Published at ICLR 2026 Workshop on AI for Mechanism Design and Strategic Decision Making.

Continuous RTS-PnO: Constraint-Aware Training for Robust Rolling-Horizon Budget Allocation

Rassul Magauin
MBZUAI
rassulmagauin@gmail.com

Fuyuan Lyu
McGill & Mila - Quebec AI Institute
MBZUAI
lyufuyuan@gmail.com

Lu Han
Amazon
lhhan@amazon.com

Steve Liu
McGill & MBZUAI
xue.liu@mail.mcgill.ca

Abstract

Fund allocation in financial markets requires decision-making frameworks that effectively balance forecasting accuracy with execution robustness. Recent advancements have successfully integrated uncertainty quantification into end-to-end learning. However, these frameworks fundamentally rely on **open-loop control**, where the entire execution plan is fixed at the initial time step conditioned solely on the initial information. This static dependency ignores intermediate market feedback, making the model vulnerable to evolving dynamics. In this work, we propose **Continuous RTS-PnO**, a unified framework that transitions to a **closed-loop** rolling-horizon paradigm, where the decision is adjusted conditioning on the latest information at each step. Apart from dynamic adjustments, close-loop paradigm also needs to prevent overconfidence in planning. Our approach introduces two key innovations targeting these challenges: (1) *Constraint-Aware Training*, which embeds execution safety caps directly into the SPO+ loss function to prevent overconfident planning, and (2) *Continuous Rolling Execution*, which enables dynamic replanning at each time step. Empirical results across 11 currency pairs demonstrate that our method outperforms static baselines in 7 out of 11 cases. The gains are particularly pronounced in low-volatility regimes, achieving up to **48%** reduction in regret by effectively mitigating model overconfidence.

1 Introduction

The problem of strategic fund allocation, acquiring a target volume of assets over a specified time horizon to minimize cost, is central to algorithmic trading and corporate treasury management. Traditional "naive" approaches typically treat forecasting and optimization as independent, sequential tasks: a model first predicts future prices to minimize a statistical metric like Mean Squared Error (MSE), and a solver subsequently optimizes allocation based on these fixed predictions. However, this separation often yields suboptimal results because the forecasting loss function is misaligned with the ultimate financial objective; a model may achieve low MSE while missing critical directional shifts that determine profitability (Lyu et al., 2025).

To address this limitation, the *Predict-and-Optimize (PnO)* (Amos & Kolter, 2017; Agrawal et al., 2019; Elmachtoub & Grigas, 2022) framework has emerged as a promising paradigm. These end-to-end methodologies integrate learning and decision-making by differentiating through the optimization layer, leveraging downstream decision regret as the training signal rather than relying solely on prediction metrics such as MSE. Building on this paradigm,

Lyu et al. (2025) recently introduced **RTS-PnO**, a risk-aware framework designed to handle stochastic volatility. RTS-PnO integrates *Conformal Prediction* (Angelopoulos & Bates, 2021) to construct calibrated confidence intervals, forcing the optimizer to make decisions based on risk-adjusted upper bounds rather than raw point forecasts.

However, deploying these static PnO frameworks in real-world financial systems reveals a critical limitation: they operate as **"one-shot" open-loop systems**. In practice, RTS-PnO commits to a full allocation schedule at time $t = 0$ and executes it without adjustment, despite new price information arriving at each subsequent time step. When sudden structural breaks occur during the execution period, the static model cannot adapt, potentially leading to substantial cumulative regret. Moreover, real-world execution rarely operates without constraints. Liquidity mandates often impose strict caps on the budget that can be spent within any single interval—constraints that standard PnO training fails to account for.

In short, fund allocation frameworks must address two critical challenges:

1. The Need for Dynamic Recourse. Standard PnO methodologies typically generate a static allocation vector $\mathbf{a}^*$ at time $t = 0$ for the entire horizon H. While Reinforcement Learning approaches (Wang et al., 2021) address sequential decision-making, they often lack the explicit safety guarantees required for liquidity management. Consequently, current PnO methods rely on an "open-loop" strategy that fails to incorporate new market information as it unfolds.

2. Training-Execution Misalignment. Real-world execution operates under strict liquidity mandates, such as a Safety Cap κ (e.g., "spend no more than 10% of the budget in a single interval"). However, standard PnO models are trained on the full, unbounded decision polytope, wasting model capacity on learning features for strategies (e.g., "buy 100% now") that are theoretically optimal but practically infeasible and risky. This fundamental disconnect means models trained without knowledge of κ generate gradients misaligned with the constrained deployment environment.

