When to Commit and When to Defer: Maturing Markov Decision Processes under Refining Information and Expiring Opportunities

Anonymous authors
Paper under double-blind review

Abstract

Sequential decisions often become better informed as action opportunities expire. We introduce *Maturing Markov Decision Processes* (MMDPs), a structured subclass of augmented finite-horizon Markov decision processes, for organizing when to commit and when to defer. The Expiring-Actions-First Principle provides a sufficient condition for deferral based on information gain and delay loss. This structure motivates stage-local policies, action abstraction, and policy-guided planning. Replenishment experiments show improved learning and test performance under the MMDP interface and, in the primary setting, a larger test-performance gain from forecast refinement. In synthetic cash management, MMDP improves PPO test returns across both network sizes; five-account equal-size mask controls show that action count alone does not explain its learned gain, and the advantage persists with action abstraction and policy-guided search. A production-scale cash-management case study adds complementary evidence. Together, these results support commitment timing as a useful inductive bias for learning and planning.

1 Introduction

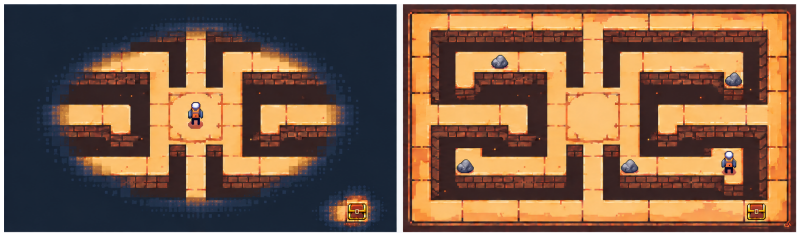


Figure 1: **Increasing information, expiring opportunities.** Early decisions use coarse information but greater flexibility; later decisions use richer information but fewer actions.

Many real-world sequential decision problems evolve in a structurally asymmetric way: as a decision cycle unfolds, the agent receives increasingly informative state signals through forecasts, observations, or operational updates, while the feasible action set contracts due to expiring opportunities, operational cutoffs, irreversible commitments, or resource constraints (see Figure 1). Such dynamics arise across domains including dynamic pricing (Netessine, 2006; Cheung et al., 2017), inventory control (Wang & Toktay, 2008; Liu et al., 2023; Xie et al., 2026), resource allocation (Ashlagi et al., 2018; Jacquillat & Li, 2024), and financial liquidity management (Pate-Cornell et al., 1990; Castro et al., 2025). Early decisions therefore operate under coarse information but high flexibility, whereas later decisions benefit from richer information but reduced optionality, creating a commitment–deferral trade-off.

A finite-horizon Markov decision process (MDP) can encode stage indices, forecast updates, and feasibility masks. This representation alone, however, does not organize decisions around which opportunities must be resolved before they expire and which choices can remain open for better information. We introduce *Maturing Markov Decision Processes* (MMDPs) to make this distinction explicit: *maturing* describes a process that becomes more informed while intervention opportunities progressively expire.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Our contributions are fourfold. First, we formalize MMDPs through strictly refining information and shrinking semantic action sets. Second, we establish the Expiring-Actions-First Principle under an information-gain versus delay-loss condition. Third, we develop structure-aware reinforcement learning with stage-local policies, action abstraction, and policy-guided planning. Fourth, replenishment experiments examine the learning benefits of deferral and the value of forecast updates, while synthetic cash-management experiments evaluate route selection, action abstraction, and planning; a production-scale case study provides complementary simulator, backtest, and deployment evidence.

2 Related Work

Standard and constrained MDPs. Classical and constrained MDPs model sequential control and feasibility requirements (Puterman, 2014; Sutton et al., 1998; Altman, 2021). An augmented finite-horizon MDP can encode stage indices, observations, and action masks. MMDPs do not extend this representational generality; they isolate the coupled refinement of information and expiration of action opportunities.

Partial observability. POMDPs address hidden-state inference and belief-based control (Kaelbling et al., 1998; Smallwood & Sondik, 1973; Spaan, 2012), while recurrent RL offers a practical approach to history-dependent decisions (Ni et al., 2021). MMDPs emphasize a complementary structure: how information revelation interacts with expiring opportunities, given an information state sufficient for control.

Information updates and decision timing. Procurement models analyze ordering flexibility under forecast updates (Donohue, 2000; Sethi et al., 2004). Related timing constraints appear in pricing with limited price changes (Cheung et al., 2017), online matching with departure deadlines (Ashlagi et al., 2018), and intraday payment decisions (Castro et al., 2025). These problems differ in how information arrives and opportunities disappear. MMDPs focus on their joint stage-level structure, establish a sufficient condition for action restriction, and evaluate that restriction as an interface for learning and planning.

Structured actions and search. Structured and parameterized-action RL exploit combinatorial or hybrid decision representations (Khalil et al., 2017; Nazari et al., 2018; Delarue et al., 2020; Hausknecht & Stone, 2015; Masson et al., 2016). MMDPs add a temporal distinction between expiring and persistent decisions. Policy-guided planning also relates to search-based policy improvement (Silver et al., 2017; Anthony et al., 2017); our controlled evaluation studies inference-time Search with frozen policies.

3 Maturing Markov Decision Processes

Maturing Markov Decision Processes (MMDPs) are a structured subclass of augmented finite-horizon MDPs in which information becomes richer while semantic action opportunities permanently expire.

3.1 Information–Action Asymmetry

Definition 3.1 (Maturing Markov Decision Process)**.** An MMDP is a finite-horizon Markov decision process with effective stages $n = 0, \dots, N$, specified by

$$\mathcal{M}^{\mathrm{M}} = \big(\mathcal{S}^\star, \mathcal{A}, \{\mathcal{S}_n, \phi_n, \mathcal{A}_n, P_n, r_n\}_{n=0}^{N}, \Gamma, \gamma\big),$$

whose information representations and structural action sets satisfy

$$\mathcal{S}_0 \prec \mathcal{S}_1 \prec \cdots \prec \mathcal{S}_N, \qquad \mathcal{A}_0 \supsetneq \mathcal{A}_1 \supsetneq \cdots \supsetneq \mathcal{A}_N. \tag{1}$$

Here $\phi_n : \mathcal{S}^\star \to \mathcal{S}_n$ maps a full-information description to the stage-n information state, and $\mathcal{A}_n \subseteq \mathcal{A}$ records unexpired opportunities in a common semantic action space. The information state together with n is sufficient for Markov control, including any required observable history and remaining horizon. Rewards $r_n(s, a)$ are defined on executable state–action pairs, and $\gamma \in (0, 1]$ discounts each underlying MDP transition. Together, P_n and Γ specify one transition kernel: P_n governs within-stage transitions, while Γ specifies boundary crossing and its successor law.

Strict information refinement. The relation $\mathcal{S}_n \prec \mathcal{S}_{n+1}$ means that the earlier representation $\phi_n(x)$ can be recovered measurably from $\phi_{n+1}(x)$, but not conversely on the support of the underlying descriptions. Later stages therefore retain the information available earlier while revealing additional decision-relevant distinctions.

Semantic action expiration. The relation $\mathcal{A}_n \supsetneq \mathcal{A}_{n+1}$ means that some opportunities permanently expire while every later opportunity is already represented earlier with the same semantic meaning. Physically later operations may be represented through valid reservations or registrations, but expired opportunities cannot reopen within the cycle. These structural sets describe stage-level expiration; resource-dependent changes in current executability are represented through the state.

Remark 3.1 (Non-Degenerate Stage Partition)**.** *A transition from stage n to stage $n+1$ defines a new effective stage only when $\mathcal{S}_n \prec \mathcal{S}_{n+1}$ and $\mathcal{A}_n \supsetneq \mathcal{A}_{n+1}$.*

The strict action contraction must include the expiration of at least one domain-action opportunity, rather than only a stage-termination control. If either condition fails, the two stages are treated as a single effective stage, with the intervening events retained in the underlying dynamics. Appendix A.1 provides further interpretation.

Commitment–Deferral Trade-Off. Acting earlier preserves more opportunities but commits under coarser information; waiting permits better-informed choice while forfeiting expiring opportunities. The resulting commitment–deferral trade-off implies neither that every expiring action should be executed nor that waiting is always optimal.

