \gamma_0$, $\gamma^- = \left[\log(\frac{MAX(u_{t-\tau} - u_t, 0)}{\Delta u_0} + 1)\right]^{K^-} \gamma_0$.

$\alpha_{1/2/3}$, h, η_{ind}^{opt}, $q_{1/2}$, $\eta_0^{+/-}$, ϕ, β, ϵ, w_m, η_{min}, t_0, w_0, Δu_0, γ_0, $K^{+/-}$ are real numbers; two temporal parameters t_p and τ are integers that capture system delay: t_p is average token holding time, and τ is average delay before quitting the platform. t_p and τ differentiate tokenized platforms' development trajectories. Two time series are to be specified in the model, i.e., control inputs of the system: weekly investment intensity $P(t)$ (dollars) and weekly laboring intensity $L(t)$ (hours). For the average participant, constant $P = P_{const}$ and $L = L_{const}$ are considered.

The model describes a three-phase trajectory for tokenized platforms' development (Figure S1): a slow initial

[1] tianyi.li@cuhk.edu.hk

stage where the platform is maintained by core laborers, a rapid-developing intermediate stage where token investment substantially promotes platform adoption and a long-term stage where platform cycles between maintaining a small and a large adopter population. In $u_{investor}$, the platform's development dynamics, in terms of its population η, is linked to token price dynamics through price sensitivity β. This allows the model to match the price series of real tokens. Results show that, while having a relatively small parameter space, the model can satisfactorily fit the price histories of various platform tokens.

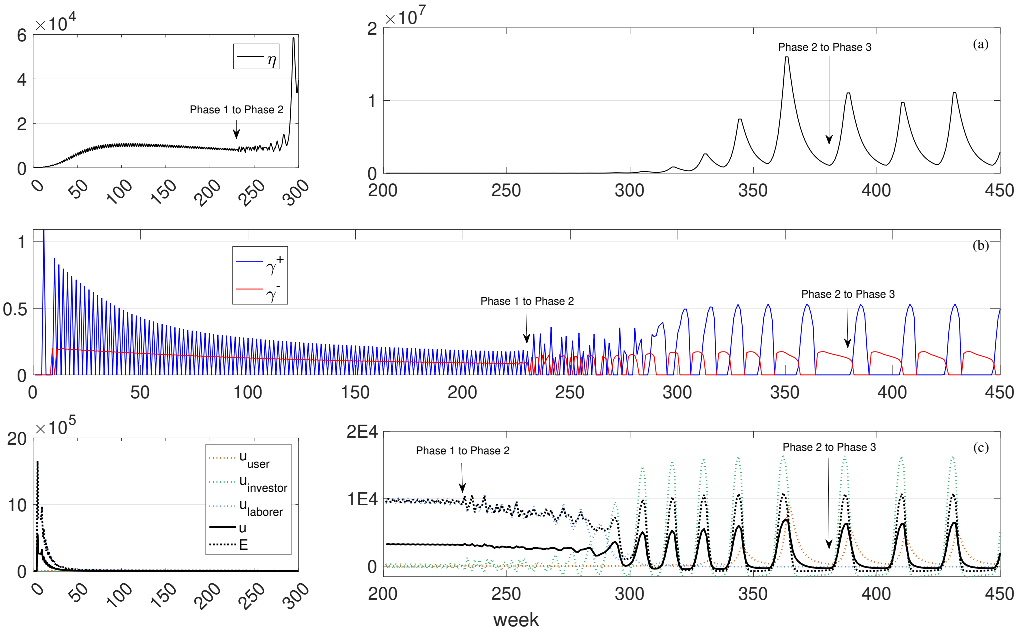


Figure S1: The three-phase platform development trajectory (base case; see model parameters in Table S1). (a) Adopter population η. (b) Adoption rate γ^+ and abandon rate γ^-. (c) Utility terms u_{user}, $u_{investor}$, $u_{laborer}$, overall utility u, and period earning E. Weeks 200–300 of (a) and (c) are shown at different scales on different subplots.

Parameter	Unit	Type	Base-line Value
Coefficients			
α_1	-	platform-specific	1/3
α_2	-	platform-specific	1/3
α_3	-	platform-specific	1/3
Utility u_{user}			
h	\$/people^2	platform-specific	0.1
ϕ	-	constant	0.3
q_1	-	constant	1E3
q_2	-	constant	1E2
η_{ind}^{opt}	people	constant	500
η_0^+	people	constant	40
η_0^-	people	constant	300
l	-	constant	3
Utility $u_{investor}$			
t_p	week	platform-specific	4
P_{const}	\$	constant	1000
β	-	sensitivity	2
ϵ	-	constant	2E-3
Utility $u_{laborer}$			
L_{const}	hour	constant	168
w_m	\$/hour	platform-specific	1E4
η_{min}	people	constant	100
t_0	week	constant	300
w_0	\$/hour	constant	0.5
Adoption			
τ	week	platform-specific	4
γ_0	1/week	constant	0.1
K^+	-	sensitivity	1
K^-	-	sensitivity	0.3
Δu_0	\$	constant	10
N	people	constant	1E9

Table S1: Base-line values of platform parameters. Platform-specific parameters and sensitivity parameters are estimated from data.

OS-B: Model dynamics

Base-line dynamics is used as the background (using parameter values in OS-A). We first show the separate effects of $t_{p,ind}$, P_{ind}^{ext}, L_{ind}, v_P, and v_L on platform earning. We then investigate the effects of (i) the fraction of earning from investment and laboring spent in re-investment, v_P and v_L, and (ii) individual participant's token-holding time $t_{p,ind}$, on period earning $E_{ave/ind}$ and accumulated earning $\boldsymbol{E}_{ave/ind}$, considering different individual external investment P_{ind}^{ext} and laboring intensity L_{ind}.

Effects of $t_{p,ind}$, P_{ind}^{ext} and L_{ind}

We show the effects of $t_{p,ind}$, P_{ind}^{ext}, and L_{ind} on period earning E_{ind} (Figure S2). When earning is positive at most periods, as in the base-line trajectory (Figure S2a), the individual participant is able to increase its earning by holding the token for more periods than platform-average, i.e., $t_{p,ind} > t_{p,ave} = 4$. A faster turnaround $t_{p,ind} < t_{p,ave}$ is less profitable in this favorable market scenario but may be beneficial at other platform development trajectories.

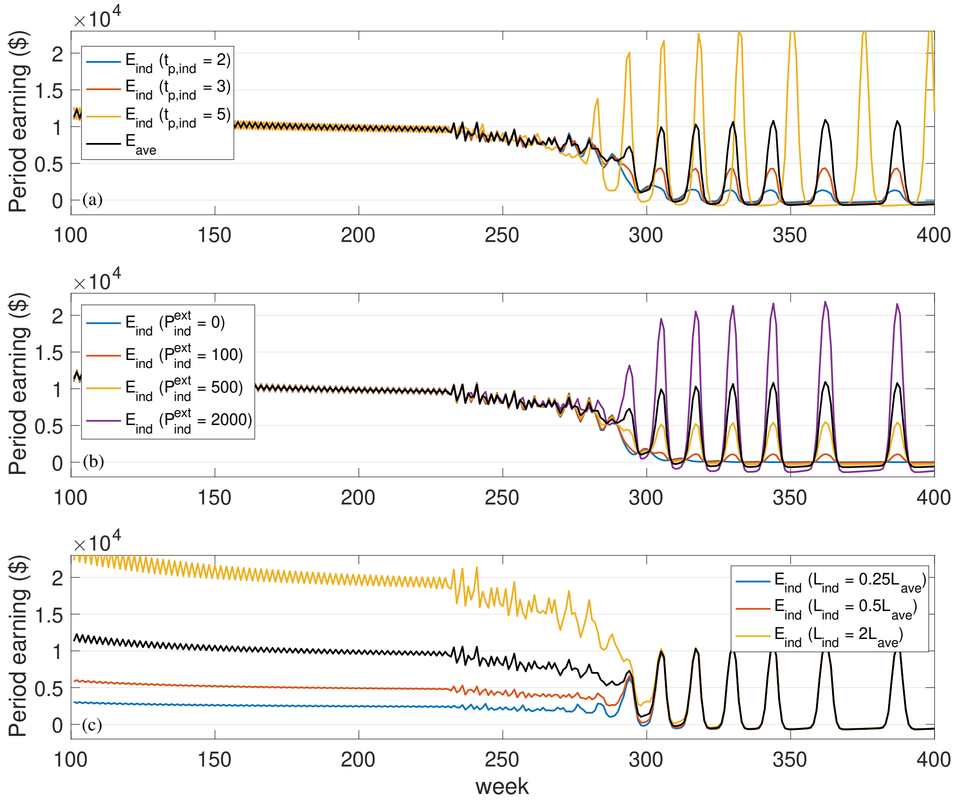


Figure S2: Effects of (a) individual token-holding time $t_{p,ind}$, (b) individual external investment intensity P_{ind}^{ext}, and (c) individual laboring intensity L_{ind}, on individual period earning E_{ind}. Platform-average period earning E_{ave} (black curve) is the same on three subplots.

During the first phase of platform development, participants get earnings mostly through laboring. An individual participant earns as much as the platform-average participant even when it invests less than the platform-average, $P_{ind}^{ext} < P_{ave}^{ext} = 1000$, and in particular, even when it makes zero external investment $P_{ind}^{ext} = 0$ (Figure S1b). But it can earn more by conducting more laboring on the platform than platform-average, $L_{ind}/L_{ave} > 1$, and will earn less if it conducts less laboring, $L_{ind}/L_{ave} < 1$ (Figure S2c). During the second phase of platform development and on, participants get earnings mostly through investment, and laboring becomes less important. This corresponds to the real-world situation where purchasing more laboring hardware may no longer pay off, after the laborer population is saturated, and after the platform enters the rapid development stage driven by large-scale token investment.

Effects of v_P and v_L

We show the effects of v_P and v_L on period earning E_{ind} (Figure S3). Non-zero re-investment fraction v_P does not have much impact on earning before the second development stage (Figure S3a-S3d), after which more re-investment (i.e., larger v_P) leads to more earning in this favorable trajectory (Figure S3e-S3h). Re-investing the laboring reward (i.e., non-zero v_L) is always profitable throughout platform's development; this profit is more pronounced during platform's initial stage where laboring return is high (Figure S3i-S3l).

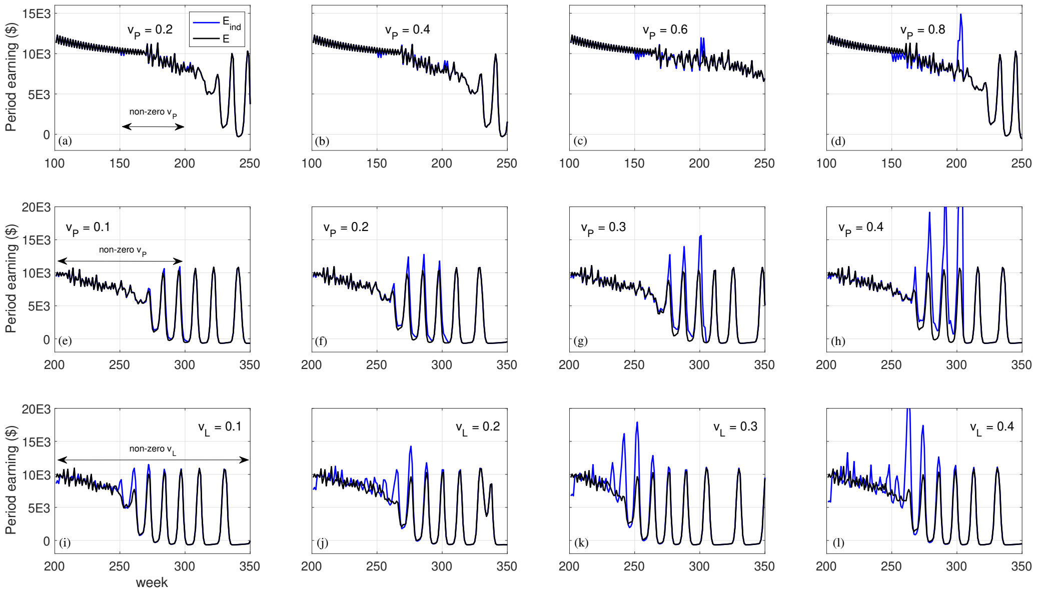


Figure S3: Effects of the fraction of period earning (a–h) from investment v_P and (i–l) from laboring v_L, spent in re-investment, on individual period earning E_{ind}, comparing to platform-average earning E_{ave}. The non-zero v_P periods and non-zero v_L periods (within double-end arrows) are the same in each row.

Impact of individual's investment intensity on platform state

We study the impact of P_{ind} on platform state (the impact of laboring intensity L_{ind} on platform state is similar). Consider linear impact function $\Delta P^{ave} = \delta_P \Delta P^{ind}$, with slope $\delta_P = 1E-2, 1E-5, 1E-8, 1E-12$. At each scenario of δ_P, the individual participant has non-zero re-investment fractions $v_P = v_L = 0.1$ from Week 150 to Week 300, resulting in individual earning E_{ind} deviating from E_{ave} (Figure S4). As discussed in *Li and Zhang* (2024), the model reproduces the chaotic feature of real-world investment markets; this feature is further demonstrated in the current analysis: even with very small impact, $\delta_P = 1E-12$, corresponding to an individual participant having one over one trillion of the market share, period earning $E_{ave/ind}$ is still sensitive to the perturbation induced by individual participants' investment (Figure S4a). Nevertheless, the accumulated earning $\boldsymbol{E}_{ave/ind}$ is robust to market perturbations (Figure S4b), even with a large impact coefficient $\delta_P = 1E-2$. Perturbations in the market environment mostly alter the phase of platform development and participant's earning trajectories, while not the cumulative profit.

Model dynamics for organizational participants

Organizational participants with considerable market standings can realize large investment and laboring intensities. The effects are shown in Figure S5. Results suggest consistent model behavior: with large market standing δ_P, δ_L (comparing the values of 1e-1/1e-2/1e-4/1e-8, i.e., 10%/1%/0.01%/1e-6% market share) or large investment and laboring intensities ($\delta_{H,ind} = 0.03$ vs. 0.01, i.e., the augmentation of laboring capacity accelerated; $L_{ind} = 168 \times 3$ vs. 168, i.e., the weekly laboring capacity expanded; $P_{ind} = 2000/5000$ vs. 1000, i.e., the weekly investment intensity elevated), the organizational participant's earning E_{ind} (solid color lines in Figure S5) is greater than the platform-average E_{ave} (dashed lines).