To bridge these gaps, we propose **Continuous RTS-PnO**[1], a novel closed-loop framework that unifies constraint-aware learning with dynamic control. Our key innovation lies in embedding the safety cap κ directly into the end-to-end training loop through a modified Smart Predict-then-Optimize (SPO+) loss – treating constraints not as a post-processing step, but as a first class component of the learning objective. We then deploy this constraint-specialized model using a rolling-horizon strategy (Boyd et al., 2017) that re-plans at every time step, enabling adaptive recourse while maintaining safety guarantees throughout execution.

We empirically evaluate Continuous RTS-PnO across 11 diverse financial datasets, ranging from stable currencies (e.g., USDSGD) to high-volatility cryptocurrencies (e.g., Coinbase). Our results demonstrate that the rolling-horizon framework outperforms the static baseline in 7 out of 11 cases, with the most significant gains observed in low-volatility regimes. Specifically, we achieve up to **48% reduction in regret** for the Bangladeshi Taka (USDBDT) and over **10% improvement** for four other major currency pairs, validating that dynamic recourse effectively mitigates the "static traps" of traditional open-loop allocation.

2 Methodology

In this section, we formally define the budget allocation problem and introduce the Continuous RTS-PnO framework. Our method builds upon the risk-aware foundation of Lyu et al. (2025) but introduces a novel constraint-aware training objective and a rolling execution mechanism.

2.1 Problem Formulation

Let $y \in \mathbb{R}^H_{>0}$ be the vector of future asset prices over a horizon H. The goal is to determine an allocation vector $\mathbf{a} \in \mathbb{R}^H_{\geq 0}$ such that the total budget is utilized ($\sum_{t=1}^{H} a_t = 1$) to minimize the total cost $\mathbf{a}^\top y$.

[1]Code available at `https://github.com/rassulmagauin/RTS-PnO-C`

To ensure robust execution, we enforce two critical constraints. First, a **Safety Cap** $\kappa \in (0, 1]$, which limits the liquidity impact at any single time step. Second, following the RTS-PnO framework (Lyu et al., 2025), we impose a **Risk Tolerance** r_0, which bounds the total exposure to predictive uncertainty. The constrained optimization problem is defined as:

$$\mathbf{a}^*(y) = \arg\min_{\mathbf{a} \in \mathcal{S}_{\kappa, r_0}} \quad \mathbf{a}^\top y \tag{1}$$

$$\text{s.t.} \quad \sum_{t=1}^{H} a_t = 1 \tag{2}$$

$$0 \leq a_t \leq \kappa \quad \forall t \in \{1, \ldots, H\} \tag{3}$$

$$\mathbf{a}^\top \mathbf{r} \leq r_0 \tag{4}$$

where $\mathbf{r} \in \mathbb{R}^H$ represents the positional uncertainty vector derived via Conformal Prediction, and $\mathcal{S}_{\kappa, r_0}$ denotes the feasible region defined by these constraints.

2.2 Risk Quantification via Conformal Prediction

To maintain the robustness of the original RTS-PnO framework, we explicitly model aleatoric uncertainty using **Conformal Prediction** (Angelopoulos & Bates, 2021). As highlighted in Lyu et al. (2025), fixing a static risk-tolerance constant is often infeasible because forecasting uncertainty varies across datasets and training epochs. To address this, we adopt an **adaptive uncertainty constraint**.

Given a trained forecasting backbone (e.g., PatchTST (Nie et al., 2023)), we compute nonconformity scores on a calibration set: $s_i = |y_i - \hat{y}_i|$. Instead of setting an arbitrary margin, we strictly bound the risk by defining q_α as the $(1 - \alpha)$ quantile of these scores.

During both training and inference, we enforce pessimism by substituting the raw point forecast $\hat{y}$ with the risk-adjusted upper bound $\tilde{y}$ in the optimization objective:

$$\tilde{y} = \hat{y} + q_\alpha \cdot \mathbf{1} \tag{5}$$

Consequently, the optimization oracle solves $\min_{\mathbf{a} \in \mathcal{S}_{\kappa, r_0}} \mathbf{a}^\top \tilde{y}$, thereby minimizing the worst-case expected cost rather than the nominal cost. This ensures the model avoids allocating budget to time steps where the lower bound of the price is uncertain or potentially high.

2.3 Constraint-Aware Training (SPO+)

A core contribution of this work is integrating the safety cap κ into the training process. Standard MSE loss treats all prediction errors equally. However, forecasting errors are only relevant if they alter the optimal decision $\mathbf{a}^*$. We adopt the **SPO+ Loss** (Elmachtoub & Grigas, 2022), a convex surrogate for decision regret. Critically, we compute this loss relative to the **constrained feasible set** $\mathcal{S}_{\kappa, r_0}$, rather than the generic simplex.