3.2 Expiring-Actions-First Principle

For the local analysis, consider the product-refinement case

$$\mathcal{S}_{n+1} = \mathcal{S}_n \times \mathcal{Z}_{n+1}, \qquad n = 0, \dots, N-1,$$

where $\mathcal{Z}_{n+1}$ is the space of information newly revealed at stage $n+1$. Under shrinking action sets, we define the expiring action set by

$$\mathcal{D}_n := \mathcal{A}_n \setminus \mathcal{A}_{n+1}, \qquad n = 0, \dots, N-1.$$

Thus, the stage-n action space splits into expiring actions $\mathcal{D}_n$, whose opportunities disappear at stage $n+1$, and persistent actions $\mathcal{A}_{n+1} \subsetneq \mathcal{A}_n$, whose opportunities remain open.

The expiring set contains the executable stage-termination action $a_n^{\mathrm{end}} \in \mathcal{D}_n$, which defers unresolved persistent decisions to stage $n+1$. For the local comparison below, suppose that executing a_n^{end} at $s \in \mathcal{S}_n$ produces $s' = (s, z) \sim \Gamma(\cdot \mid s)$ in one MDP transition, where $z \in \mathcal{Z}_{n+1}$. For the persistent actions, we compare three alternatives: selecting and executing an action now, selecting an action before observing z but executing it at the next stage, and selecting and executing an action after observing z. Their expected discounted returns, all measured from s and allowing optimal continuation after the specified action, are

$$J_{\mathrm{now}} := \max_{a \in \mathcal{A}_{n+1}} Q_n^\star(s, a), \tag{2}$$

$$J_{\mathrm{delay}} := r_n(s, a_n^{\mathrm{end}}) + \gamma \max_{a \in \mathcal{A}_{n+1}} \mathbb{E}_z\left[Q_{n+1}^\star((s, z), a)\right], \tag{3}$$

$$J_{\mathrm{adapt}} := r_n(s, a_n^{\mathrm{end}}) + \gamma \mathbb{E}_z\left[\max_{a \in \mathcal{A}_{n+1}} Q_{n+1}^\star((s, z), a)\right]. \tag{4}$$

Here, $Q^\star$ denotes the optimal discounted action value of the full model, and the expectation uses the successor distribution induced by Γ. The three maximizations use the same nonempty set consisting of all persistent actions executable at s; each of these actions is assumed to remain executable after ending the stage.

Information Gain. At state s, the information revealed at the next stage can improve the choice of a persistent action. We define this expected gain by comparing selection after observing z with selection before observing z, with execution delayed in both cases:

$$\delta_n(s) := J_{\mathrm{adapt}} - J_{\mathrm{delay}} \geq 0. \tag{5}$$

Nonnegativity follows from $\mathbb{E}[\max] \geq \max \mathbb{E}$.

Delay Cost. Let $c_n(s) \geq 0$ uniformly bound the loss from delaying any fixed persistent action, with optimal continuation and all discounted returns measured from s. Then

$$J_{\text{now}} - J_{\text{delay}} \leq c_n(s). \tag{6}$$

These quantities give $J_{\text{adapt}} - J_{\text{now}} \geq \delta_n(s) - c_n(s)$; ending the stage permits the adaptive alternative, so $Q_n^\star(s, a_n^{\text{end}}) \geq J_{\text{adapt}}$.

Theorem 3.1 (Expiring-Actions-First Principle)**.** *Under the preceding conditions, if $c_n(s) \leq \delta_n(s)$, then, maximizing over actions executable at s,*

$$\max_{a \in \mathcal{A}_n} Q_n^\star(s, a) = \max_{a \in \mathcal{D}_n} Q_n^\star(s, a). \tag{7}$$

Hence an optimal decision at s can be selected from $\mathcal{D}_n$

$$\pi_n^\star(s) \in \arg\max_{a \in \mathcal{D}_n} Q_n^\star(s, a).$$

The principle gives a local action-set reduction: under its sufficient condition, persistent decisions can remain open without excluding an optimal current choice. It does not require executing every expiring action or always waiting. When $c_n(s) = 0$, the condition holds automatically. A detailed proof appears in Appendix A.2.

4 Structure-Aware Reinforcement Learning for MMDPs

We use the maturation structure to organize policy decisions, then extend the resulting interface with action abstraction and bounded planning.

4.1 Semantic Stage-Local Policy Interface

Theorem 3.1 motivates the stage-local policy interface

$$\operatorname{supp} \pi_\theta(\cdot \mid s, n) \subseteq \begin{cases} \mathcal{D}_n, & n < N, \\ \mathcal{A}_N, & n = N, \end{cases} \tag{8}$$

with only executable actions eligible. For $n < N$, choosing $a_n^{\text{end}} \in \mathcal{D}_n$ advances through Γ, forfeiting unresolved expiring opportunities while leaving persistent decisions open.

The basic *MMDP* variant retains policy-selected action parameters, whereas *Flat* exposes all executable actions in $\mathcal{A}_n$. Both share observations, the stage indicator, and the return objective. Our cash-management implementations use interface-dependent action encodings; their mappings and comparison scope are detailed in Appendix B.1.

4.2 Action Abstraction and State-Dependent Execution

Restricting the current decision interface to $\mathcal{D}_n$ focuses the policy on expiring opportunities, but these actions may still be continuous or hybrid. Action abstraction (AA) further simplifies this stage-local action space: an abstract action $\tilde{a}$ specifies *which* expiring decision to resolve, while a state-dependent executor $E_n(s, \tilde{a}) \in \mathcal{A}_n$ supplies its parameters.

In cash management, the policy selects a transfer edge (u, v) or the stage-termination action, and the executor supplies the amount ratio ρ in $a = (u, v, \rho)$. Flat-AA and MMDP-AA share this execution rule under their respective interfaces. Because the executor may omit an optimal concrete action, AA can introduce additional approximation error. Execution details appear in Appendix B.2.

4.3 Policy-Guided Search

The stage-local interface also provides a search space for refining the learned policy's current choice. The policy guides candidate construction and evaluation order, while an action-value estimate compares their long-term consequences. For the abstract interface, search selects

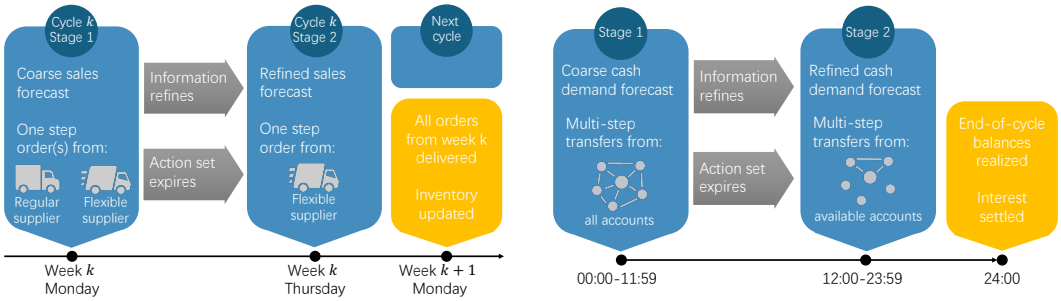
$$\tilde{a}^{\text{search}} \in \arg\max_{\tilde{a} \in \mathcal{C}_B(s,n;\theta)} \widehat{Q}(s, n, \tilde{a}), \tag{9}$$

where $\mathcal{C}_B$ is the nonempty set of candidates scored within a planning budget B. Here $\widehat{Q}$ estimates the return from executing $E_n(s, \tilde{a})$ and continuing with the frozen policy.

In the controlled benchmarks, Search scores candidates using only current observations and public model information, executes the selected action, and replans after the next observation. Implementation and budget details appear in Appendix B.3.

5 Experiments

We test whether organizing commitments and deferrals improves learning and planning as information is refined and opportunities expire. The experiments span three decision timelines. A replenishment episode contains repeated weekly cycles, each with two stages and one ordering decision per stage. A synthetic cash episode contains one daily cycle with two stages and multiple transfer decisions per stage. A production-scale cash episode contains one daily cycle with 24 hourly stages and multiple transfers per stage. Replenishment examines learning under safe deferral and forecast refinement; synthetic cash examines semantic action selection, abstraction, and bounded planning; the production-scale case study adds simulator, backtest, and human-reviewed deployment evidence. Figure 2 depicts one cycle in each controlled domain.