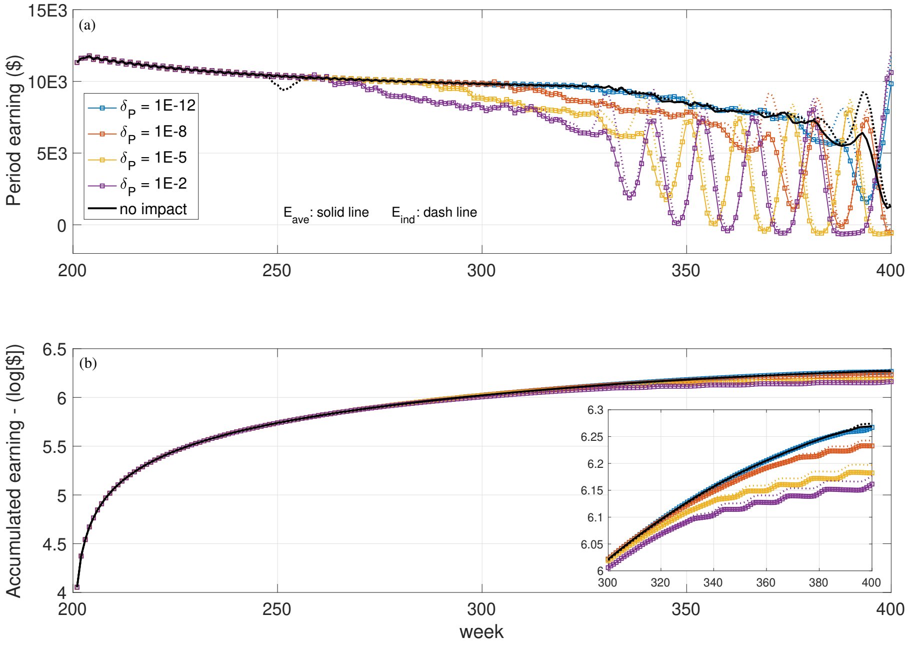


Figure S4: Impact of individual's investment intensity on platform state. Linear impact $\Delta P^{ave} = \delta_P \Delta P^{ind}$ with δ_P = 1e-2/1e-5/1e-8/1e-12. In each scenario, the participant has non-zero re-investment $v_P = v_L = 0.1$ from Weeks 150-300, resulting in its earning E_{ind} deviating from E_{ave}. Compared to (a) period earning $E_{ave/ind}$, the (b) accumulated earning $\boldsymbol{E}_{ave/ind}$ (in the logarithmic scale) is robust to market perturbations.

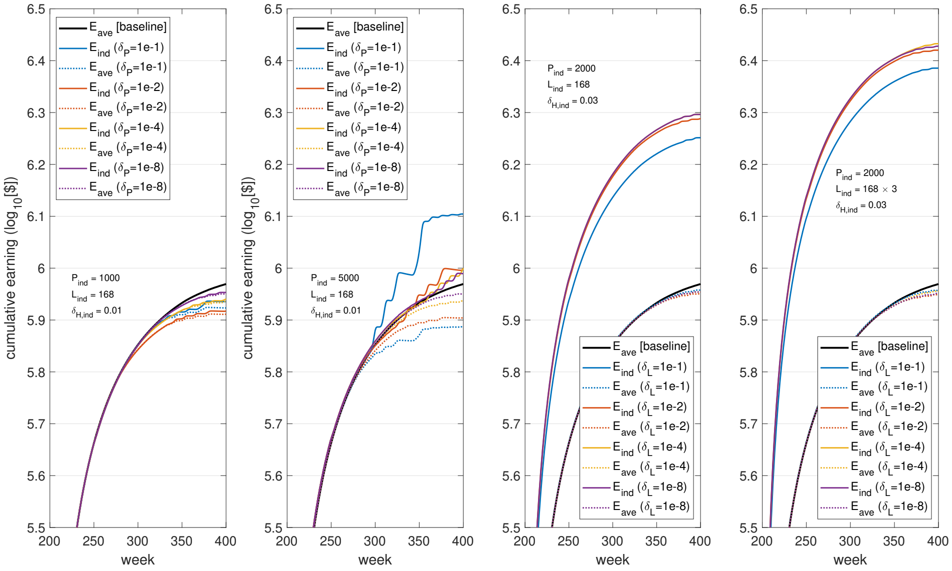


Figure S5: Model dynamics for organizational participants. Show the effects of P_{ind}, L_{ind}, and $\delta_{H,ind}$. Market standing δ_P, δ_L = 1e-1/1e-2/1e-4/1e-8; augmentation of laboring capacity $\delta_{H,ind} = 0.03$ vs. 0.01; weekly laboring capacity $L_{ind} = 168 \times 3$ vs. 168; weekly investment intensity P_{ind} = 2000/5000 vs. 1000. With large market standing or large investment/laboring intensities P_{ind}, L_{ind}, and $\delta_{H,ind}$, the participant's earning E_{ind} (solid color lines) is greater than the platform-average E_{ave} (dashed lines).

OS-C: Strategic decision on investment intensity: fixed amount execution

We analyze fixed amount execution at the decision on investment intensity. Consider that the participant starts with zero investment; at each period t, the action a_P to be made is invest a fixed amount \$100 ($a_P = 1, P_{ind}(t) = 100$), or not invest ($a_P = 0, P_{ind}(t) = 0$). Similar to fixed increment execution, we consider T_D historical values of Y when making the investment decision.

The following strategies are considered, labeled with prefix *SP**:

- *SP* 1 – Instant average.* Invest when the average projected earning is greater than last period by g_{P+}, and is non-negative: $a_P = 1$ when $\bar{y}_{T+1} > g_{P+}\bar{y}_T \geq 0$, otherwise $a_P = 0$.
- *SP* 2 – Historical average.* Invest when the average projected earning is greater than past-T_D period average by g_{P+}, and is non-negative: $a_P = 1$ when $\bar{y}_{T+1} > g_{P+} \sum_{t=[T-T_D]} \bar{y}_t / T_D \geq 0$, otherwise $a_P = 0$.
- *SP* 3 – Aggressive.* Invest when the level-Q PI upper bound of projected earning is greater than last period by g_{P+}, and is non-negative: $a_P = 1$ when $\lfloor y_{T+1} \rfloor_Q > g_{P+}\bar{y}_T \geq 0$, otherwise $a_P = 0$.
- *SP* 4 – Conservative.* Invest when the level-Q PI lower bound of projected earning is greater than past-T_D period average by g_{P+}, and is non-negative: $a_P = 1$ when $\lceil y_{T+1} \rceil_Q > g_{P+} \sum_{t=[T-T_D]} \bar{y}_t / T_D \geq 0$, otherwise $a_P = 0$.
- *SP* 5 – Instant fuzzy majority.* Invest when the fraction of projected earning that is greater than g_{P+} times the last-period value, and is non-negative, is above a threshold: $a_P = 1$ when $p_{y_{T+1}}^{F,+g_{P+}} > p_0^{F,+g_{P+}}$, otherwise $a_P = 0$.
- *SP* 6 – Historical fuzzy majority.* Invest when the fraction of projected earning that is greater than g_{P+} times the past-T_D period average, and is non-negative, is above a threshold: $a_P = 1$ when $p_{y_{T+1}}^{F,+g_{P+},T_D} > p_0^{F,+g_{P+},T_D}$, otherwise $a_P = 0$.

We consider period earning E_P or price X as the property Y. We evaluate strategies' performance by looking at the ground-truth investment earning accumulated in T_H periods: $\hat{\boldsymbol{E_P}} = \sum_{t=[T+1,T+T_H]} \epsilon_T^{t-(T+1)} \hat{E_P}(t)$. Compare to the following model-free strategies based on the trending of X:

- *SP*-a – Average price.* Invest when the average value of X over the past T_D periods is greater than last period by g_{P+}, and is non-negative: $a_P = 1$ when $\sum_{t=[T-T_D+1,T+1]} X_t > g_{P+} \sum_{t=[T-T_D,T]} X_t \geq 0$, otherwise $a_P = 0$.
- *SP*-b – Mean reversion.* Invest when the current-period X is greater than the average value of past T_D periods by g_{P+}, and is non-negative: $a_P = 1$ when $X_{T+1} > g_{P+} \sum_{t=[T-T_D,T]} X_T / T_D \geq 0$, otherwise $a_P = 0$.
- *SP*-c – Price momentum.* Invest when the current-period X is greater than the value T_D periods ago by g_{P+}, and is non-negative: $a_P = 1$ when $X_{T+1} > g_{P+} X_{T-T_D} \geq 0$, otherwise $a_P = 0$.

OS-D: Data information

Dataset statistics

No. of cleaned data series	**112**		
	MAX	**MIN**	**AVE(STD)**
Start date	Oct 12, 2018	Apr 28, 2013	Nov 17, 2016
End date	Mar 11, 2019	Dec 12, 2017	Oct 28, 2018
Series length (day)	2144	149	706(488.9)
Maximum price	$19475.8	$0.145	$357.2(1929.8)
Minimum price	$77.4	$3E-6	$3.25(12.4)
Price variance	1.1E+7	1.1E-3	1.1E+5(1.07E+6)
Maximum daily return	1147%	20%	151%(1.92)
Minimum daily return	-16%	-98%	-43%(0.19)
Overall return (since start date)	1.52E+5	-0.99	1431(1.44E+4)
Maximum return (since start date)	3.42E+5	0	3.43E+3(3.23E+4)
Minimum return (since start date)	0	-0.99	-0.60(0.312)
No. of series $>$ 100 weeks	**38**		
No. of series $>$ 200 weeks	**12**		

Table S2: Dataset statistics. MAX/MIN/AVE/STD: maximum/minimum/average/standard deviation.

Primary data series used in Section 8 (metric-based decision-making)

Length (week)	72
Maximum price	5E2
Minimum price	1E1
Price variance	9E3
Maximum daily return	98%
Minimum daily return	-71%
Overall return (since start date)	-97%
Maximum return (since start date)	4%
Minimum return (since start date)	-98%

Table S3: Statistics of the data series used in Section 8 (metric-based decision-making).

Primary data series used in Section 9 (RL-based decision-making)

Length (week)	126
Maximum price	1E2
Minimum price	3E-1
Price variance	5E2
Maximum daily return	148%
Minimum daily return	-56%
Overall return (since start date)	15%
Maximum return (since start date)	385%
Minimum return (since start date)	-24%

Table S4: Statistics of the data series used in Section 9 (RL-based decision-making).

OS-E: Implementation details

Estimation

We use Matlab's *fminsearch* optimizer to estimate non-integer parameters.

Maximum iteration and maximum function evaluation are both 200 times the number of estimated parameters. Tolerance of difference in misfit and tolerance of difference in parameters are both 1e-6.

Initial search points for non-integer parameters $[\alpha_1,\ \alpha_2,\ \alpha_3,\ \beta,\ K^+,\ K^-,\ h,\ w_m]$:
$[0.33, 0.33, 0.33, 2, 1, 0.3, 0.1, 1e4]$, $[0.33, 0.33, 0.33, 1, 1, 0.3, 0.1, 1e4]$, $[0.33, 0.33, 0.33, 2, 0.5, 0.5, 0.1, 1e4]$.

Length of forward series used in estimation: $t_{max} = MAX(500, 3|X|)$.

Minimum estimation length $T_{E0} = 12$ weeks.

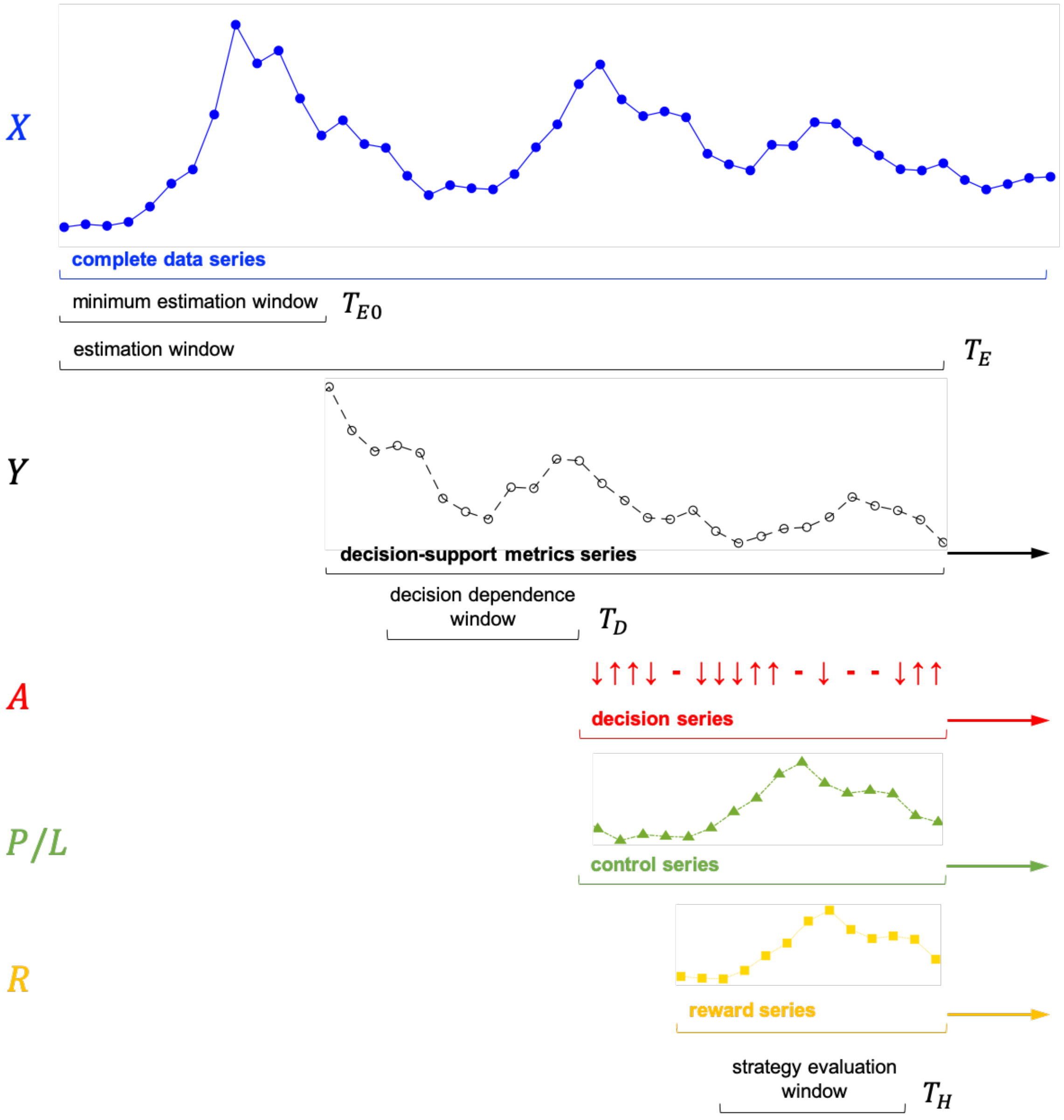


Figure S6: Illustration of time series and time windows in the metric-based decision-making.