Given a prediction $\hat{y}$ and ground truth y, the Constraint-Aware SPO+ loss is:

$$\mathcal{L}_{SPO+}(\hat{y}, y) = \underbrace{(2\hat{y} - y)^\top \mathbf{w}^*_\kappa(2\hat{y} - y)}_{\text{Perturbed Decision Cost}} - \underbrace{(2\hat{y} - y)^\top \mathbf{w}^*_\kappa(y)}_{\text{Baseline Adjustment}} \tag{6}$$

where $\mathbf{w}^*_\kappa(c) = \arg\min_{\mathbf{w} \in \mathcal{S}_{\kappa, r_0}} c^\top \mathbf{w}$.

By solving the optimization oracle $\mathbf{w}^*_\kappa$ with the constraint $a_t \leq \kappa$ inside the training loop, the gradients $\nabla_\theta \mathcal{L}_{SPO+}$ implicitly encode the execution limits. *Intuition:* If the model predicts a price drop that would theoretically justify buying 100% of the asset, but $\kappa = 0.25$, the solver returns 0.25. The resulting gradient is scaled down compared to the unconstrained case. This prevents the model from "over-learning" extreme signals that are unactionable in practice, focusing its capacity on the decision boundaries within $[0, \kappa]$.

2.4 Continuous Rolling Execution

To address the limitations of static planning, we deploy the trained model using a rolling-horizon strategy (Boyd et al., 2017). At every time step t, the model re-evaluates the market using the most recent observations.

Algorithm 1 Continuous RTS-PnO Execution

Input: History Y_{hist}, Horizon H, Budget B_t, Safety Cap κ, Risk Tolerance r_0
for $t = 1$ to H **do**
  Forecast: Generate prices $\hat{y}_{t:t+H}$ conditioned on Y_{hist} using the constraint-aware trained model.
  Calibrate: Apply conformal bounds: $\tilde{y} \leftarrow \hat{y} + q_\alpha$.
  Optimize: Solve Eq. 1 for horizon H with remaining budget B_t, cap κ, and risk tolerance r_0.
  Execute: Perform only the first action a_t^*.
  Update: $B_{t+1} \leftarrow B_t - a_t^*$; Append realized price y_t to context Y_{hist}.
end for

This rolling mechanism provides a feedback loop. If the forecast at t proves incorrect, the optimization at $t+1$ accounts for the remaining budget and the updated price trajectory, allowing the model to recover from errors—a feature absent in static PnO.

3 Experiments

To comprehensively evaluate the proposed framework, we design experiments to answer the following three research questions:

- **RQ1:** Can the Continuous RTS-PnO framework achieve superior decision quality compared to standard static baselines across diverse asset classes?
- **RQ2:** How does the continuous RTS-PnO framework perform across different markets?
- **RQ3:** What drives the superior performance of the continuous RTS-PnO framework over the static one?

3.1 Experimental Setup

Benchmarks. We evaluate our framework on 11 real-world datasets spanning three diverse financial domains:

- **Currency Pairs:** USDCNY, USDJPY, AUDUSD, USDBDT, USDRUB, USDFJD, USDSGD, USDTWD.
- **Market Indices:** SP500, DJIA.
- **Cryptocurrency:** Coinbase (BTC).

These datasets were selected to cover a broad spectrum of market conditions. To quantify the stability of each asset, we calculate the **Coefficient of Variation (CV)**, defined as:

$$CV = \frac{\sigma}{\mu} \times 100\% \tag{7}$$

where σ is the standard deviation and μ is the mean price of the asset. The volatility in our benchmarks ranges from highly stable assets like USDSGD ($CV \approx 0.33\%$) to highly volatile assets like Coinbase ($CV \approx 31.01\%$).

Baselines. We compare our proposed method against the standard **Static RTS-PnO** approach (Lyu et al., 2025). To ensure a fair evaluation of the execution strategy, both the baseline and our Continuous framework utilize the same forecasting backbone, PatchTST (Nie et al., 2023). In the baseline setting, the model is trained without execution constraints (effectively $\kappa = 1.0$), and the allocation schedule is generated once at $t = 0$ for the entire horizon H. This serves as a direct comparison between "Open-Loop Unconstrained" execution and "Closed-Loop Constrained" execution.