(a) Replenishment problem.

(b) Synthetic cash management problem.

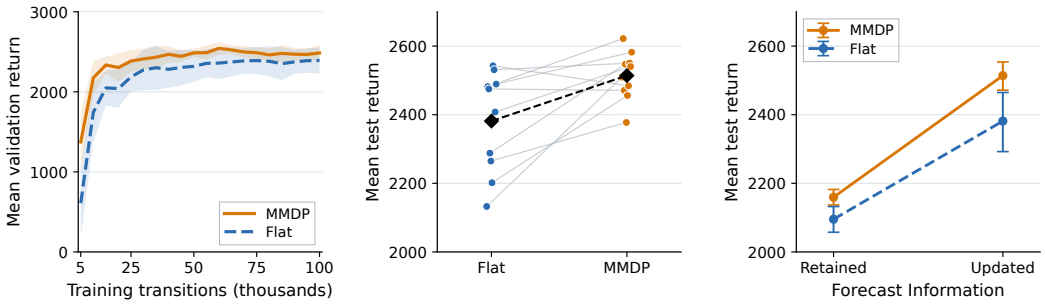
Figure 2: Information refinement and opportunity expiration within one decision cycle in each controlled domain. **(a)** Replenishment: the regular-supplier ordering window closes before the forecast update, while flexible ordering remains available. **(b)** Synthetic cash management: forecasts update as transfer routes expire.

5.1 Staged Replenishment

We study staged replenishment, motivated by recent RL applications in inventory control (Liu et al., 2023; Xie et al., 2026), dual-sourcing models (Veeraraghavan & Scheller-Wolf, 2008; Yee et al., 2024), and procurement with forecast updates (Donohue, 2000; Sethi et al., 2004; Federgruen et al., 2026). The cheaper regular supplier closes before the forecast update, while the more expensive flexible supplier remains available. Early and late orders arrive together the following Monday. The flexible supplier's price and shared capacity are unchanged across the two decisions, so a fixed feasible flexible order can be deferred without changing its cost, capacity use, or delivery date.

Flat permits early orders from either supplier; MMDP defers flexible orders. Both use PPO (Schulman et al., 2017) with identical observation fields, including remaining flexible capacity, and matched networks, action parameterization, and training budgets. All policies are trained from scratch. A control retains the coarse forecast at the later stage. Environment, training, and evaluation protocols appear in Appendix C.1.

Learning and generalization. Figure 3a shows a higher mean validation return for MMDP from the first recorded evaluation through the end of training. The advantage extends to held-out instances (Figure 3b): MMDP improves mean test return by 132.5 (95% CI $[56.6, 213.0]$). These results support a finite-budget learning and generalization advantage under the evaluated interfaces.



(a) Learning dynamics (b) Test performance (c) Forecast updating

Figure 3: Learning and forecast updating in staged replenishment. **(a)** Updated-condition validation returns: means and one standard deviation across ten training seeds. **(b)** Validation-selected policies evaluated on 200 shared test instances: colored points show seed-level means, gray lines pair seeds, and the black dashed line connects overall means. **(c)** Mean test returns with retained or updated forecasts; error bars show marginal paired-bootstrap 95% CIs over training seeds.

Value of forecast refinement. We compare separately trained policies with and without the later refined signal. At the main setting, the refined signal correctly identifies the delivery-week demand category with 80% probability. Figure 3c shows that policies trained and evaluated with this signal gain 286.1 in mean test return for Flat and 354.5 for MMDP relative to their separately trained Retained counterparts. This exploratory comparison suggests that information refinement can amplify the benefit of the MMDP interface, although the interaction depends on fine-signal accuracy. Complete results and accuracy-sensitivity analyses appear in Appendix C.2.

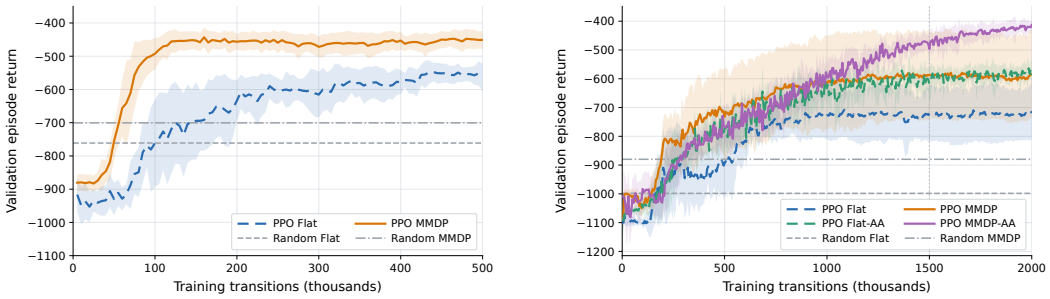
5.2 Synthetic Cash Management

We next study cash management, where liquidity protection, transfer costs, and investment yield must be balanced under uncertain demand (Baumol, 1952; Miller & Orr, 1966; Pate-Cornell et al., 1990; Gormley & Meade, 2007; Ferstl & Weissensteiner, 2010). Our synthetic five- and ten-account networks contain master, investment, and operating accounts, drawing on multi-account cash-management models with networked transfers (Salas-Molina et al., 2021; Bhaya & Kaszkurewicz, 2023). The two-stage daily cycle stylizes intraday liquidity forecasting and time-limited transfer windows (European Central Bank, 2026): forecasts refine and routes expire, transfers change balances immediately, and demand settles at cycle end. Environment specifications and comparison scope appear in Appendix D.1.

5.2.1 Five-account setting.

We compare PPO, a random policy (Random), and a forecast-based rolling-horizon mixed-integer linear programming (MILP) controller under the Flat and MMDP interfaces. PPO uses matched observations, stage information, networks, route–amount parameterization, and training budgets. Random applies the same sampling rule over the routes and termination action available under each interface. The MILP controller is a benchmark-specific implementation of the deterministic rolling-horizon approach (Perkins & Powell, 2017), using current forecasts and public model information; controller implementations and training settings are documented in Appendix D.2.

Learning and independent-test performance. Figure 4a shows an MMDP–Flat separation during PPO training. On independent test instances (Table 1), MMDP improves PPO return over Flat by 117.725 (95% CI $[85.237, 151.115]$). Random also improves by 69.515 (95% CI $[64.246, 74.890]$). This non-learning gain shows that the MMDP interface can improve outcomes without policy training in this benchmark. MILP attains the highest mean test return (-330.286) and selects identical transfers under Flat and MMDP. Thus, the MMDP deferral restriction causes no observed loss for this controller on this test bank, but this is not a general losslessness guarantee. On the separate 20-instance single-thread CPU subset, measured controller time per episode is lower under MMDP by approximately 6.5% for PPO, 15.8% for Random, and 11.8% for MILP.



(a) Five-account setting

(b) Ten-account setting

Figure 4: Raw-policy learning in cash management. Curves show mean validation returns over ten training seeds, with one standard deviation shaded.

Table 1: Five-account results. Controller cells report mean test return on 200 shared instances, with controller time in parentheses below (ms/episode; 20 shared instances, single-thread CPU). Mask averages five equal-size funding-preserving masks. Contrast cells report paired return differences with 95% CIs below.

	Interface comparison			Mask comparison	
Controller	Flat (ms/episode)	MMDP (ms/episode)	MMDP − Flat [95% CI]	Mask (ms/episode)	MMDP − Mask [95% CI]
PPO	−552.003 (0.581)	−434.278 (0.543)	+117.725 [85.237, 151.115]	−486.853 (0.610)	+52.575 [41.175, 64.955]
Random	−789.646 (0.190)	−720.130 (0.160)	+69.515 [64.246, 74.890]	−721.888 (0.160)	+1.758 [−1.107, 4.687]
MILP	−330.286 (2.255)	−330.286 (1.988)	0.000 [0.000, 0.000]	−330.286 (2.082)	0.000 [0.000, 0.000]

Route selection beyond action count. To test whether action count alone explains the learned interface gain, we compare MMDP with five equal-size masks that preserve all three direct funding routes. In the Mask comparison of Table 1, MMDP PPO gains 52.575 (95% CI $[41.175, 64.955]$) over the mask average, whereas Random shows no clear separation ($+1.758$, 95% CI $[-1.107, 4.687]$). Thus, for PPO in this benchmark, matching action count and direct-funding coverage does not eliminate MMDP's advantage; the comparison does not isolate expiration semantics from other route and decoding differences. Complete evaluation protocols, controller and mask results, and interpretation limits appear in Appendix D.3.