Reinforcement learning

We use Matlab's *Reinforcement Learning Toolbox* to build codes. We conduct pre-training to determine appropriate stopping criteria and discover potential local optima. We choose *average episode reward* (over 10 episodes) as the stopping criterion. The appropriate stopping value varies across data series and estimation windows. 500 epochs are initiated in each learning session, and 500 episodes are initiated during each epoch of training. In each episode, the agent interacts with the environment for $T_{RL} = 10$ consecutive periods.

We select the best training epochs from the 500 epochs, according to three criteria: (1) training stops at the stopping criterion, i.e., reach the threshold for average episode reward, (2) the stopped epoch experiences sufficient training episodes; epochs stopped before 20 episodes are dropped, (3) the trained agent is tested on random states; agents producing uniform actions at random states signal local optima and are dropped. Hyperparameters are tuned to restrict the acceptance of training epochs; the training session is optimal when only one epoch passes all three criteria (which is the case for both the DQN and the PPO agent reported in the results). To speed up the tuning of hyper parameters, we stop the 500-epoch session early if no epoch satisfies two of the three criteria after 50 epochs.

The current environment encodes a large state space $\boldsymbol{S}$ and a small action space $\boldsymbol{A}$ and the training is prone to get trapped in local optima. Some epochs fall rapidly to all-zero or all-decreasing actions. Unlike in many RL contexts where fast convergence is desired, to slow down the learning, we have the following treatments:

For DQN agent: A slower decay of the exploration factor ϵ is welcomed. Large ϵ helps the agent better explore the action space. A large mini-batch size and large overall size of the experience buffer are helpful. Storing more experiences avoids rapid convergence resulting from experience replay.

For PPO agent: Similar to DQN, a relatively large mini-batch size and a large experience horizon are helpful. Increasing the entropy loss weight promotes exploration and can help the agent move out of local optima; increasing the clip factor has a similar effect as the range of policy updates can be enlarged. Very large entropy loss weight or clip factor can nevertheless have negative effects.

For both agents: Increasing the stochasticity of the environment slows down convergence. Incorporating multiple parameter sets from the MC estimation (i.e., considering *parameter uncertainty*), accounting for stochasticity in model dynamics via increased ϵ_{sys} (i.e., considering *system uncertainty*), and augmenting participant-platform interaction via elevated δ_P and δ_L (i.e., considering *input uncertainty*), facilitate more detailed learning. Adding hidden units to the neural network does not mitigate the convergence, as the learning architecture needs to be commensurate with the complexity of the environment. Average episode reward is an appropriate stopping criterion, while the threshold should be neither too high nor too low (a high threshold may favor all-zero/all-decreasing actions).

DQN agent. Double DQN is activated; during experience replay, the experience buffer size is tested at 1e4/1e5/1e6/1e7, mini-batch size is tested at 64/128/256. For ϵ-greedy exploration of the action space, ϵ starts at 1 and limits at 0.01, with the decay rate tested at 0.001/0.005/0.01. The reward discount factor is 0.99. The Adam (adaptive movement estimation) optimizer is used, at default learning rate 0.01. A standard 7-layer neural network is used as the critic: input layer, fully-connected layer, two times of ReLU plus fully-connected layer, and final output layer. The number of hidden units in each layer is 256. As the state vector concatenates different variables, a pre-processing RNN layer (LTSM) is not used.

PPO agent. Experience horizon is tested at several times of 512. Mini-batch size is tested at several times of 128. Large values for clip factor (e.g., 0.9) and entropy loss weight (e.g., 0.7) encourage the exploration of the action space. A generalized advantage estimator (i.e., gae) is used as the advantage estimate method, with a default smoothing factor of 0.95; the advantage function is not normalized. The reward discount factor is 0.99. The Adam optimizer with a default learning rate of 0.01 is used in both actor and critic optimization. The critic network is the same as the DQN agent (7-layer standard feed-forward network). The actor network (8-layer standard feed-forward network) has an extra softmax layer appended to the last fully connected layer. For both actor and critic networks, the number of hidden units in each layer is 256.

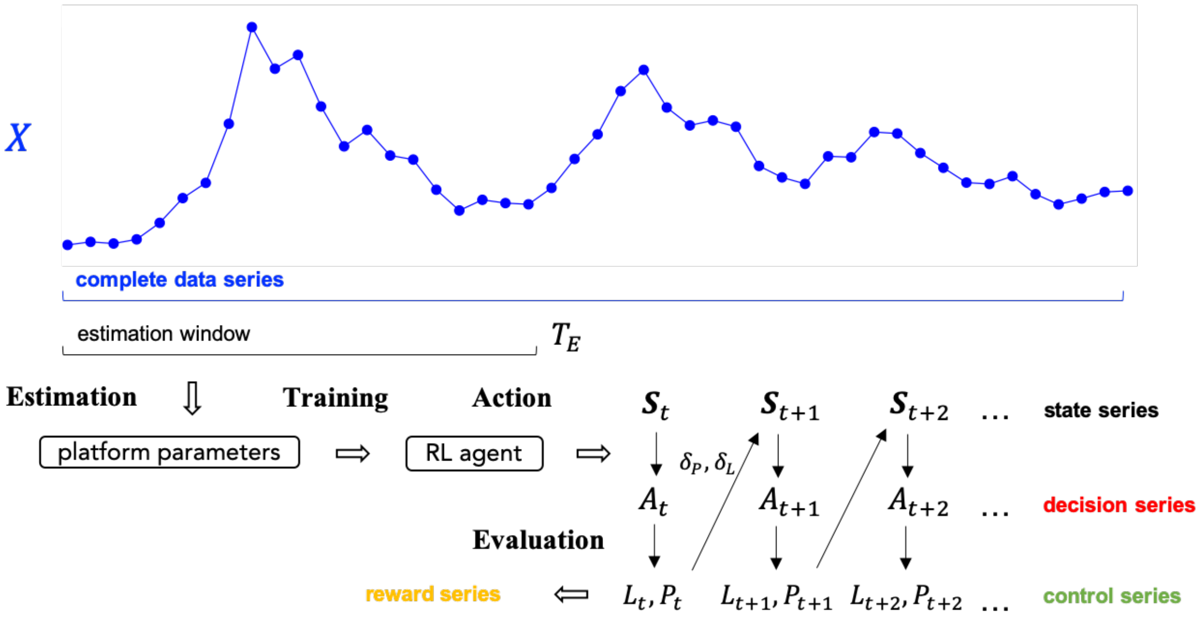


Figure S7: Illustration of the procedures for the RL-based decision-making.

OS-F: Uncertainties in model projection

System uncertainty. We consider stochastic model dynamics, i.e., $\epsilon_{sys,1/2+/2-} \neq 0$, but only best-fit platform parameters, i.e., $\epsilon_{misf} = \epsilon_{misf,global} = 0$. Consider $t_{p,ind} = 6 > t_{p,ave} = 4$. We show two stochastic levels: $\epsilon_{sys,1/2+/2-} = 0.2, 0.8$; $H_{sto} = 50$ MC runs are generated at each estimation window. Results agree with the expectation (Figure S8). For both E_P and E_L, as the stochastic level increases from 0.2 to 0.8, PI gets wider, while the projected mean remains largely invariant. Investment in this token can be either profitable ($\bar{E_P} > 0$) or not ($\bar{E_P} < 0$), while laboring on this platform is always profitable ($\lfloor E_L \rfloor_{0.95} > 0$).

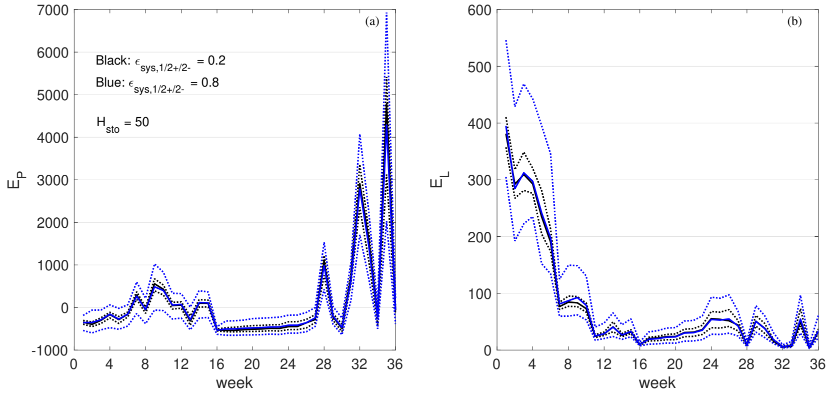


Figure S8: Demonstration of system uncertainty. Showing the mean (solid lines) and 95% PI (dashed lines on both sides) of projected (a) E_P and (b) E_L. $H_{sto} = 50$ MC runs are generated at each estimation window.

Parameter uncertainty. We consider deterministic model dynamics, $\epsilon_{sys,1/2+/2-} = 0$, while an accepted pool for platform parameters. Three initial search points ($H_{ini} = 3$) for non-integer parameters are used to explore the optimization space. Two acceptance levels on misfits are considered: for each initial search point, $\epsilon_{misf} = 0.1, 0.3$; during the re-pooling of results across multiple initial search points, $\epsilon_{misf,global} = 0.2, 0.5$. MC acceptance criteria (ϵ_{misf} and $\epsilon_{misf,global}$) concern the percentage difference between the candidate misfit and the best (i.e., smallest) misfit; instances having a smaller percentage difference than the threshold are accepted. Consider $t_{p,ind} = t_{p,ave} = 4$. Results suggest that as more parameter sets are accepted into the MC ensemble (i.e., ϵ_{misf} and $\epsilon_{misf,global}$ get larger), model projection embodies greater variance (Figure S9). Same as Figure S8, investment can be either profitable or not, while laboring is always profitable. Model projection on E_P and E_L in Figure 3 aggregates features of Figure S8 and S9.

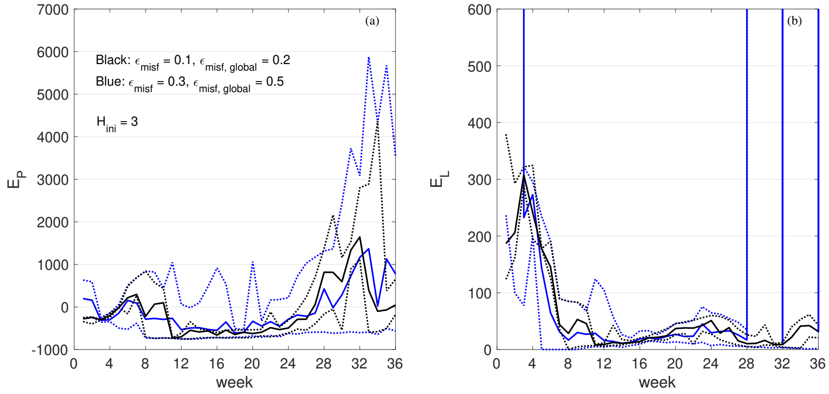


Figure S9: Demonstration of parameter uncertainty. Showing the mean (solid lines) and 95% PI (dashed lines on both sides) of projected (a) E_P and (b) E_L. $H_{ini} = 3$ initial search points are used. Certain portions of E_L projection at the larger ϵ_{misf} and $\epsilon_{misf,global}$ lie outside the plotted area.

OS-G: Strategy parameters

Parameters of laboring strategies (*SL**) are experimented with (Figure S10). Four cases are shown in each plot. Invariant parameter values on each subplot are (a) $g_L = 1$, (b) $T_D = 2$, (f) $p_0^{F,+g_L} = 0.6$, (g) $g_L = 0.01$.

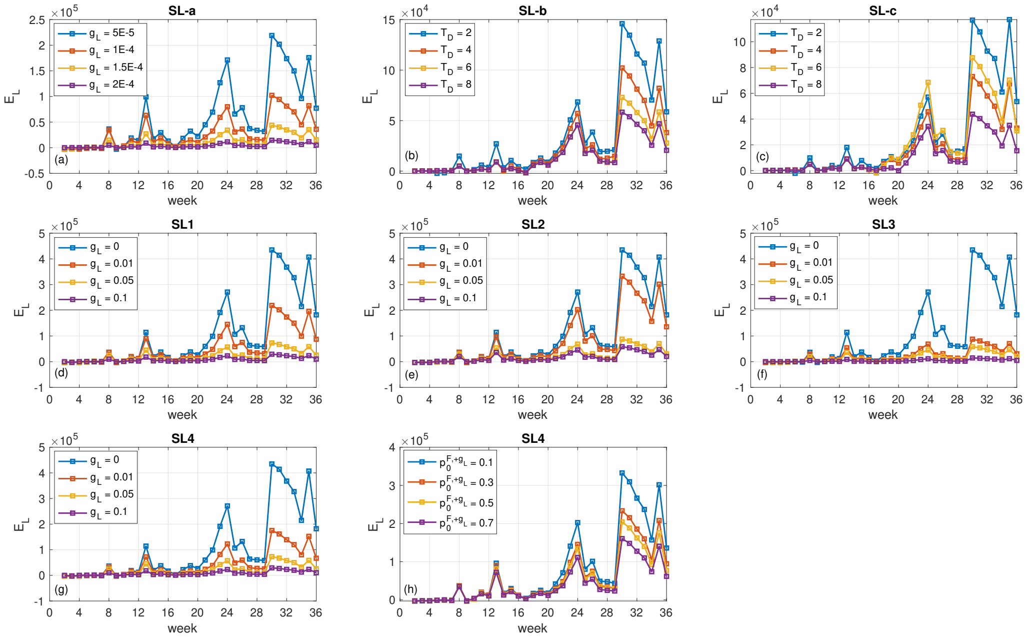


Figure S10: Parameters for strategies on laboring intensity (*SL-a/b/c*, *SL1* - *SL4*).