Metrics. We measure performance using **Regret**, which quantifies the opportunity cost of a decision strategy relative to the optimal hindsight solution. Let $\mathbf{a}$ be the realized allocation

vector and $\mathbf{a}^*$ be the optimal allocation given full knowledge of future prices y. The regret is defined as:

$$\text{Regret} = \sum_{t=1}^{H} a_t y_t - \sum_{t=1}^{H} a_t^* y_t \tag{8}$$

To allow for comparison across assets with vastly different price magnitudes (e.g., BTC vs. JPY), we also report improvement percentages relative to the baseline.

3.2 Main Results (RQ1)

We first evaluate whether the proposed Continuous RTS-PnO framework yields superior decision quality compared to the standard Static PnO baseline. Table 1 summarizes the performance across all 11 benchmarks.

Table 1: Performance Comparison. "Best Cap" denotes the constraint κ that yielded the lowest regret. To visualize the economic scale of the error, the "% Price" columns show Regret as a percentage of the average asset price.

Dataset	Volatility (CV)	Static Regret	% Price	Ours Regret	% Price	Improvement	Best Cap (κ)
USDBDT	2.61%	0.3503	0.31%	**0.1820**	**0.16%**	**+48.04%**	0.60
USDSGD	0.33%	0.0020	0.15%	**0.0017**	**0.13%**	**+15.92%**	0.40
USDCNY	0.21%	0.0036	0.05%	**0.0032**	**0.04%**	**+11.18%**	0.75
USDRUB	2.28%	0.7347	0.79%	**0.6576**	**0.71%**	**+10.49%**	0.30
USDTWD	0.37%	0.0439	0.14%	**0.0394**	**0.12%**	**+10.34%**	0.50
AUDUSD	0.75%	0.0020	0.31%	**0.0019**	**0.29%**	**+5.32%**	0.25
USDJPY	1.31%	0.0050	0.33%	**0.0048**	**0.32%**	**+4.45%**	0.70
SP500	21.16%	124.05	7.77%	**123.59**	7.74%	+0.37%	0.90
COINBASE	31.01%	**1908.40**	5.91%	1932.49	5.99%	-1.26%	0.75
DJIA	15.76%	**995.23**	7.28%	1044.50	7.64%	-4.95%	0.70
USDFJD	0.53%	**0.0050**	0.22%	0.0054	0.24%	-8.99%	0.20

Overall Performance. As illustrated in Figure 1, Continuous RTS-PnO achieves positive regret reduction in 7 out of 11 datasets. The performance gains are particularly pronounced in emerging market currencies (e.g., USDBDT, USDRUB) and stable major pairs (USDSGD, USDCNY), with improvements exceeding 10%.

Remark on Policy Constraints. We acknowledge that, in practice, the Safety Cap κ is often an exogenous policy constraint rather than a tunable hyperparameter. We report the "Best Cap" performance here to demonstrate the maximum potential efficacy of the rolling framework under optimal conditions. A detailed analysis of performance sensitivity across the full range of policy values $\kappa \in [0.1, 1.0]$ is presented in Section 3.3 (Figure 2).

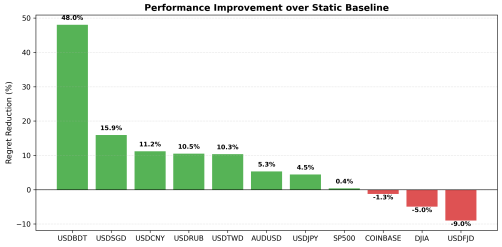


Figure 1: Regret Reduction across datasets. Positive values indicate our method outperformed the static baseline. The method excels in currency markets but struggles in high-noise equity/crypto indices.

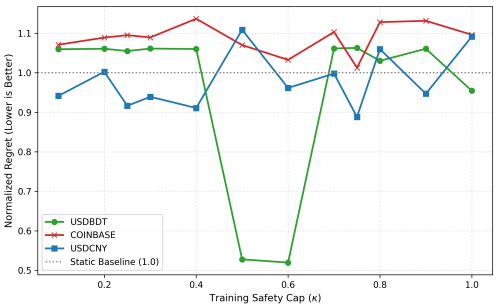


Figure 2: Impact of Safety Cap κ on Regret (Normalized). USDBDT shows a clear optimal constraint region, demonstrating the benefit of Constraint-Aware Training. Coinbase shows erratic performance, highlighting the difficulty of dynamic control in high-volatility environments.

Sensitivity to Constraint κ. A key component of our framework is training the model with a specific safety cap. Figure 2 visualizes the relationship between the cap size κ and normalized regret for representative datasets.

- For **USDBDT** (Green), we observe a clear convex "U-shape," indicating that there exists an optimal level of restriction ($\kappa = 0.6$). A cap that is too loose ($\kappa = 1.0$) fails to constrain overconfidence, while a cap that is too strict ($\kappa = 0.1$) stifles necessary execution.
- In contrast, **Coinbase** (Red) exhibits noisy behavior with no clear optimum, suggesting that in high-volatility regimes, the signal-to-noise ratio is too low for fine-grained constraint tuning to be effective.