5.2.2 Ten-account setting.

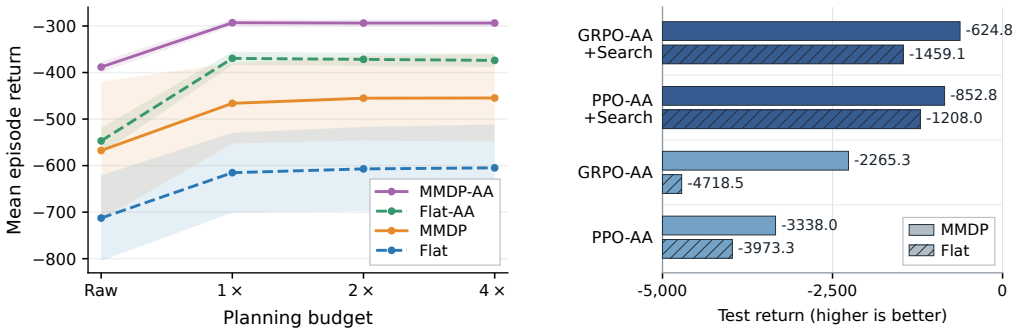
Compared with the five-account setting, the ten-account network expands the early-stage route catalogue from 14 to 74 routes, including 16 rather than 6 routes that expire at the stage boundary. We use this larger action-selection space to examine action abstraction (AA; Section 4.2), policy-guided search (Section 4.3), and planning-budget sensitivity. Controller implementations and training settings appear in Appendix D.4.

Learning with action abstraction. We train Flat, MMDP, Flat-AA, and MMDP-AA with matched observations, hidden-layer widths, PPO settings, and training budgets. Figure 4b shows two distinct learning patterns. Without AA, MMDP starts above Flat and maintains a validation-return advantage throughout training. With AA, the two policies begin at similar returns, but MMDP-AA pulls ahead as training proceeds. One possible interpretation is that AA initially eases action execution for both interfaces, while the benefit of selecting among stage-appropriate opportunities becomes more visible as the policies learn.

Test interface gaps. Table 2 reports Raw MMDP-minus-Flat test gaps under each controller. The gains are 145.09 for PPO and 158.63 for PPO-AA, showing that the MMDP advantage persists

Table 2: Ten-account Raw MMDP-minus-Flat test gaps. Paired 95% CIs resample training seeds for PPO and instances for Random/MILP; Random averages ten action draws per instance.

PPO	PPO-AA	Random	MILP
$+145.09$ [45.27,228.85]	$+158.63$ [143.50,173.97]	$+128.64$ [121.29,136.12]	$+10.02$ [6.58,13.85]



(a) Planning-budget sensitivity

(b) Production-scale comparison

Figure 5: **(a)** Ten-account returns for the policies and instances in Table 3; $1\times = 576$ simulated transitions per decision, and Raw uses no Search. Shading is the across-seed SD. **(b)** Five-seed mean returns in the 27-account, 24-stage simulator; begin at $-5{,}000$; higher is better.

with and without AA on the held-out test instances. Random also improves by 128.64. Unlike in the five-account setting, MMDP raises the ten-account MILP's mean test return by 10.02 (95% CI $[6.58, 13.85]$). Since MILP obtains the highest mean returns of -206.29 under MMDP, it shows that MMDP can also add value to a strong planning baseline.

Policy-guided search. We apply policy-guided search to the same frozen final policies on the same test instances, without further training or checkpoint selection. Candidate-generation rules and candidate-count limits are shared within each AA and non-AA interface pair. Search executes the selected action and replans under the same per-decision simulation cap for all four policy families, although realized planning work may differ.

Table 3: Ten-account independent-test mean returns for the same frozen policies on 200 shared instances. Raw and Search brackets show marginal bootstrap 95% CIs over ten training seeds; Search-gain brackets show paired-bootstrap 95% CIs for within-seed differences. Search uses the primary planning budget. Higher return is better.

Policy	Raw	Search	Search gain
Flat	-712.64 [-765.65,-654.86]	-604.80 [-656.28,-547.31]	$+107.84$ [94.73,120.96]
MMDP	-567.55 [-665.37,-509.30]	-454.69 [-517.07,-414.68]	$+112.85$ [87.24,151.90]
Flat-AA	-546.93 [-564.09,-531.37]	-373.88 [-381.99,-365.32]	$+173.05$ [156.05,191.69]
MMDP-AA	-388.30 [-395.33,-381.45]	-293.69 [-298.21,-289.54]	$+94.61$ [88.65,100.74]

Table 3 shows Search gains for all four families, positive in every training-seed pair. MMDP-AA with Search achieves the highest mean return among the four evaluated learned policy families.

Planning-budget sensitivity. We vary the per-decision planning cap over $1\times$, $2\times$, and $4\times$ a fixed base budget, keeping the policies, test instances, and candidate-generation rules unchanged. Figure 5a shows substantial gains for both AA controllers at $1\times$, but no consistent further improvement as the cap increases. MMDP-AA has the highest mean return among the four policy families at each evaluated cap. Complete ten-account evaluation protocols, paired comparisons, budget results, and computational accounting appear in Appendix D.5.

5.3 PRODUCTION-SCALE CASH MANAGEMENT

We next examine the methods in a production-scale cash-management system with 27 accounts and 24 hourly decision stages per day. Unlike the two-stage synthetic benchmarks, this setting involves hourly updates to balances, forecasts, and network availability. We combine simulator comparisons, a pre-deployment historical backtest, and observations from human-reviewed deployment to examine performance, estimated economic value, and operational uptake. The production simulator's environment, evaluation protocol, and results appear in Appendix E.1.

Simulator evaluation. We compare Flat and MMDP interfaces with PPO-AA and GRPO-AA, each with and without Search. GRPO-AA forms group-relative advantages from trajectories sharing an initial condition, without a learned value baseline (Shao et al., 2024); Search scores candidate plans under the current forecast and replans after new observations. Figure 5b reports five-seed means on a shared simulator test set generated from held-out operational source data. MMDP has higher means in all four matched pipeline comparisons. Under Flat, PPO-AA exceeds GRPO-AA both Raw and with Search; under MMDP, this ordering reverses, and Search-assisted MMDP-GRPO-AA attains the highest mean. This reversal suggests that the relative performance of PPO-AA and GRPO-AA depends on the decision interface in this simulator.

Pre-deployment backtest and human-reviewed deployment. A historical backtest spanning six calendar months and 94 eligible business days compares the Search-assisted MMDP-GRPO-AA controller with the executed incumbent workflow, including human adjustments, under the same external cash flows and constraints. Mean daily relative net-income improvement is 8.25% (approximate 95% CI: $[3.80\%, 12.70\%]$), where net income comprises overnight deposit interest minus shortfall financing costs and penalties.

After deployment of a separate operational model, mean daily amount-based adoption increases from 23.92% in the pre-deployment period to 72.87% across 86 eligible days, spanning three complete months and a partial fourth month. Adoption measures the share of executed transfer amount attributable to accepted algorithmic recommendations, not realized financial benefit. The before–after comparison does not isolate a causal deployment effect. Model scope, definitions, monthly results, and aggregation details appear in Appendix E.2.

6 CONCLUSION

Maturing Markov Decision Processes make commitment timing explicit when information is refined as action opportunities expire. The Expiring-Actions-First Principle gives a sufficient information-gain-versus-delay-loss condition for deferring persistent actions without loss. This structure motivates a stage-local policy interface that can be combined with action abstraction and policy-guided search.

In staged replenishment, MMDP improves finite-budget learning and held-out performance. Forecast updating helps both interfaces; the primary setting suggests that MMDP can amplify the benefit of refined information. In synthetic cash management, MMDP improves PPO, Random, and ten-account MILP results; equal-size mask controls show that route count and direct-funding coverage do not fully explain the five-account PPO gain. The evaluated AA encodings retain an MMDP advantage, and Search improves all four frozen policy families. The production case adds descriptive simulator, backtest, and human-reviewed adoption evidence.