Parameters of investment strategies (*SP**, E_P-dependent) are experimented with. Four cases are shown in each plot. Invariant parameter values on each plot are:

Figure S11: (a) $g_{P+} = 1$, $g_{P-} = 0.8$, (b) $T_D = 6$, $g_{P-} = 0.8$, (c) $T_D = 2$, $g_{P+} = 1.5$, (d) $g_{P-} = 0.5$, (e) $g_{P+} = 1.5$, (f) $g_{P+} = 1$, $g_{P-} = 0.8$, (g) $T_D = 6$, $g_{P-} = 0.6$, (h) $T_D = 6$, $g_{P+} = 1.5$, (i) $g_{P+} = 1$, $g_{P-} = 0.2$, (j) $T_D = 3$, $g_{P-} = 0.6$, (k) $T_D = 5$, $g_{P+} = 1.5$, (l) $g_{P+} = 2$, $g_{P-} = 0.2$, (m) $T_D = 4$, $g_{P-} = 0.8$, (n) $T_D = 2$, $g_{P+} = 1.5$.

Figure S12: (a) $g_{P-} = 0.8$, $p_0^{F,+P+/-P-} = 0.8/0.3$, (b) $g_{P+} = 1.5$, $p_0^{F,+P+/-P-} = 0.8/0.3$, (c) $g_{P+/P-} = 1.5/0.8$, $p_0^{F,-P-} = 0.3$, (d) $g_{P+/P-} = 1.5/0.8$, $p_0^{F,+P+} = 0.8$, (e) $g_{P-} = 0.8$, $p_0^{F,+P+/-P-} = 0.8/0.3$, $T_D = 3$, (f) $g_{P+} = 1.5$, $p_0^{F,+P+/-P-} = 0.8/0.3$, $T_D = 6$, (g) $g_{P+/P-} = 1.5/0.8$, $p_0^{F,-P-} = 0.3$, $T_D = 6$, (h) $g_{P+/P-} = 1.5/0.8$, $p_0^{F,+P+} = 0.8$, $T_D = 6$, (i) $g_{P+/P-} = 1.5/0.8$, $p_0^{F,+P+/-P-} = 0.8/0.3$.

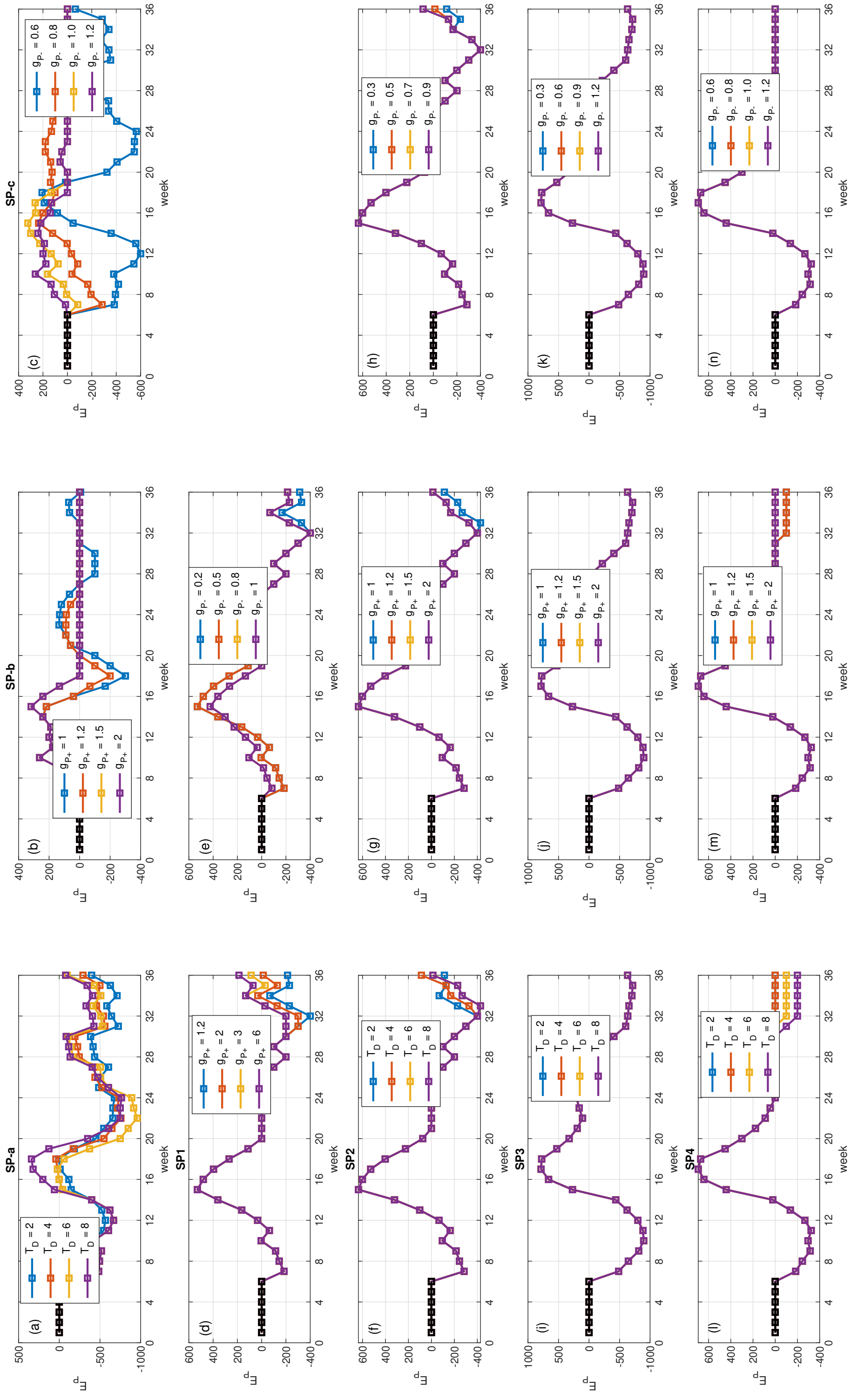


Figure S11: Parameters for strategies on investment intensity (*SP-a/b/c*, *SP1* - *SP4*).

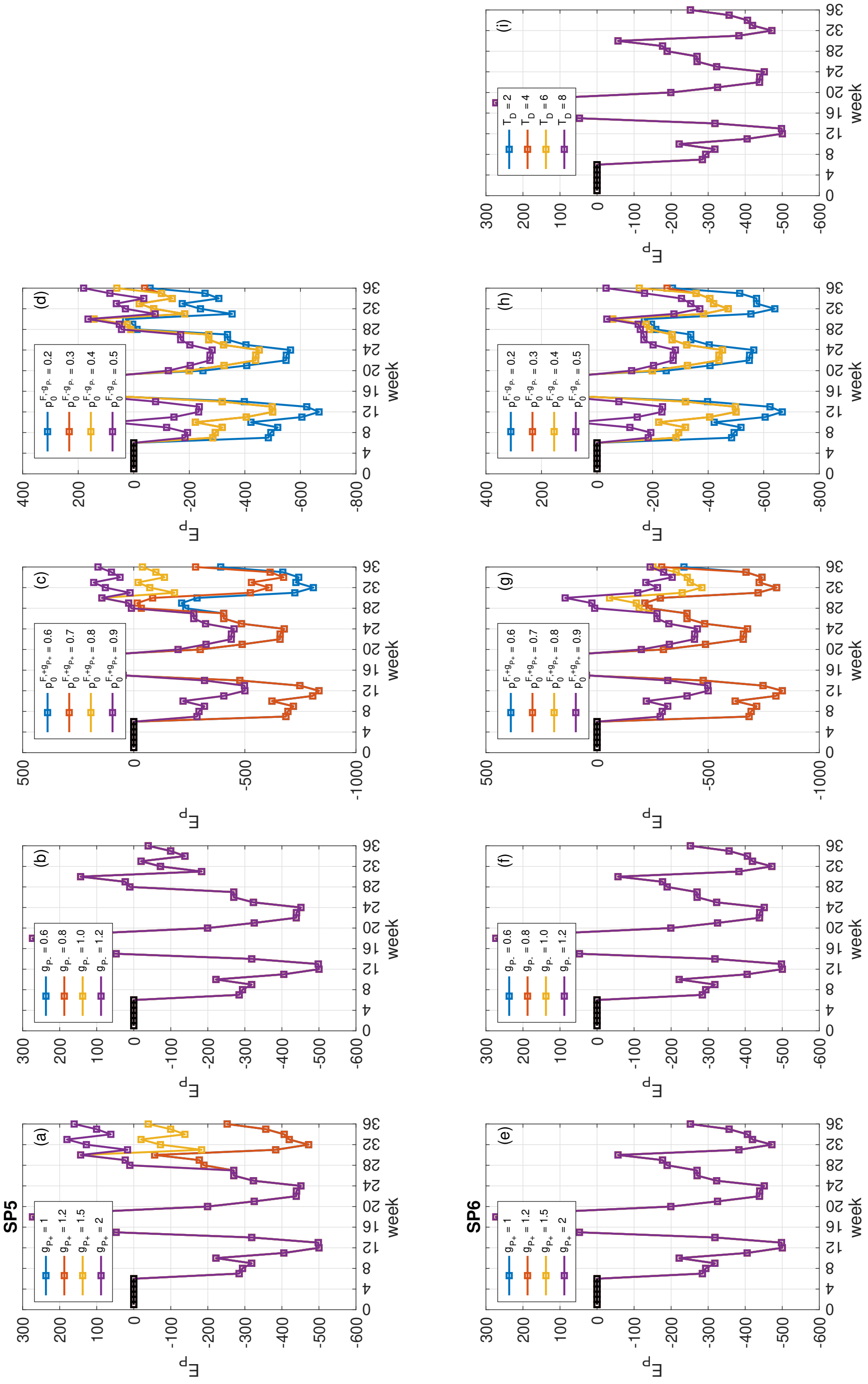


Figure S12: Parameters for strategies on investment intensity (*SP5* - *SP6*).

OS-H: Robustness checks at metric-based decision-making

We conduct robustness checks to supplement the main results at metric-based decision-making (Section 7).

Strategies on investment intensity - η^β dependent. We investigate investment strategies based on metrics for projected price η^β, instead of projected earning E_P (Figure S13). Model-based strategies have X in the suffix; model-free *SP-a/b/c* remain the same. Other conditions are the same as the main result. Under the same strategy parameters, PI-based strategies are less profitable than the main result (Figure S11); even the conservative strategy is not securing much earning. This derives from price series' higher variance than E_P series; the latter integrates price over a $t_{p,ind}$ interval and is thus less volatile. The strategy based on historical mean, *SP2X*, performs the best.

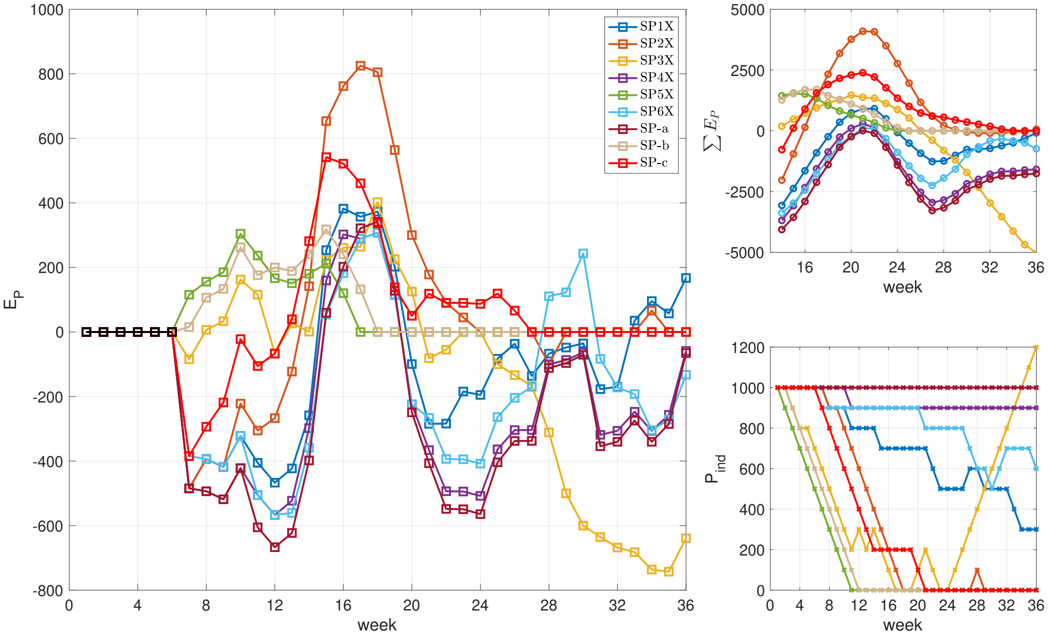


Figure S13: Strategies on investment intensity (η^β-dependent). Comparing model-free strategies *SP-a/b/c* and model-based strategies *SP1X* to *SP6X*. Notations same as Figure 6.

Strategy parameters are experimented with (Figures S14 and S15). Four cases are shown in each plot; invariant parameter values on each plot are the same as in Figures S11 and S12.

Fixed amount execution at the investment decision. We analyze six strategies *SP*1* to *SP*6* and the model-free *SP*-a/b/c* (OS-C). Start with zero investment; at each period t, the action a_P to be made is invest a fixed amount \$100 ($a_P = 1, P_{ind}(t) = 100$), or not invest ($a_P = 0, P_{ind}(t) = 0$). For both strategies based on E_P projection (Figure S16) and strategies based on price projection (Figure S17), results are consistent with main results as fixed increment execution. For example, the aggressive strategy gains much earning during the first mid-term, but suffers a great loss in the second mid-term.

Choice on system parameters. Parameters in model estimation, model projection, and strategy and evaluation (Table 2) are experimented with extensively. The main results are shown with appropriate parameter values. Model estimation and projection are robust to uncertainty parameters; time window parameters behave consistently under different values; strategy parameters lead to desired behavior of strategies. The choice of parameter values is nonetheless subject to future discussions.