3.3 The Volatility Paradox (RQ2)

A common hypothesis in control theory is that dynamic replanning is most valuable in high-volatility environments where uncertainty is high. However, our empirical results contradict this. As illustrated in Figure 3, where datasets are sorted by volatility, we observe a distinct negative correlation between market instability and performance improvement.

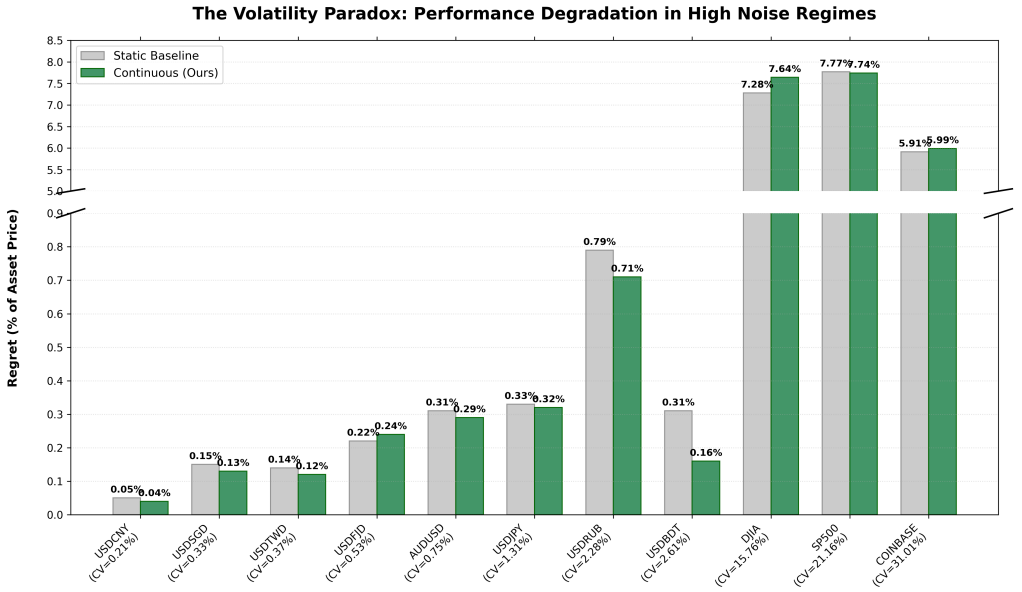
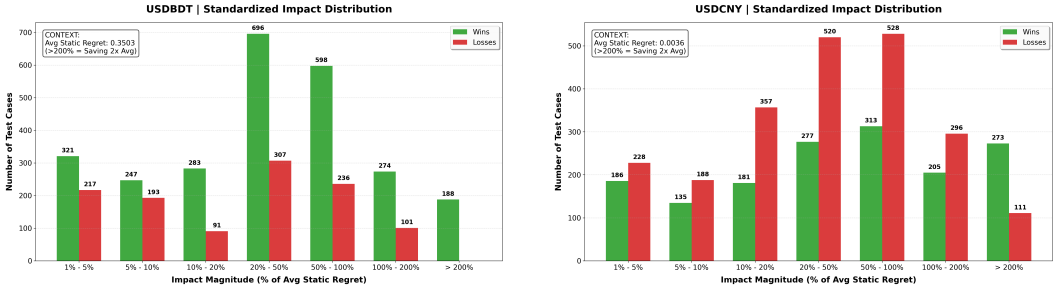


Figure 3: **The Volatility Paradox.** Datasets are sorted by volatility (CV) from low (left) to high (right). We observe a clear degradation in relative improvement as volatility increases. While the rolling framework thrives in stable regimes (e.g., USDBDT, +48%), it yields diminishing returns in high-noise environments.

This phenomenon can be explained by the reliability of the re-prediction signal:

- **Low Volatility Regime (Trustworthy Trends):** In stable markets (e.g., USDBDT, USDSGD), price movements often follow consistent drifts. Here, the rolling update is highly effective because the model can "trust" the emerging trend. Re-predicting at each step allows the model to lock onto these reliable signals and refine its execution precision, correcting the "overconfidence" of static baselines that commit budget too early.
- **High Volatility Regime (Uncertainty Dominance):** In chaotic markets (e.g., Coinbase, $CV \approx 31\%$), the signal-to-noise ratio is inherently low. Although the theoretical room for improvement is large, re-predicting at every step provides little benefit because the model "cannot be sure" of the new information. The updated



(a) USDBDT: High-Conviction (b) USDCNY: Asymmetric Risk

Figure 4: Standardized Impact Distributions. (a) In USDBDT, the model dominates with wins $> 200\%$. (b) In USDCNY, the model secures a net victory by winning rare, high-magnitude events (right tail) despite frequent small losses.

forecasts remain highly uncertain, rendering the dynamic recourse ineffective and potentially causing the model to over-react to mean-reverting noise.