Several limitations qualify these findings. The Expiring-Actions-First guarantee depends on a problem's information and delay-cost structure, so applying the same restriction elsewhere requires checking those conditions. In the production system, network-coupled transfers required a system-wide transition rather than an account-level A/B test; the deployment observations therefore describe adoption, not a causal revenue effect. Evaluating the approach under other information structures and network configurations remains an important next step.

AI USE STATEMENT

We utilize tools based on large language models to assist with literature searches, language editing, refining theoretical formulations and proofs, experimental diagnosis, implementation and debugging,

and the creation of figures and tables. Controlled benchmark instances are sampled by programmatic simulators; AI-assisted work also includes the development of experimental software and the verification of controlled experiments. These verifications do not replace the authors' validation of mathematical arguments, references, and the interpretation of results. Authors bear full responsibility for the final manuscript, including all text, assertions, code, and experimental results generated with the assistance of artificial intelligence.

ETHICS STATEMENT

The controlled replenishment and cash-management experiments use synthetic operational environments. Production evidence is presented through aggregate results and system descriptions; raw operational records and identifying account or institution details are not released. The deployed workflow retains human review of model-generated recommendations. Retrospective backtest gains and observed recommendation adoption should not be interpreted as causal evidence of increased realized revenue. Errors in forecasts, models, or optimization can cause liquidity shortfalls and financial losses. Operational use therefore requires application-specific validation, enforceable feasibility and risk constraints, human oversight, and ongoing monitoring; performance in the studied environments does not establish safety or effectiveness in other operational settings.

REPRODUCIBILITY STATEMENT

The assumptions and proof of the Expiring-Actions-First Principle are provided in Appendix A.2, and implementation details are given in Appendix B. The replenishment environment, training protocol, and complete results are documented in Appendix C. Cash-management environment specifications appear in Appendix D.1, with five-account methods and evaluations and ten-account learning, Search, and budget protocols in Appendices D.2, D.3, D.5.1, D.5.3, and D.5.5. The cash-management appendix also documents the ten-account Random and MILP references and serial runtime measurements. These descriptions distinguish training, validation, and independent testing, and specify the replication units used for uncertainty estimation. The complete production-derived simulator and underlying operational records cannot be released because of confidentiality restrictions. Appendix E.1 describes that setting, but the production results are not fully independently reproducible from the disclosed materials.

REFERENCES

Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Itai Ashlagi, Maximilien Burq, Chinmoy Dutta, Patrick Jaillet, Amin Saberi, and Chris Sholley. Maximum weight online matching with deadlines. *arXiv preprint arXiv:1808.03526*, 2018.

William J Baumol. The transactions demand for cash: An inventory theoretic approach. *The Quarterly journal of economics*, 66(4):545–556, 1952.

Dimitri P. Bertsekas, John N. Tsitsiklis, and Cynara Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3:245–262, 1997. URL `https://www.mit.edu/~jnt/Papers/J066-97-rollout.pdf`.

Amit Bhaya and Eugenius Kaszkurewicz. The generalized cash balance problem: optimization-based one step ahead optimal control. *International Transactions in Operational Research*, 30(6): 3993–4025, 2023.

Pablo Castro, Ajit Desai, Han Du, Rodney Garratt, and Francisco Rivadeneyra. Estimating policy functions in payment systems using reinforcement learning. *ACM Transactions on Economics and Computation*, 13(1):1–31, 2025.

Wang Chi Cheung, David Simchi-Levi, and He Wang. Dynamic pricing and demand learning with limited price experimentation. *Operations Research*, 65(6):1722–1731, 2017.

Arthur Delarue, Ross Anderson, and Christian Tjandraatmadja. Reinforcement learning with combinatorial actions: An application to vehicle routing. *Advances in Neural Information Processing Systems*, 33:609–620, 2020.

Karen L. Donohue. Efficient supply contracts for fashion goods with forecast updating and two production modes. *Management Science*, 46(11):1397–1411, 2000. doi: 10.1287/mnsc.46.11.1397.12088.

European Central Bank. Information guide for TARGET participants: Part 2—CLM and RTGS. Version R2026.JUN, June 2026. URL https://www.ecb.europa.eu/paym/target/consolidation/profuse/shared/pdf/Infoguide-R2026.JUN-Part-2-CLM-and-RTGS.pdf.

Awi Federgruen, Zhe Liu, and Jiaqi Lu. Sourcing with demand updates. *Management Science*, 2026.

Robert Ferstl and Alex Weissensteiner. Cash management using multi-stage stochastic programming. *Quantitative Finance*, 10(2):209–219, 2010.

Fionnuala M Gormley and Nigel Meade. The utility of cash flow forecasts in the management of corporate cash balances. *European journal of operational research*, 182(2):923–935, 2007.

Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *arXiv preprint arXiv:1511.04143*, 2015.

Qi Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5.

Alexandre Jacquillat and Michael Lingzhi Li. Learning to cover: online learning and optimization with irreversible decisions. *arXiv preprint arXiv:2406.14777*, 2024.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

Jiaxi Liu, Shuyi Lin, Linwei Xin, and Yidong Zhang. Ai vs. human buyers: A study of alibaba's inventory replenishment system. *INFORMS Journal on Applied Analytics*, 53(5):372–387, 2023.

Warwick Masson, Pravesh Ranchod, and George Konidaris. Reinforcement learning with parameterized actions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Merton H Miller and Daniel Orr. A model of the demand for money by firms. *The Quarterly journal of economics*, 80(3):413–435, 1966.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

Serguei Netessine. Dynamic pricing of inventory/capacity with infrequent price changes. *European Journal of Operational Research*, 174(1):553–580, 2006.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent model-free rl can be a strong baseline for many pomdps. *arXiv preprint arXiv:2110.05038*, 2021.

M Elisabeth Pate-Cornell, George Tagaras, and Kathleen M Eisenhardt. Dynamic optimization of cash flow management decisions: a stochastic model. *IEEE Transactions on Engineering Management*, 37(3):203–212, 1990.

Raymond T. Perkins, III and Warren B. Powell. Stochastic optimization with parametric cost function approximations, 2017. URL https://arxiv.org/abs/1703.04644.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Francisco Salas-Molina, Juan A Rodriguez-Aguilar, David Pla-Santamaria, and Ana García-Bernabeu. On the formal foundations of cash management systems: F. salas-molina et al. *Operational Research*, 21(2):1081–1095, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Suresh P. Sethi, Houmin Yan, and Hanqin Zhang. Quantity flexibility contracts: Optimal decisions with information updates. *Decision Sciences*, 35(4):691–712, 2004. doi: 10.1111/j.1540-5915.2004.02873.x.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973. doi: 10.1287/opre.21.5.1071.

Matthijs TJ Spaan. Partially observable markov decision processes. In *Reinforcement learning: State-of-the-art*, pp. 387–414. Springer, 2012. doi: 10.1007/978-3-642-27645-3_12.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

Senthil Veeraraghavan and Alan Scheller-Wolf. Now or later: A simple policy for effective dual sourcing in capacitated systems. *Operations Research*, 56(4):850–864, 2008.

Tong Wang and Beril L Toktay. Inventory management with advance demand information and flexible delivery. *Management Science*, 54(4):716–732, 2008.

Yaqi Xie, Xinru Hao, Jiaxi Liu, Will Ma, Linwei Xin, Lei Cao, and Yidong Zhang. Deepstock: Reinforcement learning with policy regularizations for inventory management. *arXiv preprint arXiv:2603.19621*, 2026.

Hannah Yee, Heletjé E van Staden, and Robert N Boute. Dual sourcing under non-stationary demand and partial observability. *European Journal of Operational Research*, 314(1):94–110, 2024.

A Additional Details on MMDP Structure

A.1 Interpretation of Non-Degenerate Stages

Remark 3.1 specifies which boundaries define distinct effective stages in the MMDP abstraction. A new effective stage requires both strict information refinement and the permanent expiration of at least one domain-action opportunity. A transition exhibiting only one component does not define a separate effective stage; its events, rewards, and state transitions remain part of the underlying MDP. Treating such events as belonging to the same effective stage does not mean deleting them or replacing several underlying transitions with a single transition.

This partition does not imply that waiting is optimal whenever opportunities remain open, or that immediate execution is optimal whenever no new information arrives. Rewards, resource availability, and other state consequences may still depend on timing. The partition identifies the joint maturation structure, while Theorem 3.1 gives a separate, conditional criterion for deferring persistent actions.