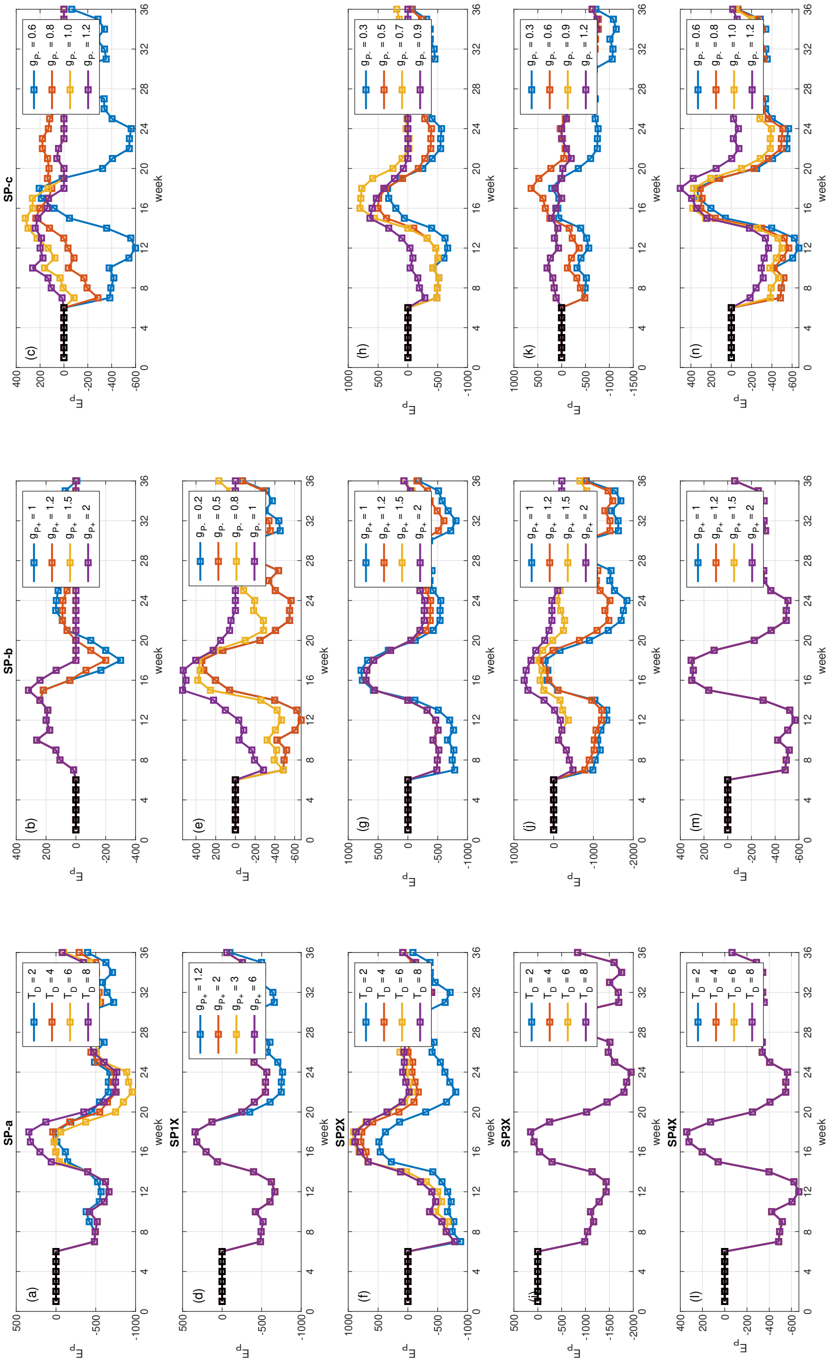


Figure S14: Parameters for strategies on investment intensity (*SP-a/b/c*, *SP1X* - *SP4X*).

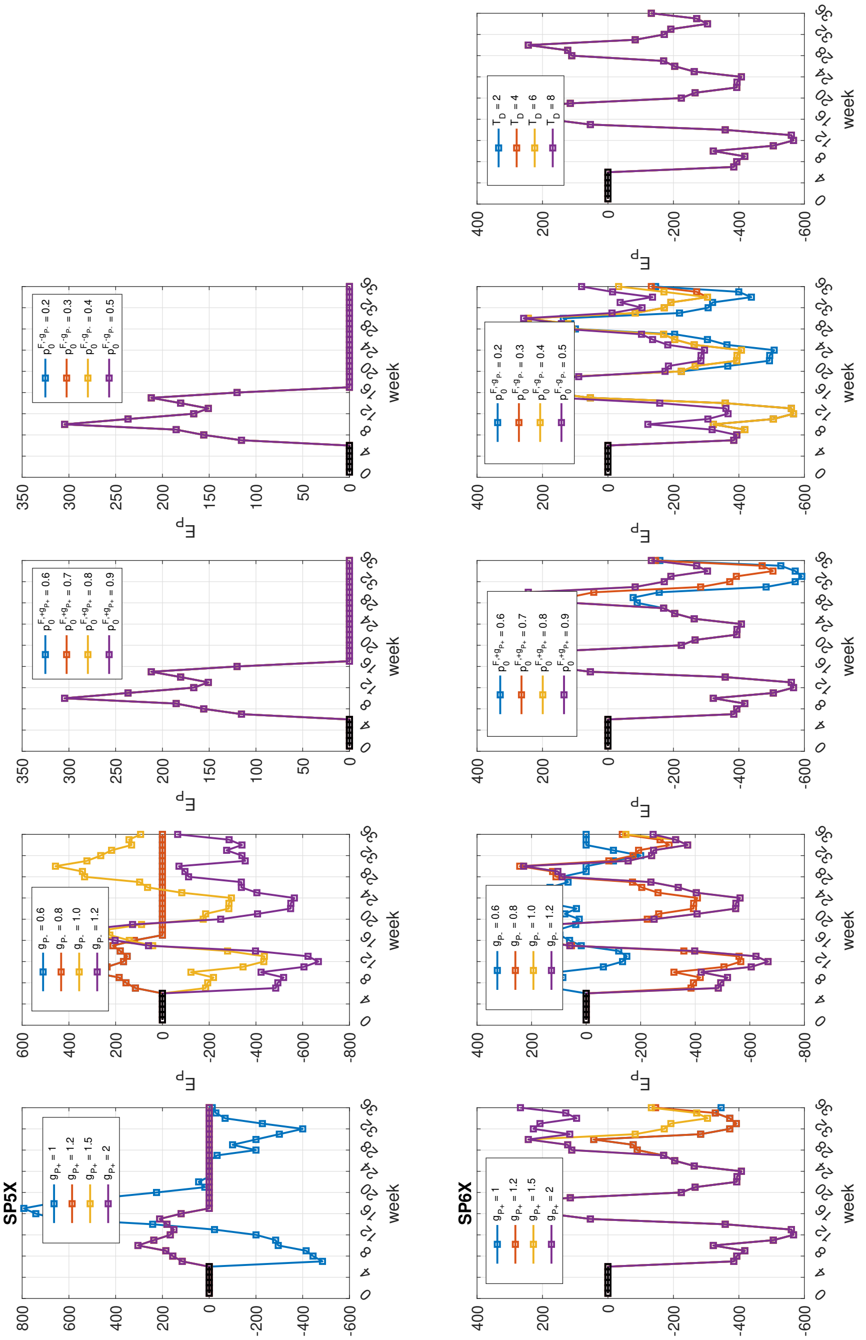


Figure S15: Parameters for strategies on investment intensity (*SP5X* - *SP6X*).

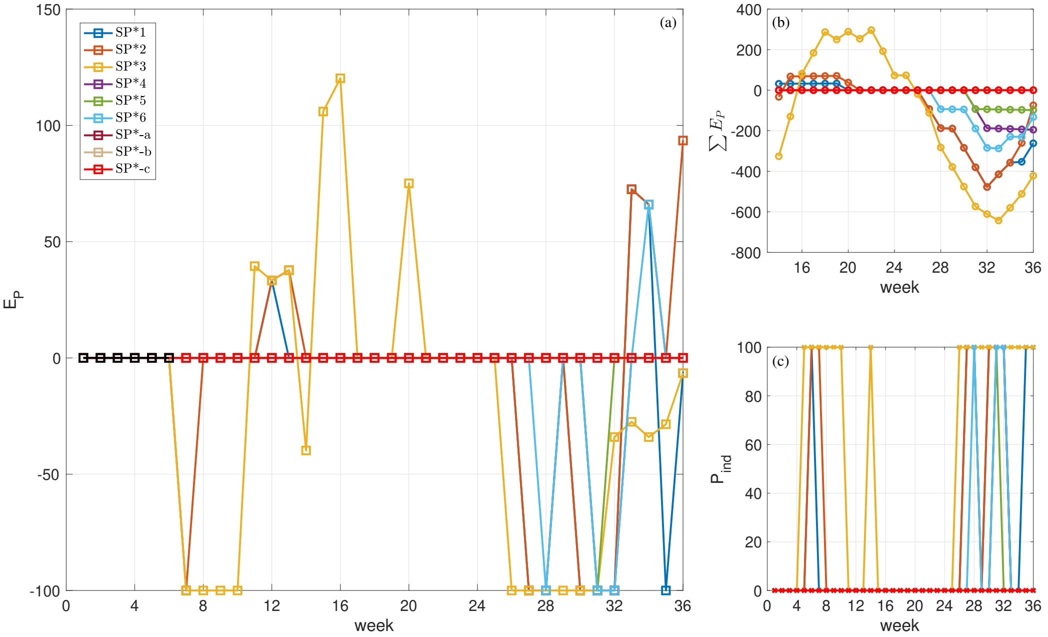


Figure S16: Strategies on investment intensity (E_P-dependent) at fixed amount execution (OS-C). Comparing model-free strategies *SP*-a/b/c* and model-based strategies *SP* 1–SP* 6*. Notations same as Figure 6.

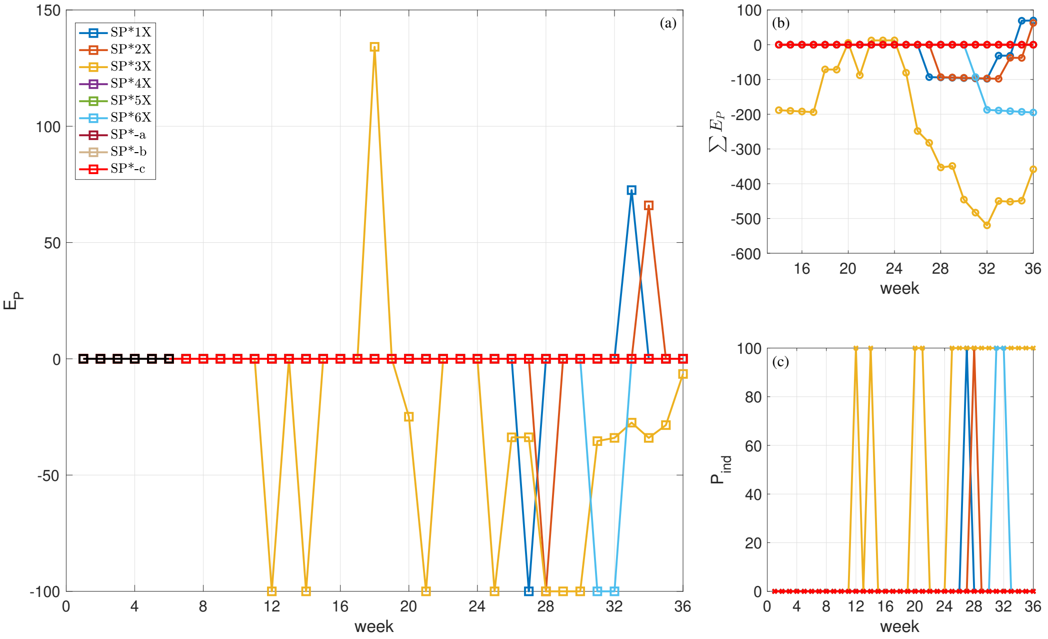
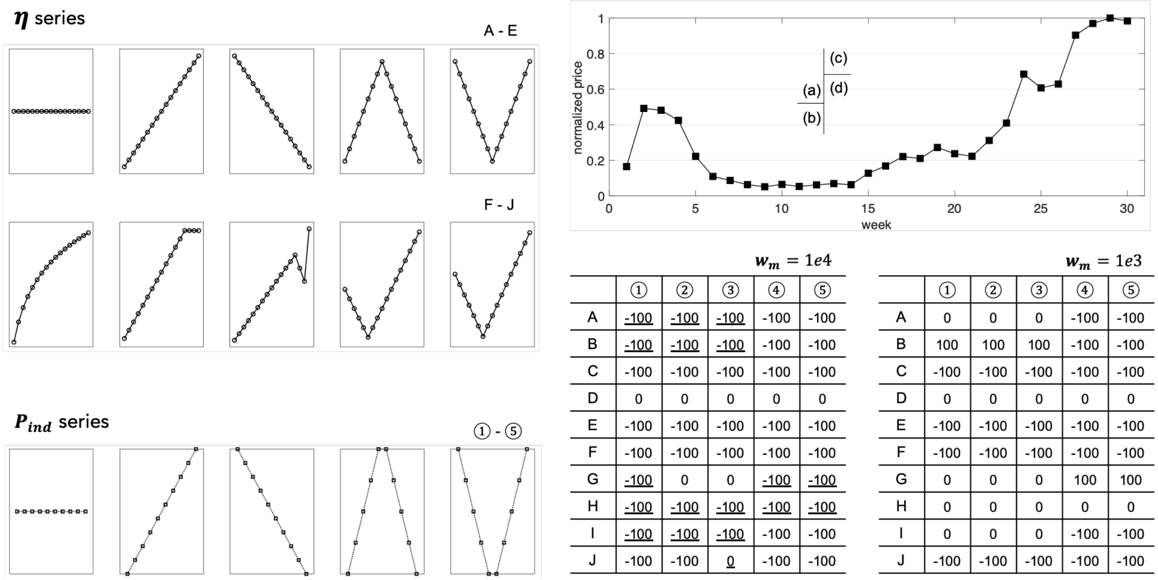


Figure S17: Strategies on investment intensity (η^β-dependent) at fixed amount execution (OS-C). Comparing model-free strategies *SP*-a/b/c* and model-based strategies *SP* 1X–SP* 6X*. Notations same as Figure 6.