Boundary Condition (The Pegged Asset Anomaly). A notable exception in the low-volatility cluster is USDFJD (-8.99% improvement). We attribute this to the specific exchange rate mechanics. According to the IMF (International Monetary Fund, 2024), the Fiji Dollar operates under a strict "conventional peg," which enforces structural mean-reversion. Unlike "stabilized" currencies like USDSGD where trends exist, strict pegs make distinct directional drifts rare. In this specific case, the rolling controller likely misinterprets peg-maintenance noise as trend initiation, leading to transaction churn, whereas the static baseline efficiently captures the stable mean.

3.4 Impact Analysis (RQ3)

To investigate the source of the performance gain brought by the continuous rolling execution, we analyze the distribution of regret reduction on a given asset time series. We categorize test cases by their **Standardized Impact** δ_i, defined as the regret reduction normalized by the average static regret:

$$\delta_i = \frac{\text{Regret}^{(i)}_{\text{static}} - \text{Regret}^{(i)}_{\text{ours}}}{\text{Avg}(\text{Regret}_{\text{static}})} \times 100\% \tag{9}$$

Values of $\delta_i > 200\%$ indicate savings that are double the average error of the baseline. Figure 4 contrasts two distinct "winning" archetypes found in our results: **USDBDT** (High-Conviction Regime) and **USDCNY** (Asymmetric Risk Profile).

High-Conviction Regime (USDBDT). As shown in Figure 4(a), the USDBDT dataset exemplifies a high-conviction regime. The Continuous model achieves a win rate of 64% and captures a "fat tail" of massive opportunities. In the $> 200\%$ impact bucket, wins outnumber losses by nearly 5:1 (450 vs 94). We attribute this performance disparity to the rolling controller's ability to adapt to structural breaks—most notably the significant devaluation of the Taka in FY23 (International Monetary Fund, 2023). This highlights the structural advantage of closed-loop control: unlike open-loop systems committed to a pre-computed schedule, the rolling mechanism actively detected the regime shift and corrected its trajectory as the devaluation unfolded.

Asymmetric Risk Profile (USDCNY). Figure 4(b) illustrates a more subtle, yet critical, behavior. Here, the Continuous model actually has a low win rate (37%). The red bars dominate the low-impact regions (1%–10%), implying that on average days, the constraint κ incurs a small opportunity cost. However, in the critical $> 200\%$ region, the model wins decisively (309 vs 146). This confirms that in stable markets like USDCNY, the value of RTS-PnO lies not in daily efficiency, but in preventing catastrophic errors during rare volatility spikes.

Synthesis. Across all winning datasets (including USDSGD and USDRUB), a consistent pattern emerges: the performance delta is heavily weighted towards the right tail of the distribution. This validates that the Constraint-Aware Training successfully focuses model capacity on high-stakes decisions rather than fitting noise in low-stakes periods.

3.4.1 Case Study

To visualize how the rolling-horizon strategy yields better decisions, we examine two specific test cases where Continuous RTS-PnO significantly outperformed the Static baseline. While Figure 5 depicts high-impact 'Strong Win' scenarios to illustrate the mechanism, the behavior is consistent across the broader distribution of positive-impact cases.

Analysis. These cases illustrate two distinct failure modes of the Static PnO baseline:

1. **Premature Commitment (Fig. 5a):** In the USDJPY case, the static plan allocated budget before the primary price drop occurred. The rolling model, observing the price evolution, successfully deferred execution to the optimal window.
2. **Lagged Execution (Fig. 5b):** In the USDCNY case, the static plan's timing was roughly correct but imprecise, missing the deepest point of the dip. The rolling model refined its target as the dip materialized, securing a lower weighted average price.

In both instances, the Constraint-Aware training ($\kappa = 0.7$ for USDJPY, $\kappa = 0.75$ for USDCNY) played a crucial role by forcing the model to spread bets, preventing it from locking into suboptimal positions early.