A.2 Details and Proof of the Expiring-Actions-First Principle

This subsection makes the comparison set and delay-cost definition in Section 3.2 explicit and proves Theorem 3.1. Fix a stage $n < N$ and a state s satisfying the local comparison conditions. Assume bounded returns, attained optimal continuations, and attained measurable maxima, so that the values and decision alternatives below are well defined.

The common comparison set. Structural action sets record unexpired opportunities, while executability may additionally depend on the current state. Define

$$B = B_{n+1}(s) := \{a \in \mathcal{A}_{n+1} : a \text{ is executable at } s\}. \tag{10}$$

The local feasibility assumption requires that B be nonempty and that there be a set of successor states of probability one under $\Gamma(\cdot \mid s)$ on which every action in B remains executable. All three quantities J_{now}, J_{delay}, and J_{adapt} use this same set. Observing z allows the adaptive alternative to select among these actions, but does not enlarge the comparison set.

If every action in $\mathcal{A}_{n+1}$ is executable at s, then $B = \mathcal{A}_{n+1}$. Otherwise, the maximizations over $\mathcal{A}_{n+1}$ in the three main-text definitions are understood as maximizations over B. The full next-stage value $V^\star_{n+1}$ may optimize over additional actions that become executable only at the successor state.

Transitions and value scales. Executing a_n^{end} produces $(s, z) \sim \Gamma(\cdot \mid s)$ in one underlying MDP transition, as assumed in the local comparison. All expectations below use this distribution at the fixed current state s. Write $r_e = r_n(s, a_n^{\text{end}})$ for the expected immediate reward of ending the stage. The value $Q^\star_{n+1}((s, z), a)$ is measured from the next-stage decision state; multiplying it by γ and adding r_e expresses the delayed alternative on the same scale as $Q^\star_n(s, a)$.

The next-stage action value specifies only the first next-stage action. Subsequent decisions use the full model and may depend on the revealed information. Thus, J_{delay} selects its first next-stage action before observing z, while J_{adapt} selects that action after observing z. Both permit optimal continuation thereafter.

Information gain. The two delayed alternatives share the same stage-termination transition and reward. Consequently,

$$\begin{aligned}\delta_n(s) &= J_{\text{adapt}} - J_{\text{delay}} \\ &= \gamma \left[\mathbb{E}_z \left[\max_{a \in B} Q^\star_{n+1}((s, z), a) \right] - \max_{a \in B} \mathbb{E}_z \left[Q^\star_{n+1}((s, z), a) \right] \right].\end{aligned} \tag{11}$$

For every fixed $a \in B$, the pointwise maximum inside the first expectation is at least $Q^\star_{n+1}((s, z), a)$. Taking expectations and then maximizing over the fixed action gives $\delta_n(s) \geq 0$. This quantity measures the gain from adapting the first next-stage action, rather than the net advantage of waiting over acting now. Strict information refinement need not yield a strictly positive gain: the same action may remain optimal across all relevant realizations.

Delay cost. For a fixed action $a \in B$, the expected discounted return from executing it immediately and continuing optimally is $Q^\star_n(s, a)$. The return from first ending the stage, then executing the same a as the first next-stage action, and continuing optimally is

$$r_e + \gamma \mathbb{E}_z \left[Q^\star_{n+1}((s, z), a) \right].$$

Define the signed return loss from delaying this fixed action as

$$\ell_n(s, a) := Q^\star_n(s, a) - \left[r_e + \gamma \mathbb{E}_z \left[Q^\star_{n+1}((s, z), a) \right] \right]. \tag{12}$$

This loss may be negative if delaying the fixed action improves its return. A valid delay-cost bound is any finite $c_n(s) \geq 0$ satisfying

$$c_n(s) \geq \sup_{a \in B} \left[\ell_n(s, a) \right]_+, \qquad [u]_+ := \max\{u, 0\}. \tag{13}$$

This is the precise meaning of the uniform upper bound used in the main text.

For every $a \in B$, the bound implies

$$Q_n^\star(s,a) \le r_e + \gamma \mathbb{E}_z\big[Q_{n+1}^\star((s,z),a)\big] + c_n(s).$$

Maximizing over the same set B gives

$$\begin{aligned} J_{\text{now}} &\le r_e + \gamma \max_{a\in B} \mathbb{E}_z\big[Q_{n+1}^\star((s,z),a)\big] + c_n(s) \\ &= J_{\text{delay}} + c_n(s), \end{aligned} \tag{14}$$

which establishes (6).

The loss in (12) includes the complete effect of delay on discounted return, including changes in rewards, resources, state consequences, and subsequent opportunities. A model-specific bound on these effects is admissible whenever it satisfies (13). An immediate execution-fee difference alone need not provide such a bound. In particular, the absence of an explicit waiting fee does not imply $c_n(s) = 0$; zero is valid when delaying every action in B weakly preserves its optimal return. The existence of a bound does not guarantee that it satisfies the theorem's threshold $c_n(s) \le \delta_n(s)$.

The value of deferral. Using delayed execution of a preselected action as the common reference gives

$$\begin{aligned} J_{\text{adapt}} - J_{\text{now}} &= \big(J_{\text{adapt}} - J_{\text{delay}}\big) - \big(J_{\text{now}} - J_{\text{delay}}\big) \\ &\ge \delta_n(s) - c_n(s). \end{aligned} \tag{15}$$

All rewards and costs are already included in the three returns. The quantity $\delta_n(s)$ measures the advantage of adaptive selection over delayed fixed-action execution, while $c_n(s)$ bounds the advantage of immediate execution over that same reference. Thus, this decomposition does not charge any cost twice. Since $c_n(s)$ is an upper bound, the final relation is generally an inequality rather than an equality.

Proof of Theorem 3.1. The Bellman equation for the executable stage-termination action gives

$$\begin{aligned} Q_n^\star(s, a_n^{\text{end}}) &= r_e + \gamma \mathbb{E}_z\big[V_{n+1}^\star((s,z))\big] \\ &\ge r_e + \gamma \mathbb{E}_z\left[\max_{a\in B} Q_{n+1}^\star((s,z),a)\right] \\ &= J_{\text{adapt}}. \end{aligned} \tag{16}$$

The inequality follows because every action in B is feasible on the specified probability-one set of successor states, while $V_{n+1}^\star$ optimizes over all executable next-stage actions.

If $c_n(s) \le \delta_n(s)$, equation (15) implies

$$Q_n^\star(s, a_n^{\text{end}}) \ge J_{\text{adapt}} \ge J_{\text{now}} = \max_{a\in B} Q_n^\star(s,a).$$

Thus, ending the stage weakly dominates every currently executable persistent action. Since

$$\mathcal{A}_n = \mathcal{D}_n \,\dot\cup\, \mathcal{A}_{n+1} \qquad \text{and} \qquad a_n^{\text{end}} \in \mathcal{D}_n,$$

maximization over actions executable at s yields

$$\max_{a\in\mathcal{A}_n} Q_n^\star(s,a) = \max_{a\in\mathcal{D}_n} Q_n^\star(s,a).$$

An attained maximizer on the right is therefore an optimal decision at s, establishing the stated choice of $\pi_n^\star(s)$. □

Scope of the conclusion. The theorem is local to the specified state and stage. If its conditions hold at every state where the restriction is imposed, the finite-horizon Bellman argument permits an optimal policy satisfying these restrictions, under the usual measurable-selection assumptions. Other optimal policies may select persistent actions when values tie. If no persistent action is currently executable, the restriction is immediate and requires no information-gain comparison.

B Implementation Details for Structure-Aware Reinforcement Learning

This section explains the stage-local interface, action abstraction, and execution-time Search. It separates the semantic restriction from the policy encoding, amount executor, and planning approximation.

B.1 Semantic Interfaces and Policy Implementations

Stage-local decisions. The MMDP interface restricts the current choice to executable expiring actions at nonterminal stages and to the remaining executable actions at the final stage:

$$\begin{aligned}\pi_\theta(\{a \in \mathcal{D}_n : a \text{ is executable at } s\} \mid s, n) = 1, \quad n < N,\\ \pi_\theta(\{a \in \mathcal{A}_N : a \text{ is executable at } s\} \mid s, N) = 1.\end{aligned} \tag{17}$$

The stage-termination action $a_n^{\mathrm{end}} \in \mathcal{D}_n$ remains a choice alongside expiring domain actions. Selecting it advances through Γ and forfeits unresolved expiring opportunities. Flat instead permits all currently executable actions in $\mathcal{A}_n$, including persistent actions. Structural membership describes opportunity expiration; executability additionally depends on the current state.