OS-I: RL agents' performance on synthetic states

DQN agent. The agent is trained at environment *E1*, learning the policy on ΔP_{ind}. We consider 10 shapes of the η series (Figure S18a), 5 shapes of the P_{ind} series (Figure S18b) (results remain robust for different absolute values of η and P_{ind}), and 2 choices of w_m, totaling at $10 \times 5 \times 2$ synthetic states. The first 30-week data (Figure S18c) is used to estimate platform parameters and specify the environment. The agent has three actions: $\Delta P_{ind} = [-100, 0, 100]$. One epoch passes the acceptance criteria, and the trained agent is collected. Agent's actions on synthetic states are shown (Figure S18d) for both values of w_m.



$w_m = 1e4$

	①	②	③	④	⑤
A	$\underline{-100}$	$\underline{-100}$	$\underline{-100}$	-100	-100
B	$\underline{-100}$	$\underline{-100}$	$\underline{-100}$	-100	-100
C	-100	-100	-100	-100	-100
D	0	0	0	0	0
E	-100	-100	-100	-100	-100
F	-100	-100	-100	-100	-100
G	$\underline{-100}$	0	0	$\underline{-100}$	$\underline{-100}$
H	$\underline{-100}$	$\underline{-100}$	$\underline{-100}$	$\underline{-100}$	$\underline{-100}$
I	$\underline{-100}$	$\underline{-100}$	$\underline{-100}$	-100	-100
J	-100	-100	$\underline{0}$	-100	-100

$w_m = 1e3$

	①	②	③	④	⑤
A	0	0	0	-100	-100
B	100	100	100	-100	-100
C	-100	-100	-100	-100	-100
D	0	0	0	0	0
E	-100	-100	-100	-100	-100
F	-100	-100	-100	-100	-100
G	0	0	0	100	100
H	0	0	0	0	0
I	0	0	0	-100	-100
J	-100	-100	-100	-100	-100

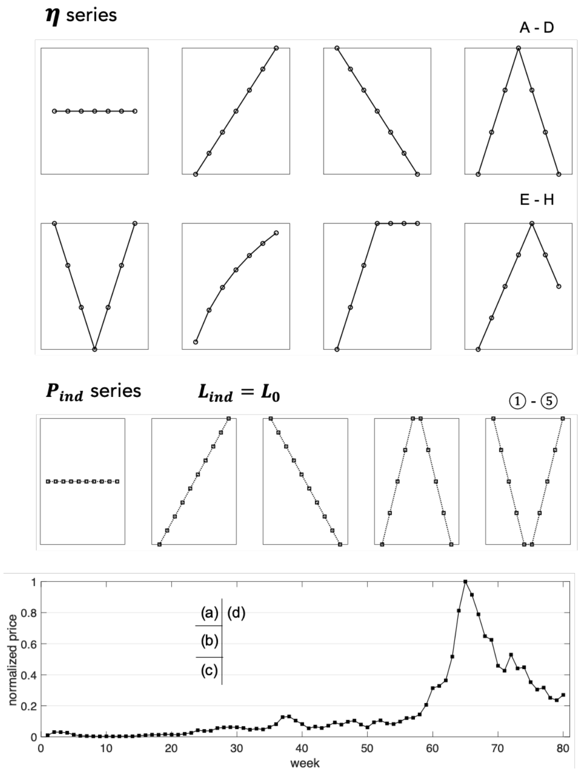
Figure S18: Trained DQN agent's performance on synthetic states. (a) Synthetic platform state η. (b) Synthetic investment history P_{ind}. (c) Data for specifying the RL environment. (d) Agent's actions on synthetic states for two values of w_m.

With other state values being the same, $w_m = 1e4$ corresponds to an earlier stage of platform development compared to $w_m = 1e3$. Overall, the agent's investment decisions are conservative at the platform's early stage (Figure S18d left), decreasing investment ($\Delta P_{ind} = -100$) on most occasions, hold ($\Delta P_{ind} = 0$) on the rest. This agrees with the three-phase platform development (Figure 1): when the platform is in the first phase of development, the laboring reward is high (w_m is large), and massive investors have yet to come to the scene. At this stage, investment activities are dormant on the platform, so decreasing P_{ind} is desired.

When the laboring reward is low ($w_m = 1e3$), the platform enters the second stage where token investment becomes major. On most occasions (Figure S18d right), the agent's investment actions remain invariant; on other occasions (underlined in Figure S18d left), the decisions become more progressive. For example, for the linearly-increasing η curve (type-B), the agent increases P_{ind} at the first three investment histories, showing a positive attitude toward market growth. When the increase on η stops early (type-G), the agent is cautious, putting on hold ($\Delta P_{ind} = 0$) for the same three investment histories. For ④ and ⑤, the agent is conservative ($\Delta P_{ind} = -100$) with the type-B curve, while aggressive ($\Delta P_{ind} = 100$) with the type-G curve, possibly expecting a growth after the plateau.

For both $w_m = 1e3$ and $w_m = 1e4$, when the platform's development has been constantly decreasing (type-C curve) or increasing with an exponentially decaying speed (type-F curve), the agent always chooses to reduce the investment. When platform growth gets down from the peak (type D), the agent chooses to hold level ($\Delta P_{ind} = 0$) instead of fleeing the market; conversely, when the platform recovers from the trough (type E), the agent chooses to reduce investment, and only when the growth sustains beyond previous highs (type I, J) will the agent stop depleting the stock and may hold level (e.g., under ①-③). These actions suggest a good rationality of the trained agent (i.e., buy low, sell high).

PPO agent. We show the agent's performance on synthetic states. The agent is trained at environment *E3*, learning both investment and laboring adjustments $[\Delta P_{ind}, m_L]$. Similar to the DQN agent, we consider 8 different η series (Figure S19a), 5 different P_{ind} series (Figure S19b), and 3 choices of w_m, combining them into $8 \times 5 \times 3$ synthetic states (L_{ind} is fixed at L_0). The first 80-week data of the same series (Figure S19c) is used to specify the environment. The agent has a 3×2 action space: $\Delta P_{ind} = [-100, 0, 100], m_L = [0, 1]$.



$\boldsymbol{w_m} = 1e2$ $\overline{\boldsymbol{\Delta P_{ind}}}$

	①	②	③	④	⑤	ave
A	-24	-34	-34	-23	-39	
B	-15	-11	-35	-19	-21	
C	-19	-29	-12	-19	-29	
D	-20	-32	-28	-28	-34	*-28.4*
E	-26	-30	-11	-29	-31	
F	**-100**	**-100**	**-100**	**-100**	**-100**	
G	-29	-31	-18	-11	-16	
H	-32	-29	-30	-17	-24	

$\boldsymbol{w_m} = 1e2$ $\overline{\boldsymbol{m_l}}$

	①	②	③	④	⑤	ave
A	0.58	0.7	0.61	0.59	0.69	
B	0.62	0.52	0.54	0.66	0.6	
C	0.62	0.61	0.62	0.65	0.6	
D	0.63	0.62	0.62	0.65	0.66	*0.64*
E	0.61	0.62	0.55	0.63	0.64	
F	**1**	**1**	**1**	**1**	**1**	
G	0.6	0.67	0.52	0.61	0.56	
H	0.71	0.63	0.68	0.55	0.61	

$\boldsymbol{w_m} = 1e3$ $\overline{\boldsymbol{\Delta P_{ind}}}$

	①	②	③	④	⑤	
A	-21	-19	-40	-15	-16	
B	-22	-36	-25	-31	-29	
C	-32	-27	-14	-32	-25	
D	-15	-35	-18	-24	-34	*-25.2*
E	-16	-39	-26	-20	-18	
F	**-100**	**-100**	**-100**	**-100**	**-100**	
G	-6	-22	-20	-22	-21	
H	-18	-22	-26	-31	-20	

$\boldsymbol{w_m} = 1e3$ $\overline{\boldsymbol{m_l}}$

	①	②	③	④	⑤	
A	0.62	0.59	0.62	0.57	0.52	
B	0.61	0.69	0.62	0.65	0.59	
C	0.67	0.66	0.58	0.56	0.55	
D	0.65	0.58	0.58	0.56	0.63	*0.60*
E	0.65	0.62	0.62	0.59	0.59	
F	**1**	**1**	**1**	**1**	**1**	
G	0.63	0.64	0.54	0.58	0.64	
H	0.66	0.6	0.61	0.69	0.64	

$\boldsymbol{w_m} = 1e4$ $\overline{\boldsymbol{\Delta P_{ind}}}$

	①	②	③	④	⑤	
A	-31	-24	-26	-14	-17	
B	-23	-32	-13	-31	-29	
C	-12	-18	-28	-31	-25	
D	-28	-19	-23	-19	-22	*-22.2*
E	-37	-27	-31	-21	-18	
F	**-100**	**-100**	**-100**	**-100**	**-100**	
G	-21	-28	-26	-29	-22	
H	-15	-28	-40	-17	-35	

$\boldsymbol{w_m} = 1e4$ $\overline{\boldsymbol{m_l}}$

	①	②	③	④	⑤	
A	0.68	0.64	0.58	0.6	0.54	
B	0.6	0.61	0.52	0.61	0.62	
C	0.63	0.56	0.57	0.6	0.57	
D	0.6	0.58	0.47	0.53	0.59	*0.55*
E	0.61	0.58	0.54	0.64	0.62	
F	**1**	**1**	**1**	**1**	**1**	
G	0.68	0.66	0.61	0.62	0.66	
H	0.55	0.67	0.64	0.67	0.61	

Figure S19: Trained PPO agent's performance on synthetic states. (a) Synthetic platform state η. (b) Synthetic investment history P_{ind}. (c) Data for specifying the RL environment. (d) Agent's actions $[\Delta P_{ind}, m_L]$ on synthetic states for three values of w_m.

The agent's performance is shown in Figure S19d. Since the PPO agent produces stochastic actions, we initiate 100 trials at each state and calculate the average action $\overline{\Delta P_{ind}}$ and $\overline{m_L}$. Overall, the agent's actions are conservative in token investment ($\overline{\Delta P_{ind}} < 0$) while progressive in expanding laboring capacity ($\overline{m_L} > 0$). We highlight two phenomena in the result, avoiding over-interpretation of the agent's performance. First, at type-F platform development trajectory, the agent always disfavors the future growth of token price and chooses to decrease the investment amount across 5 types of investment history P_{ind}. This agrees with the DQN agent's actions (Figure S18d), suggesting that both agents successfully anticipate the limit on growth speed coming from the exponential decay. Meanwhile, under the current environment *E3*, besides decreasing investment intensity, the agent favors the expansion of laboring capacity. This suggests a shift of the participant's primary activity from token investment to laboring on the chain. Second, at type-D trajectory, the agent's actions demonstrate a clear trend across three values of w_m, each decreasing one order of magnitude. For $w_m = 1e4, 1e3, 1e2$, platform development goes from the early to the late stage, along which token investment becomes more and more popular. The agent anticipates this depreciation of investment activities: across 5 investment histories (① – ⑤), the agent's average next-step investment decision is more conservative (i.e., $\overline{\Delta P_{ind}}$ gets smaller) at the late platform stage compared to at the early stage. By contrast, the agent invests more in laboring capacity (i.e., $\overline{m_L}$ gets larger) as the platform becomes mature, showing confidence in the platform's long-term development.

OS-J: Detailed discussion of the literature on methodologies

Besides the Information Systems literature on crowdsourcing, crowdfunding, blockchains, and strategic participation on digital platforms, our study is related to the literature on uncertainties in forecast-based decision-making, optimal execution and algorithm trading, and RL applications in management sciences.

Uncertainties in forecast-based decision-making

For decision-making based on model projection, the uncertainty of forecasts can be indicated with prediction intervals (PI) around the point forecast. A set of PI consists of an upper bound and a lower bound, where a certain fraction (e.g., 95%) of forecast instances lie within this interval. Different methods for constructing PIs are adopted in various fields: wind-power forecasting (*Yuan et al.*, 2019), energy management (*Valencia et al.*, 2015), electricity price forecast (*Shrivastava et al.*, 2015), etc. As summarized in *Shrestha and Solomatine* (2006), uncertainty in model forecast can be characterized from different approaches: construct forecast using the Bayesian methodology (*Bishop*, 1995), estimate uncertainty with statistical properties of the model's error structure (e.g., Gaussian errors in linear regression), use re-sampling or Monte-Carlo methods to yield forecast ensembles (*Stine*, 1985; *Heskes*, 1996), or adopt fuzzy methods with the non-probabilistic logic (*Sáez et al.*, 2014; *Valencia et al.*, 2015; *Kavousi-Fard et al.*, 2015). Along with the recent advance of machine learning, PIs are also constructed using various machine learning models, e.g., neural networks (*Hwang and Ding*, 1997; *Khosravi et al.*, 2010), LSTM (*Yuan et al.*, 2019), SVM (*Shrivastava et al.*, 2015). See *Marín et al.* (2019) for a latest review of PI modeling. Among different approaches, Monte-Carlo methods for PI construction are widely adopted due to their general applicability, despite the associated computational cost. In this approach, an ensemble of model forecasts are collected, and PIs are derived directly from the statistics of this ensemble. Without further assumptions on the uncertainty's structure, Monte-Carlo methods help determine PIs in a straightforward way.

Optimal execution and algorithm trading

The optimal execution problem (*Bertsimas and Lo*, 1998; *Almgren and Chriss*, 2001) determines the optimal execution of orders (i.e., liquidation) over a finite trading horizon, minimizing the trading cost. With a model describing price dynamics, this problem can be cast into a stochastic optimal control problem and solved using dynamic programming (DP). Besides traditional DP solvers, *Nevmyvaka et al.* (2006) presented the first application of RL on optimal execution. Within a time horizon, the system optimizes the execution of a target trading volume upon interaction with the market. Through the discretization of the trading unit and trading interval, the problem is formulated as a Q-learning procedure, which can be solved greedily thanks to the relatively small state-action space. It is shown that the learned policies have a large reward margin over heuristics such as submit-and-leave or immediate-execution.