4 Conclusion

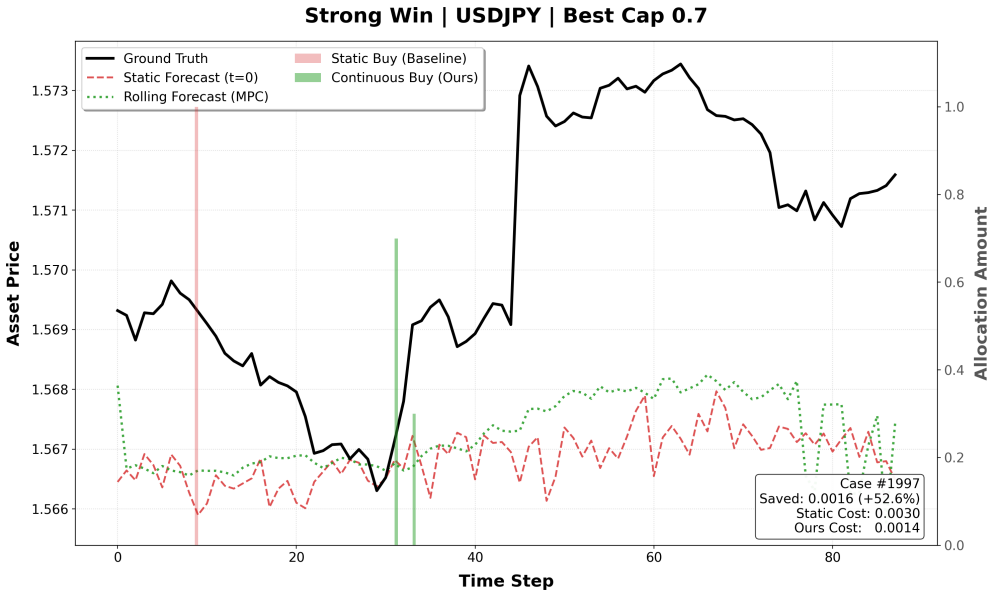
In this work, we introduced **Continuous RTS-PnO**, a unified framework for strategic fund allocation that integrates constraint-aware learning with rolling-horizon execution. By embedding real-world safety caps directly into the SPO+ training objective, we align the forecasting model's gradients with the feasible decision space.

Our extensive evaluation across 11 diverse financial datasets yields two counter-intuitive yet critical insights:

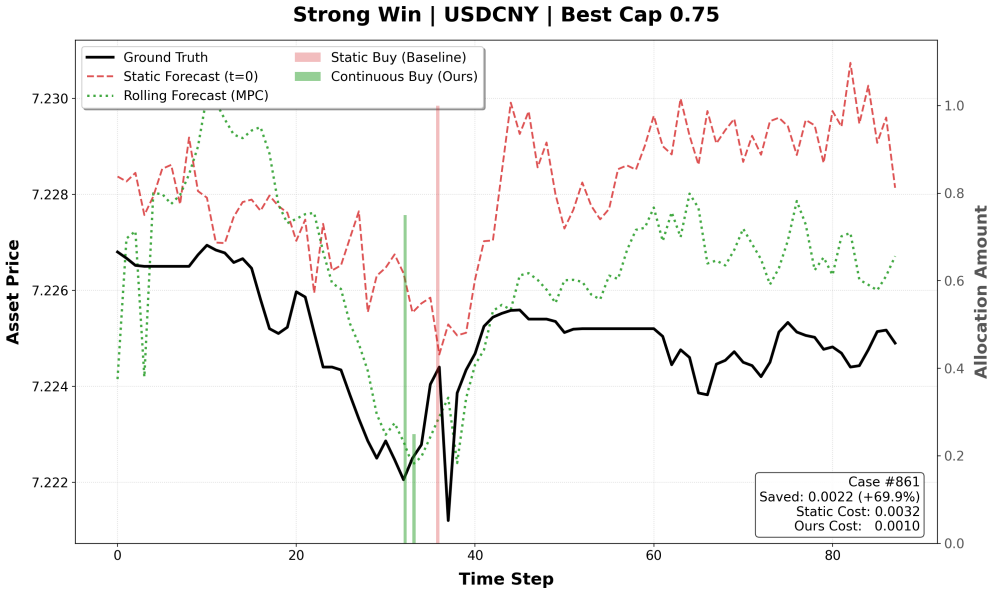
1. **The Volatility Paradox:** Contrary to the expectation that dynamic control excels in turbulent environments, we find that rolling-horizon strategies deliver the highest relative value in *low-volatility* regimes. In stable markets (empirically defined in our study as having a Coefficient of Variation $CV < 3\%$) with latent structural drifts, static models are prone to overconfidence, committing capital prematurely to exploit smooth trends. Our continuous framework serves as a necessary safeguard against this fragility, preserving liquidity for rare but impactful structural breaks.
2. **Asymmetric Advantage:** The performance gains don't stem from marginal daily improvements, but from an asymmetric risk profile. The model incurs a modest opportunity cost during normal periods to secure substantial savings during "Black Swan" events, effectively functioning as an algorithmic insurance mechanism.