Relation to the theorem. Theorem 3.1 supplies a sufficient condition for this restriction to preserve an optimal decision. The implementation uses the specified expiration structure without estimating $c_n(s)$ or $\delta_n(s)$ online. Those quantities involve optimal continuation values, not the learned critic or Search rollout scores. Where the deferral condition is not established, the restriction is evaluated as a structured approximation.

Encoding and comparison scope. The restriction can be implemented through a mask or a stage-dependent decoder and does not prescribe a policy-output representation. A continuous selector decoded against a local route list can assign the same output coordinate to different physical routes across interfaces; such comparisons are not fixed-global-logit mask ablations. Categorical implementations may also differ in output dimensions and termination aliases. Matching observations and hidden-layer widths therefore does not by itself equalize action distributions, parameter counts, or effective entropy regularization. These implementation properties are distinct from the semantic eligibility rule.

B.2 Action Abstraction and State-Dependent Execution

Action abstraction separates the choice of a semantic decision from its parameters. The policy selects an eligible route or termination control, and the executor $E_n(s, \tilde{a})$ produces the concrete action. Route eligibility is determined by the interface before execution; the executor does not make an excluded route eligible.

Forecast-based amount rule. For a synthetic cash route $\tilde{a} = (i, j)$, let b_i denote the source balance and f_i its currently visible outflow forecast. Define source surplus $u_i = (b_i - f_i)^+$ and recipient shortfall $g_j = (f_j - b_j)^+$. The executor protects the source's forecast requirement and allocates surplus to a recipient shortfall or investment:

$$\bar{q}_{ij} = \begin{cases} u_i, & j \text{ is the investment account},\\ \min\{u_i, g_j\}, & \text{otherwise}, \end{cases} \qquad q_{ij} = \min\{\bar{q}_{ij}, b_i^+\}. \tag{18}$$

AA supplies this amount directly. Basic route–amount policies additionally select a ratio $\rho \in [0, 1]$ multiplying the route-specific amount before balance clipping. Thus, AA removes the remaining learned ratio choice rather than introducing forecast-based amount semantics for the first time.

Concrete execution and approximation. Writing (i, j, q) for a transfer after numerical clipping, the executor takes the form

$$E_n(s, \tilde{a}) = \begin{cases} (i, j, q_{ij}), & \tilde{a} = (i, j),\\ a_n^{\mathrm{end}}, & \tilde{a} = \mathrm{STOP}. \end{cases} \tag{19}$$

For a given route and state, AA supplies one amount rather than optimizing over all feasible amounts. Its image can therefore omit an optimal concrete action even when the semantic deferral restriction is lossless. Comparisons between AA and route–amount policies evaluate their complete implementations, not an isolated executor effect.

B.3 Policy-Guided Search: Values and Execution

Search uses a frozen policy to guide candidate construction and continuation rollouts. It evaluates alternative current actions, executes only the selected first action, and replans after receiving a new observation. Policy parameters and the executor remain fixed; Search is not a policy-training update.

Rollout action values. For weighted hypothetical worlds with $w_k \geq 0$ and $\sum_k w_k = 1$, let $r_t^{(k)}$ be the reward at simulated transition t, with $t = 0$ denoting the candidate action, and T_k the number of transitions to termination. The evaluated synthetic controller uses

$$\widehat{Q}_\theta^{\mathrm{roll}}(s, n, \tilde{a}) = \sum_k w_k \sum_{t=0}^{T_k - 1} r_t^{(k)}. \tag{20}$$

AA candidates use the amount executor; non-AA candidates include an amount ratio. Subsequent actions follow the frozen policy. The score includes step rewards and terminal settlement without discounting or a learned terminal-value bootstrap. It is a planning-model estimate under policy continuation, not the optimal $Q^\star$ in the theorem.

Information and computation. Planning uses current observations and public model information, not the evaluation trajectory's unrevealed outcomes. Complete policy rollouts follow the rollout approach to sequential decision making (Bertsekas et al., 1997) and account for delayed consequences without fitting a tail-value estimator.

A per-decision transition cap bounds planning work, separately from the candidate-count limit. Candidates admitted to comparison must receive complete scores. Model and scenario approximations remain even with complete rollouts, so a larger cap need not improve realized return and a common cap need not imply equal realized computation. The interface itself does not require this particular score estimator or planner.

C Controlled Replenishment

C.1 Environment and Experimental Protocol

Ordering and inventory dynamics. Each episode contains W weekly ordering cycles followed by a final delivery week. In each cycle, the early decision occurs on Monday and the late decision on Thursday, after Monday–Wednesday demand. A regular supplier accepts orders only at the early decision, whereas a flexible supplier remains available until the late decision. Both suppliers deliver that cycle's orders together on the following Monday, before the next early observation. Inventory carries over between days and weeks. Daily demand is served from available inventory, and unmet demand is lost rather than backlogged. The final delivery week is fully settled without further ordering.

Let c_r and c_f denote regular and flexible unit procurement costs, v unit sales revenue, h daily holding cost, and b the unit unmet-demand penalty. For daily demand d_t and inventory x_t available before demand, sales and unmet demand are $s_t = \min(x_t, d_t)$ and $\ell_t = (d_t - x_t)^+$, and end-of-day inventory is $x_t^+ = (x_t - d_t)^+$. With total regular and flexible orders Q_r and Q_f, episode return is

$$G = \sum_{t=1}^{7(W+1)} \left(v s_t - h x_t^+ - b \ell_t \right) - c_r Q_r - c_f Q_f. \tag{21}$$

Procurement is charged when orders are placed, the objective is undiscounted, and terminal inventory has zero salvage value. Initial inventory I_0 is fixed independently of realized demand. Flexible orders share capacity K_f within each ordering cycle; the regular-order decoder has upper bound K_r.

Demand and forecast information. Weekly demand categories $Z_w \in \{1, 2, 3\}$ are independent and equiprobable, with associated totals $D_w \in \{\bar{D}/2, \bar{D}, 3\bar{D}/2\}$. Conditional on D_w, daily demand follows a multinomial allocation across the seven days with equal probabilities. A fine signal F_w identifies the category with probability p, and a coarse signal C_w is a symmetric degradation of F_w:

$$\begin{aligned} \Pr(F_w = f \mid Z_w = z) &= \begin{cases} p, & f = z, \\ (1-p)/2, & f \neq z, \end{cases} \\ \Pr(C_w = c \mid F_w = f) &= \begin{cases} q, & c = f, \\ (1-q)/2, & c \neq f. \end{cases} \end{aligned} \tag{22}$$

The signal model therefore factors as $Z_w \to F_w \to C_w$. To vary fine-signal accuracy while keeping coarse accuracy r fixed, we use

$$q = \frac{r - (1-p)/2}{p - (1-p)/2}. \tag{23}$$

This preserves the complete conditional distribution of the coarse signal given the demand category. Accuracy denotes correct classification of the weekly demand category, not a percentage error in a continuous forecast.

At the early decision in week w, both interfaces observe C_w and C_{w+1}. Updated additionally reveals F_{w+1} at the late decision, targeting the same delivery-week demand as C_{w+1} and retaining the coarse signal. The released fine signal remains visible when the delivery week becomes the current week. Retained withholds fine signals; intervening demand and inventory observations remain available in both conditions. Neither interface observes latent demand categories or future realized demand.

Decision interfaces and observations. Flat permits flexible orders at both decisions, whereas MMDP defers them to the late decision. Both use the same three-dimensional action box and continuous quantity decoder. For $a \in [-1, 1]^3$, define $u_j = \mathrm{clip}((a_j + 1)/2, 0, 1)$. The early regular order is $\bar{D}(u_0 + u_1)$, with $K_r = 2\bar{D}$; regular ordering is disabled at the late decision. Requested flexible quantity is $K_f u_2$, clipped to remaining weekly capacity, and is suppressed at the early decision under MMDP. Quantities are not rounded to integers. All four interface–information combinations use the common observation fields in Table 5, including explicit remaining flexible capacity. Unreleased fine-signal slots and visibility indicators are zero.