The above works inspired studies on RL-based trading and portfolio optimization (e.g., *Eilers et al.*, 2014; *Fischer*, 2018; *Liu et al.*, 2018; *Wu et al.*, 2020; *Zhang et al.*, 2020). People adopt different learning methods: value-based (SARSA (*Pendharkar and Cusatis*, 2018), Q-learning (*Hendricks and Wilcox*, 2014; *Pendharkar and Cusatis*, 2018), deep Q-learning (*Jeong and Kim*, 2019), DDQN (*Ning et al.*, 2018; *Dabérius et al.*, 2019)), policy based (PPO (*Dabérius et al.*, 2019), DDPG (*Xiong et al.*, 2018)), or actor-critic methods (*Yang et al.*, 2020). Algorithms are applied to different trading targets: stock (*Hendricks and Wilcox*, 2014), foreign exchange (*Carapuço et al.*, 2018), or cryptocurrency (*Jiang and Liang*, 2017), and RL solutions are compared to heuristic trading strategies such as time-weighted average price (*Dabérius et al.*, 2019) or volume-weighted average price (*Berkowitz et al.*, 1988). With fuzzy layer (*Deng et al.*, 2016), recurrent structure (*Jiang et al.*, 2017), and autoencoder (*Li et al.*, 2019), etc., RL has developed into a key component of algorithm trading (*Cartea et al.*, 2015).

When the agent makes an action (trade), it impacts the environment (market price). The impact can be decomposed into permanent price impact, which persists during the liquidation horizon, and temporary price impact, which exists in the current period (*Bertsimas and Lo*, 1998; *Almgren and Chriss*, 2001). Linear price impact function is often used (*Dabérius et al.*, 2019) (the slope is called Kyle's Lambda (*Kyle*, 1985)). When actions' influence on market price is negligible (e.g., *Nevmyvaka et al.*, 2006), the impact function is omitted.

Deep reinforcement learning

Reinforcement learning (RL) considers the Markov decision process (MDP) where an agent interacts with the environment in discrete time steps (*Arulkumaran et al.*, 2017; *Sutton and Barto*, 2018). It is rooted in value iteration (*Bellman*, 1957) and temporal difference (*Sutton*, 1988) methods. At each time step t, the agent is in state $\boldsymbol{S}_t \in \mathbb{S}$ that describes its own status and the status of the environment. According to a policy π that determines the agent's action under the specific state, the agent performs action $\boldsymbol{A}_t \in \mathbb{A}$, under which $\boldsymbol{S}_t$ transits to $\boldsymbol{S}_{t+1}$ for the next time step. A reward r_t is assigned to the agent, depending on its action. The goal is to learn an optimal policy π^* that maximizes the expectation of the discounted cumulative reward R over a time horizon t_H: $R = \sum_{t=1}^{t_H} \epsilon_T^{t-1} r_t$, where ϵ_T is the discount factor.

Best policy π^* can be learned via value function methods, direct policy search, or the actor-critic approach that is the hybrid of the two (*Arulkumaran et al.*, 2017). Value function methods utilize (i) state-value function $V^\pi(\boldsymbol{S})$ which specifies the expected return of each state $\boldsymbol{S}$ under a certain policy π, (ii) state-action value function $Q^\pi(\boldsymbol{S}, \boldsymbol{A})$ which specifies the expected return of each state-action pair $(\boldsymbol{S}, \boldsymbol{A})$ under π, and (iii) advantage function $A^\pi(\boldsymbol{S}, \boldsymbol{A}) = Q^\pi(\boldsymbol{S}, \boldsymbol{A}) - V^\pi(\boldsymbol{S})$ which records different actions' value difference.

Bellman equation (*Bellman*, 1952) exploits the Markov property of $Q^\pi(\boldsymbol{S}, \boldsymbol{A})$:

$$Q^\pi(\boldsymbol{S}_t, \boldsymbol{A}_t) = \mathbb{E}_{\boldsymbol{S}_{t+1}}[r_{t+1} + \epsilon_T Q^\pi(\boldsymbol{S}_{t+1}, \pi(\boldsymbol{S}_{t+1}))], \tag{2}$$

which suggests that Q^π can be solved gradually through minimizing the temporal difference (TD) error, and then policy π is derived from a determined Q^π. Depending on the TD learning format, Q-learning (off-policy) (*Watkins and Dayan*, 1992) and SARSA (on-policy) (*Rummery and Niranjan*, 1994) are constructed.

The state-action table form of $Q^\pi(\boldsymbol{S}, \boldsymbol{A})$ can be replaced with a neural network that approximates more complex value functions, leading to deep Q-learning (DQN) (*Mnih et al.*, 2015). DQN starts the era of deep reinforcement learning (DRL) and inspires a series of designs (e.g., double DQN (*Van Hasselt et al.*, 2016), asynchronous training (*Mnih et al.*, 2016), dueling architecture (*Wang et al.*, 2016), prioritized experience replay (*Schaul et al.*, 2015)), where deep networks help RL achieve beyond-human performance on many gaming tasks such as Atari games (*Mnih et al.*, 2015) and Go (*Silver et al.*, 2016). For more real-world tasks such as performing physics experiments (*Denil et al.*, 2016) or indoor navigation (*Zhu et al.*, 2017), which are often model-free, DRL is shown to achieve human-comparable performance. Policy-based methods (e.g., PPO - Proximal Policy Optimization (*Schulman et al.*, 2017), TRPO (*Schulman et al.*, 2015)) and actor-critic methods (e.g., DDPG (*Lillicrap et al.*, 2015), multi-agent DDPG (*Lowe et al.*, 2017)) enjoy the benefits of deep networks and join DRL as distinct branches besides value function methods (see below). Recent efforts explore multi-agent RL (*Littman*, 1994), inverse RL (*Ng and Russell*, 2000), hierarchical RL (*Barto and Mahadevan*, 2003), and Meta RL (*Wang et al.*, 2018) within the realm of DRL (see *Arulkumaran et al.* (2017) for a detailed discussion).

TRPO - Trust Region Policy Optimization (*Schulman et al.*, 2015) is an on-policy, policy-gradient RL method for environments with a discrete or continuous action space. It directly estimates a stochastic policy and uses a value function critic to estimate the value of the policy. The KL-divergence between the old and the new policy is used as a constraint during optimization; this algorithm thus prevents significant performance drops compared to standard policy-gradient methods by keeping the updated policy within a trust region close to the current policy.

DDPG - Deep Deterministic Policy Gradient (*Lillicrap et al.*, 2015) is an off-policy, actor-critic RL method for environments with a continuous action space. A DDPG agent learns a deterministic policy while also using a Q-value function critic to estimate the value of the optimal policy. It features a target actor and critic as well as an experience buffer. DDPG agents support offline training, i.e., training from saved data, without an environment.

For more information on the different types of RL agents and their comparisons, one may refer to Matlab's RL toolbox (https://www.mathworks.com/help/reinforcement-learning/ug/create-agents-for-reinforcement-learning.html). For our RL setting, the considerations of a small/large and continuous/discrete/hybrid action space (corresponding to coarse or delicate decision-making on investment and laboring intensities), with an on-/off-policy training agenda, specific abundance condition of the training data, and computation/memory capacity of the training procedure, which depend on the scale of deployment of the decision-making paradigm and its operating resources, determine the choice of effective agents. DQN and PPO are widely-acknowledged starting agents that are relatively easy to tune, memory-efficient, transparent to policy analysis, and well-performing; advanced agents and elaborated training/fine-tuning can be explored in future works.

Parallel to model-free RL is model-based RL, which is closely related to the planning problem (*Sutton*, 1990). Planning typically uses local solutions and focuses on one state or a subset of states, while RL derives global solutions over the entire state space. Despite many successful applications (e.g., *Doya et al.*, 2002; *Polydoros and Nalpantidis*, 2017; *Kaiser et al.*, 2019; *Wang et al.*, 2019), there are various challenges with model-based RL solvers (*Moerland et al.*, 2020), such as stochasticity, limited data, partial observability, multi-step learning, non-stationarity of MDP, etc., some of which also apply to model-free RL. For both model-based and model-free RL, abstraction and interpretation of learned policies require sophisticated representation techniques (e.g., saliency map (*Simonyan et al.*, 2013)) to generate practical insights in the strategy space.

RL applications in management sciences

Optimal control (*Bertsekas*, 2012) is an important topic in operations research and management sciences. Sequential decision-making problems under uncertainty can be modeled as MDP (*Bellman*, 1957; *Howard*, 1960) or semi-MDP (*Das et al.*, 1999; *Gosavi*, 2004), which can be solved with dynamic programming (DP). DP nevertheless often requires a complete and accurate model of the environment and may be feasible only in a constrained state-action space. When the MDP employs a large state space or the environment is only partially observed (i.e., POMDP (*Burnetas and Katehakis*, 1997; *Kaelbling et al.*, 1998)), people use RL to generate near-optimal solutions (*Sutton and Barto*, 2018).

RL and DRL witness applications on a wide range of topics in management studies. A key application domain is supply chain management and inventory control (*Das et al.*, 1999; *Gosavi*, 2004; *Oroojlooyjadid et al.*, 2021). When inventory optimization is single-period, the problem is bandit learning and can be solved with DP; when inventory control is more complicated, such as supervising multiple periods or with unknowns, conducting DP is difficult and RL methods are employed (e.g., *Chen and Simchi-Levi*, 2004a,b; *Chen et al.*, 2021). For example, in a seminal work, *Das et al.* (1999) proposed an algorithm for continuous-time SMDP and applied it to optimizing preventative maintenance in a production inventory system. Along this line of research, *Li et al.* (2012) applies RL to joint pricing, lead-time, and scheduling decisions in make-to-order systems. More recently, *Dai and Gluzman* (2022) applies and advances PPO and TRPO methods to the stochastic control of queueing networks; *Oroojlooyjadid et al.* (2021) applies DQN to the beer game, a classic demonstration of the bullwhip effect in supply chain management, extending earlier works (e.g., *Chaharsooghi et al.*, 2008).

Besides inventory control management and trading optimization as mentioned above, RL is applied to other management domains. *Kokkodis and Ipeirotis* (2021) applies RL to career path recommendation, relying on a Markov decision process to operate on a graph of feasible actions and dynamically recommend profitable career paths. *Chen et al.* (2021) reports combining DRL and prediction models to develop a multi-objective markdown system for the merchandising units at Walmart. *Liang et al.* (2022) reports applying DRL to laptop manufacturing in a major personal computer manufacturing company (Lenovo). RL is also frequently adopted in transportation science. *Oda and Joe-Wong* (2018) employs DQN in a model-free setting to study fleet management. *Shou et al.* (2020) employs a MDP to model idle e-hailing drivers' optimal sequential decisions in passenger-seeking, leveraging an inverse RL technique in a multi-agent setting. *Qin et al.* (2020) reports applying DRL to ride-hailing order dispatching at a large-scale ride-hailing platform (DiDi). *Li et al.* (2021) applies multi-agent RL to adaptive traffic signal control through knowledge-sharing protocols.

OS-K: Addressing the challenges for real-world RL applications

Dulac-Arnold et al. (2019) laid out nine challenges when applying RL to real-world problems. Several challenges can be addressed in our RL application:

- *Off-line and off-policy training.* Our training can be either on- or off-policy. When considering slow deviations of platform development from the projected trajectory, i.e., slow changes in platform parameters, policies can be determined offline.

- *Sample efficiency.* Sample efficiency issues are not applicable to our model-based solution.

- *High-dimensional state/action space.* Our state vector $\boldsymbol{S}$ has an unfixed length (determined by τ and t_p). To reduce the state space, continuous state values can be rendered discrete at different resolutions. Action space $\boldsymbol{A}$ is discrete and small.

- *Safety constraints.* Safety issues are not applicable to our decision-making.

- *Partial observability.* To describe the platform's state, only η is used, calculated from the token price. w_m indicates the platform's development, requiring trial-and-error when training agents for the first time; at re-training, its value can derive from the previous value.

- *Reward function.* Discounted earnings over a finite horizon is a natural reward for our RL environment. We can separate investment and laboring earnings or combine them.

- *Policy explainability.* Policies that govern state-action transitions are more interpretable with value functions (i.e., Q-learning) and less with neural network architectures (i.e., DQN). Summarizing and visualizing the learned policies is a key challenge for our RL application.

- *Real-time operations.* The current analysis assumes weekly decision-making and does not consider real-time operations. It is useful to investigate the framework's applicability to high-frequency decision-making, addressing the constraints for training on the fly.

- *System delays.* A finite history of η is recorded in the state vector to capture system delays. Delay in investment return is a fundamental real-world feature and is addressed in our model. Our RL agents show good foreseeing capacities in learned policies.

OS-L: Granular estimation of platform parameters

We can utilize other data attributes besides price to conduct a granular estimation of platform parameters. Suppose we have data on transaction volume (V), market capital (C, which equals current price times circulating token supply S), and high (X_{high}) and low (X_{low}) token price in each transaction day. Assume that the maximum token supply S_{max} is constant. Platform parameters could then get estimated in the following manner:

- The decaying laboring return in $u_{laborer}$ can be supported by S instead of assuming a constant decay t_0. We can utilise this gradually saturating $S(t)$ in three ways:

  (1) Fit $S(t)$ to the curve $1 - e^{t/t_0}$, then extract the best t_0 for each platform.

  (2) Replace e^{-t/t_0} in $u_{laborer}$ with the normalized series $1 - S(t)/S_{max}$ as the decay.

  (3) Use $1 - S(t)/S_{max}$ to indicate the overall decay, fully replacing the term $\frac{\eta_{min}}{\eta(t)} e^{-t/t_0}$.

- Transaction volume (V) normalized by market capital (C), which equals the proportion of transacted tokens among the circulating token supply, indicates the heat of market activities. The V/C series can be used to characterize the adoption/abandonment strength $\gamma^{+/-}$, in either of the following ways:

  (1) Use V/C series to replace the base adoption/abandon rate γ_0, for each platform.

  (2) Use V/C to indicate the sum of absolute adoption and abandonment strengths: $|\gamma^+| + |\gamma^-| \propto V/C$, with the ratio of $|\gamma^+|$ to $|\gamma^-|$ maintained as in the model.

- Maximum price difference of each transaction day $X_{high} - X_{low}$ reflects the fluctuation of the focal platform's development. We could use this information to anchor the scaling constant Δu_0 for the change in platform utility: $X_{high} - X_{low} \propto \frac{1}{\Delta u_0}$, i.e., high/low fluctuation corresponds to fast/slow change of platform utility.

With these additional data, scaling constants t_0 (at laboring utility decay), Δu_0 (at platform utility change), and γ_0 (at adoption/abandonment strength) are empirically supported.