These findings suggest a paradigm shift for neural fund allocation: rather than solely pursuing higher forecasting accuracy, research should prioritize optimizing execution dynamics to mitigate the "static traps" inherent in stable regimes.

However, several limitations present open frontiers for future research. First, while this study treats the safety cap κ as a tunable hyperparameter, real-world deployment requires frameworks to robustly adapt to strictly exogenous policy constraints. Second, our current formulation treats the market as a non-reactive environment; modeling market pushback, such as price slippage caused by large block executions, is a critical next step to align this framework with broader mechanism design principles. Finally, the challenge of high-volatility environments ($CV > 15\%$) remains. Promising avenues include incorporating *regime-switching priors* to better model multi-modal uncertainty in chaotic markets, as well



(a) **USDJPY (Trend Timing):** The Static model (Pink bar) commits the majority of its budget early at $t = 12$, locking in a price of ≈ 1.569. The Continuous model (Green bars) detects the developing downward trend and withholds liquidity. It executes heavily at $t = 34$, effectively catching the absolute local minimum (≈ 1.566) before the price skyrockets to 1.573 at $t = 48$. This demonstrates the value of dynamic "patience."



(b) **USDCNY (Dip Catching):** The market experiences a sharp, short-lived dip at $t = 33$. The Continuous model (Green) identifies this opportunity in real-time, deploying capital exactly at the trough ($t = 33, 34$). The Static model (Pink), relying on a $t = 0$ forecast, executes slightly later at $t = 36$, by which time the price has already rebounded. The Continuous model's ability to update its plan allows for superior temporal precision.

Figure 5: Visualizing Execution Dynamics. The Black line represents the ground truth asset price. Pink bars indicate the static allocation schedule ($t = 0$). Green bars indicate the realized rolling allocation.

as expanding empirical validation across multiple execution trajectories to rigorously bound execution robustness and standard deviation.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive feedback. This research was supported by the computing resources provided by the Mohamed bin Zayed University of Artificial Intelligence (MBZUAI).

References

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *Advances in neural information processing systems*, 32, 2019.

Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pp. 136–145. PMLR, 2017.

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *CoRR*, abs/2107.07511, 2021. URL https://arxiv.org/abs/2107.07511.

Stephen P. Boyd, Enzo Busseti, Steven Diamond, Ronald N. Kahn, Kwangmoo Koh, Peter Nystrup, and Jan Speth. Multi-period trading via convex optimization. *Found. Trends Optim.*, 3(1):1–76, 2017. doi: 10.1561/2400000023. URL https://doi.org/10.1561/2400000023.

Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Manag. Sci.*, 68 (1):9–26, 2022. doi: 10.1287/MNSC.2020.3922. URL https://doi.org/10.1287/mnsc.2020.3922.

International Monetary Fund. *Bangladesh: Selected Issues*. IMF Country Report No. 23/410. International Monetary Fund, Washington, DC, December 2023. ISBN 9798400260728. URL https://www.elibrary.imf.org/view/journals/002/2023/410/article-A001-en.xml.

International Monetary Fund. *Annual Report on Exchange Arrangements and Exchange Restrictions 2023*. Annual Report on Exchange Arrangements and Exchange Restrictions. International Monetary Fund, Washington, DC, December 2024. ISBN 9798400260391. doi: 10.5089/9798400260391.012. URL https://www.elibrary.imf.org/display/book/9798400260391/9798400260391.xml.

Fuyuan Lyu, Linfeng Du, Yunpeng Weng, Qiufang Ying, Zhiyan Xu, Wen Zou, Haolun Wu, Xiuqiang He, and Xing Tang. Timing is important: Risk-aware fund allocation based on time-series forecasting. In Luiza Antonie, Jian Pei, Xiaohui Yu, Flavio Chierichetti, Hady W. Lauw, Yizhou Sun, and Srinivasan Parthasarathy (eds.), *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.2, KDD 2025, Toronto ON, Canada, August 3-7, 2025*, pp. 4694–4704. ACM, 2025. doi: 10.1145/3711896.3737268. URL https://doi.org/10.1145/3711896.3737268.

Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Kai Wang, Sanket Shah, Haipeng Chen, Andrew Perrault, Finale Doshi-Velez, and Milind Tambe. Learning mdps from features: Predict-then-optimize for sequential decision making by reinforcement learning. *Advances in Neural Information Processing Systems*, 34:8795–8806, 2021.