In general, extensive empirical analysis and detailed estimation help reveal the quality of decision-making under various market conditions and against actual market outcomes.

References

Almgren, R., & Chriss, N. (2001), Optimal execution of portfolio transactions, *Journal of Risk*, 3, 5-40.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017), Deep reinforcement learning: A brief survey, *IEEE Signal Processing Magazine*, 34(6), 26-38.

Barto, A. G., & Mahadevan, S. (2003), Recent advances in hierarchical reinforcement learning, *Discrete Event Dynamic Systems*, 13(1), 41-77.

Bellman, R. (1952), On the theory of dynamic programming, *PNAS*, 38(8), 716.

Bellman, R. (1957), A Markovian decision process, *Journal of Mathematics and Mechanics*, 679-684.

Benjaafar, S., & Hu, M. (2020), Operations management in the age of the sharing economy: What is old and what is new? *Manufacturing & Service Operations Management*, 22(1), 93-101.

Berkowitz, S. A., Logue, D. E., & Noser Jr, E. A. (1988), The total cost of transactions on the NYSE, *The Journal of Finance*, 43(1), 97-112.

Bertsekas, D. (2012), Dynamic programming and optimal control: Volume I (Vol. 1)., *Athena Scientific*.

Bertsimas, D., & Lo, A. W. (1998), Optimal control of execution costs, *Journal of Financial Markets*, 1(1).

Bishop, C. M. (1995), *Neural networks for pattern recognition*, Oxford university press.

Burnetas, A. N., & Katehakis, M. N. (1997), Optimal adaptive policies for Markov decision processes, *Mathematics of Operations Research*, 22(1), 222-255.

Carapuço, J., Neves, R., & Horta, N. (2018), Reinforcement learning applied to Forex trading, *Applied Soft Computing*, 73, 783-794.

Cartea, Á., Jaimungal, S., & Penalva, J. (2015), *Algorithmic and high-frequency trading*, Cambridge University Press.

Chaharsooghi, S. K., Heydari, J., & Zegordi, S. H. (2008), A reinforcement learning model for supply chain ordering management: An application to the beer game, *Decision Support Systems*, 45(4), 949-959.

Chen, Y., Mehrotra, P., Samala, N. K. S., Ahmadi, K., Jivane, V., Pang, L., ... & Pleiman, S. (2021), A multiobjective optimization for clearance in walmart brick-and-mortar stores, *INFORMS Journal on Applied Analytics*, 51(1), 76-89.

Chen, X., & Simchi-Levi, D. (2004a), Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The finite horizon case, *Operations Research*, 52(6), 887-896.

Chen, X., & Simchi-Levi, D. (2004b), Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The infinite horizon case, *Mathematics of Operations Research*, 29(3), 698-723.

Chen, B., Simchi-Levi, D., Wang, Y., & Zhou, Y. (2022), Dynamic pricing and inventory control with fixed ordering cost and incomplete demand information, *Management Science*, 68(8), 5684-5703.

Dabérius, K., Granat, E., & Karlsson, P. (2019), Deep execution-value and policy based reinforcement learning for trading and beating market benchmarks, available at SSRN 3374766.

Dai, J. G., & Gluzman, M. (2022), Queueing network controls via deep reinforcement learning, *Stochastic Systems*, 12(1), 30-67.

Das, T. K., Gosavi, A., Mahadevan, S., & Marchalleck, N. (1999), Solving semi-Markov decision problems using average reward reinforcement learning, *Management Science*, 45(4), 560-574.

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2016), Deep direct reinforcement learning for financial signal representation and trading, *IEEE Transactions on Neural Networks and Learning Systems*, 28(3).

Denil, M., Agrawal, P., Kulkarni, T. D., Erez, T., Battaglia, P., & De Freitas, N. (2016), Learning to perform physics experiments via deep reinforcement learning, arXiv:1611.01843.

Doya, K., Samejima, K., Katagiri, K. I., & Kawato, M. (2002), Multiple model-based reinforcement learning, *Neural Computation*, 14(6), 1347-1369.

Dulac-Arnold, G., Mankowitz, D., & Hester, T. (2019), Challenges of real-world reinforcement learning, arXiv:1904.12901.

Eilers, D., Dunis, C. L., von Mettenheim, H. J., & Breitner, M. H. (2014), Intelligent trading of seasonal effects: A decision support algorithm based on reinforcement learning, *Decision Support Systems*, 64, 100-108.

Fischer, T. G. (2018), Reinforcement learning in financial markets-a survey (No. 12/2018), *FAU Discussion Papers in Economics.*

Gosavi, A. (2004), Reinforcement learning for long-run average cost, *European Journal of Operational Research*, 155(3), 654-674.

Hendricks, D., & Wilcox, D. (2014, March), A reinforcement learning extension to the Almgren-Chriss framework for optimal trade execution, In *2014 IEEE Conference on CIFEr (pp. 457-464)*, IEEE.

Heskes, T. (1996), Practical confidence and prediction intervals, *Advances in Neural Information Processing Systems*, 9.

Howard, R. A. (1960), *Dynamic programming and markov processes.*

Hwang, J. G., & Ding, A. A. (1997), Prediction intervals for artificial neural networks, *Journal of the American Statistical Association*, 92(438), 748-757.

Jeong, G., & Kim, H. Y. (2019), Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning, *Expert Systems with Applications*, 117, 125-138.

Jiang, Z., & Liang, J. (2017, September), Cryptocurrency portfolio management with deep reinforcement learning, In *2017 Intelligent Systems Conference (IntelliSys)* (pp. 905-913). IEEE.

Jiang, Z., Xu, D., & Liang, J. (2017), A deep reinforcement learning framework for the financial portfolio management problem, arXiv:1706.10059.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998), Planning and acting in partially observable stochastic domains, *Artificial Intelligence*, 101(1-2), 99-134.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., ... & Michalewski, H. (2019), Model-based reinforcement learning for atari, arXiv:1903.00374.

Kavousi-Fard, A., Khosravi, A., & Nahavandi, S. (2015), A new fuzzy-based combined prediction interval for wind power forecasting, *IEEE Transactions on Power Systems*, 31(1), 18-26.

Khosravi, A., Nahavandi, S., Creighton, D., & Atiya, A. F. (2010), Lower upper bound estimation method for construction of neural network-based prediction intervals, *IEEE Transactions on Neural Networks*, 22(3), 337-346.

Kokkodis, M., & Ipeirotis, P. G. (2021), Demand-aware career path recommendations: A reinforcement learning approach, *Management Science*, 67(7), 4362-4383.

Kyle, A. S. (1985), Continuous auctions and insider trading, *Econometrica*, 1315-1335.

Li, B., Wang, J., Huang, D., & Hoi, S. C. (2018), Transaction cost optimization for online portfolio selection, *Quantitative Finance*, 18(8), 1411-1424.

Li, X., Wang, J., & Sawhney, R. (2012), Reinforcement learning for joint pricing, lead-time and scheduling decisions in make-to-order systems, *European Journal of Operational Research*, 221(1), 99-109.

Li, Z., Yu, H., Zhang, G., Dong, S., & Xu, C. Z. (2021), Network-wide traffic signal control optimization using a multi-agent deep reinforcement learning, *Transportation Research Part C: Emerging Technologies*, 125, 103059.

Li, T., & Zhang, X. (2024), Development trajectory of blockchain platforms: The role of multirole, *Information Systems Research*, 35(3), 1296-1323.

Li, Y., Zheng, W., & Zheng, Z. (2019), Deep robust reinforcement learning for practical algorithmic trading, *IEEE Access*, 7, 108014-108022.

Liang, Y., Sun, Z., ... & Bai, P. (2022), Lenovo Schedules Laptop Manufacturing Using Deep Reinforcement Learning, *INFORMS Journal on Applied Analytics*, 52(1), 56-68.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015), Continuous control with deep reinforcement learning, arXiv:1509.02971.

Littman, M. L. (1994), Markov games as a framework for multi-agent reinforcement learning, In *Machine Learning Proceedings 1994* (pp. 157-163), Morgan Kaufmann.

Liu, X., Wang, W., Niyato, D., Zhao, N., & Wang, P. (2018), Evolutionary game for mining pool selection in blockchain networks, *IEEE Wireless Communications Letters*, 7(5), 760-763.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Pieter Abbeel, O., & Mordatch, I. (2017), Multi-agent actor-critic for mixed cooperative-competitive environments, *Advances in Neural Information Processing Systems*, 30.

Marín, L. G., Cruz, N., Sáez, D., Sumner, M., & Núñez, A. (2019), Prediction interval methodology based on fuzzy numbers and its extension to fuzzy systems and neural networks, *Expert Systems with Applications*, 119, 128-141.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016, June), Asynchronous methods for deep reinforcement learning, In *International Conference on Machine Learning (pp. 1928-1937)*, PMLR.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015), Human-level control through deep reinforcement learning, *Nature*, 518(7540), 529-533.

Moerland, T. M., Broekens, J., & Jonker, C. M. (2020), Model-based reinforcement learning: A survey, arXiv:2006.16712.

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006, June), Reinforcement learning for optimized trade execution, In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 673-680).

Ng, A. Y., & Russell, S. J. (2000, June), Algorithms for inverse reinforcement learning, *ICML* (Vol. 1, p. 2).

Ning, B., Lin, F. H. T., & Jaimungal, S. (2018), Double deep q-learning for optimal execution, arXiv:1812.06600.

Oda, T., & Joe-Wong, C. (2018, April), Movi: A model-free approach to dynamic fleet management, In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications (pp. 2708-2716)*, IEEE.

Oroojlooyjadid, A., Nazari, M., Snyder, L. V., & Taká č, M. (2021), A deep q-network for the beer game: Deep reinforcement learning for inventory optimization, *Manufacturing & Service Operations Management.*

Pendharkar, P. C., & Cusatis, P. (2018), Trading financial indices with reinforcement learning agents, *Expert Systems with Applications*, 103, 1-13.

Polydoros, A. S., & Nalpantidis, L. (2017), Survey of model-based reinforcement learning: Applications on robotics, *Journal of Intelligent & Robotic Systems*, 86(2), 153-173.

Qin, Z., Tang, X., Jiao, Y., Zhang, F., Xu, Z., Zhu, H., & Ye, J. (2020), Ride-hailing order dispatching at didi via reinforcement learning, *INFORMS Journal on Applied Analytics*, 50(5), 272-286.

Rummery, G. A., & Niranjan, M. (1994), *On-line Q-learning using connectionist systems* (Vol. 37, p. 14), Cambridge, UK: University of Cambridge, Department of Engineering.

Sáez, D., Ávila, F., Olivares, D., Cañizares, C., & Marín, L. (2014), Fuzzy prediction interval models for forecasting renewable resources and loads in microgrids, *IEEE Transactions on Smart Grid*, 6(2), 548-556.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015), Prioritized experience replay, arXiv:1511.05952.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, June), Trust region policy optimization, In *International Conference on Machine Learning (pp. 1889-1897)*, PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017), Proximal policy optimization algorithms, arXiv:1707.06347.

Shou, Z., Di, X., Ye, J., Zhu, H., Zhang, H., & Hampshire, R. (2020), Optimal passenger-seeking policies on E-hailing platforms using Markov decision process and imitation learning, *Transportation Research Part C: Emerging Technologies*, 111, 91-113.

Shrestha, D. L., & Solomatine, D. P. (2006), Machine learning approaches for estimation of prediction interval for the model output, *Neural Networks*, 19(2), 225-235.

Shrivastava, N. A., Khosravi, A., & Panigrahi, B. K. (2015), Prediction interval estimation of electricity prices using PSO-tuned support vector machines, *IEEE Transactions on Industrial Informatics*, 11(2).

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... & Hassabis, D. (2016), Mastering the game of Go with deep neural networks and tree search, *Nature*, 529(7587), 484-489.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2013), Deep inside convolutional networks: Visualising image classification models and saliency maps, arXiv:1312.6034.

Stine, R. A. (1985), Bootstrap prediction intervals for regression, *Journal of the American Statistical Association*, 80(392), 1026-1031.

Sutton, R. S. (1988), Learning to predict by the methods of temporal differences, *Machine Learning*, 3(1), 9-44.

Sutton, R. S. (1990), Integrated architectures for learning, planning, and reacting based on approximating dynamic programming, In *Machine Learning Proceedings 1990 (pp. 216-224)*, Morgan Kaufmann.

Sutton, R. S., & Barto, A. G. (2018), Reinforcement learning: An introduction, *MIT press*.

Valencia, F., Collado, J., Sáez, D., & Marín, L. G. (2015), Robust energy management system for a microgrid based on a fuzzy prediction interval model, *IEEE Transactions on Smart Grid*, 7(3), 1486-1494.

Van Hasselt, H., Guez, A., & Silver, D. (2016, March), Deep reinforcement learning with double q-learning, In *Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 30, No. 1)*.

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., ... & Ba, J. (2019), Benchmarking model-based reinforcement learning, arXiv:1907.02057.

Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., ... & Botvinick, M. (2018), Prefrontal cortex as a meta-reinforcement learning system, *Nature Neuroscience*, 21(6), 860-868.

Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June), Dueling network architectures for deep reinforcement learning, In *ICML (pp. 1995-2003)*, PMLR.

Watkins, C. J., & Dayan, P. (1992), Q-learning, *Machine Learning*, 8(3), 279-292.

Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020), Adaptive stock trading strategies with deep reinforcement learning methods, *Information Sciences*, 538, 142-158.

Xiong, Z., Liu, X. Y., Zhong, S., Yang, H., & Walid, A. (2018), Practical deep reinforcement learning approach for stock trading, arXiv:1811.07522.

Yang, H., Liu, X. Y., Zhong, S., & Walid, A. (2020, October), Deep reinforcement learning for automated stock trading: An ensemble strategy, In *First ACM International Conference on AI in Finance* (pp. 1-8).

Yuan, X., Chen, C., Jiang, M., & Yuan, Y. (2019), Prediction interval of wind power using parameter optimized Beta distribution based LSTM model, *Applied Soft Computing*, 82, 105550.

Zhang, Z., Zohren, S., & Roberts, S. (2020), Deep reinforcement learning for trading, *Journal of Financial Data Science*, 2(2), 25-40.

Zhu, Y., Mottaghi, R., ..., & Farhadi, A. (2017, May), Target-driven visual navigation in indoor scenes using deep reinforcement learning, In *2017 IEEE-ICRA (pp. 3357-3364)*, IEEE.