Postponing a fixed feasible flexible order preserves its price, delivery date, total capacity consumption, and subsequent physical availability. Since neither early nor late orders can serve demand before the next Monday, postponement changes only the timing of the procurement charge within an undiscounted objective. This establishes the benchmark's lossless local deferral property, not a finite-budget learning guarantee.

Environment parameters. Table 4 lists the numerical settings. Inventory and capacity are expressed relative to mean weekly demand, and monetary quantities relative to regular unit procurement cost. These are synthetic benchmark parameters shared across interfaces and information conditions, rather than a calibration to a particular business.

Matched training and policy selection. Each setting crosses Flat and MMDP with Retained and Updated, using N paired training seeds. All policies are trained from scratch with identical observation dimensions and actor/critic architectures. Within each seed, the four cells share initial weights and paired exogenous episode streams; every training reset generates a fresh episode. PPO samples Gaussian actions and clips them to the common action box. Training rewards are scaled by a constant κ, while all reported returns use original economic units. Table 6 lists the architecture, optimizer, and evaluation settings.

Checkpoints are evaluated at regular transition intervals on a shared validation bank. Each run selects the highest validation mean, with ties resolved in favor of the earliest checkpoint. Initialization evaluations and the checkpoint after completion of the final rollout batch are excluded from the selection candidates. Selected policies are evaluated deterministically on a shared test bank after all selections in that setting have been fixed. The test instances are excluded from training and per-run checkpoint selection.

Table 4: Replenishment environment parameters. The main-text setting uses $p = 0.80$; the corresponding degradation probability q follows Eq. (23).

Parameter	Symbol	Value
Ordering cycles	W	13 (14 demand weeks)
Mean weekly demand	$\bar{D}$	84 units
Weekly demand totals	D_w	$42, 84, 126$, equiprobable
Initial inventory	I_0	$0.75\bar{D} = 63$
Regular-order upper bound	K_r	$2\bar{D} = 168$
Shared flexible capacity	K_f	$0.8\bar{D} = 67.2$
Regular / flexible unit costs	c_r, c_f	$1.0, 1.5$
Unit sales revenue	v	6.0
Daily unit holding cost	h	0.5
Unit unmet-demand penalty	b	3.0
Coarse classification accuracy	r	0.57875
Fine classification accuracy	p	0.80 (main text)

Table 5: Common 24-dimensional observation. Indices are zero-based; signal categories use three-dimensional one-hot encodings.

Indices	Field	Encoding / scale
0–1	Early/late stage	One-hot
2	Remaining ordering cycles	$/W$
3	Current inventory	$/K_r$
4	Regular orders arriving next Monday	$/K_r$
5	Early flexible orders placed this week	$/K_f$
6	Remaining flexible capacity	$/K_f$
7	Inventory at this week's Monday opening	$/K_r$
8	Observed demand accumulated this week	$/(1.5\bar{D})$
9–11	Current-week coarse signal	One-hot
12–14	Current-week fine signal, if released	One-hot or zero
15	Current-week fine-signal visibility	$0/1$
16–18	Delivery-week coarse signal	One-hot
19–21	Delivery-week fine signal, if released	One-hot or zero
22	Delivery-week fine-signal visibility	$0/1$
23	Updated-condition indicator	$0/1$

Matched information conditions. Training, validation, and test generators use distinct split keys; training streams also depend on training seed and reset index. Demand, signals, and daily allocations use separate random-number streams. Revealing or withholding a fine signal does not change exogenous demand. Across fine-accuracy settings, demand and coarse-signal realizations remain fixed, and fine signals are sampled from $\Pr_p(F \mid Z, C)$ under the required joint model. Latent variables used for this coupling are never policy inputs.

Metrics and uncertainty. Each validation point averages the validation instances within a seed; learning curves show the across-seed mean and one sample standard deviation. For validation return $R_i(t_k)$ at checkpoint $k = 1, \ldots, K$, normalized validation AUC is

$$\mathrm{AUC}_i = \frac{1}{t_K - t_1} \sum_{k=1}^{K-1} \frac{R_i(t_k) + R_i(t_{k+1})}{2} (t_{k+1} - t_k). \tag{24}$$

It covers only the observed training interval and is expressed in return units. Test return first averages all test instances within a selected policy, yielding one value per training seed, rather than treating individual trajectories as independent training replicates.

Let $Y_{i,m,j}$ denote either seed-level metric for interface $m \in \{\mathrm{Flat}, \mathrm{MMDP}\}$ and information condition $j \in \{\mathrm{R}, \mathrm{U}\}$. In addition to interface differences and update gains, we calculate the paired interaction

$$I_i = \big(Y_{i,\mathrm{MMDP,U}} - Y_{i,\mathrm{MMDP,R}}\big) - \big(Y_{i,\mathrm{Flat,U}} - Y_{i,\mathrm{Flat,R}}\big). \tag{25}$$

Percentile-bootstrap intervals resample complete seed rows, preserving the pairing of all four cells. The reported 95% intervals quantify training-seed variability conditional on the evaluation bank; they are marginal intervals without adjustment for multiple comparisons or setting selection.

Table 6: Base PPO and evaluation settings, shared across the four interface–information combinations.

Parameter	Value
Paired training seeds N	10 (42–51)
Actor / critic hidden layers	$[128, 128]$, Tanh
Trainable parameters per model	39,943
Learning rate	3×10^{-4}
Rollout steps / minibatch size / epochs	256 / 64 / 10
Discount / GAE parameter	1.0 / 0.95
PPO clipping parameter	0.2
Entropy / value coefficients	0 / 0.5
Maximum global gradient norm	0.5
Training reward scale κ	0.01
Nominal / actual training transitions	100,000 / 100,096
Validation checkpoints	$5{,}000, 10{,}000, \ldots, 100{,}000$
Validation / test instances	20 / 200
Normalized AUC interval	$[5{,}000, 100{,}000]$
Bootstrap resamples / seed	100,000 / 9245000 (PCG64)
Implementation	Stable-Baselines3 2.7.1, PyTorch 2.6.0

C.2 Results and Interpretation

Learning and test performance. Tables 7 and 8 summarize the main-text setting. MMDP achieves higher mean validation AUC and test return under both information conditions. Forecast updating improves both policy families' mean test returns, with a larger improvement for MMDP in this setting: the test interaction is 68.4 with 95% CI $[3.0, 136.2]$. The AUC interaction, however, is not resolved. These are comparisons between separately trained and validation-selected policies, not responses of a single frozen policy to changed signals.

Forecast-accuracy sensitivity. Table 9 varies fine accuracy with the coarse-signal law, demand realizations, and base PPO configuration held fixed. Both interfaces' Updated mean test returns

Table 7: Mean and sample SD are calculated over ten training-seed means, each averaging 200 shared instances.

Interface	Information	Mean return	Seed SD
Flat	Retained	2095.4	64.1
Flat	Updated	2381.5	147.6
MMDP	Retained	2159.5	38.5
MMDP	Updated	2514.0	70.1

Table 8: Paired contrasts with marginal paired-bootstrap 95% CIs. Interaction is defined in Eq. (25).

Contrast	Validation AUC	Test return
MMDP − Flat, Retained	$+170.0$ $[113.1, 219.2]$	$+64.1$ $[20.2, 107.1]$
MMDP − Flat, Updated	$+180.3$ $[112.1, 247.8]$	$+132.5$ $[56.6, 213.0]$
Updated − Retained, Flat	$+244.0$ $[184.1, 301.1]$	$+286.1$ $[218.3, 347.7]$
Updated − Retained, MMDP	$+254.4$ $[221.1, 281.7]$	$+354.5$ $[298.5, 409.8]$
Interaction	$+10.3$ $[-34.8, 60.4]$	$+68.4$ $[3.0, 136.2]$

increase with fine accuracy; their relative advantage and interaction are not uniform across accuracy levels. The test interaction's interval excludes zero only at the displayed 80% setting, whereas MMDP has a higher Updated mean return than Flat at all three displayed accuracy levels.

Table 9: Test sensitivity to fine-signal accuracy under base PPO. Flat and MMDP columns are Updated mean returns; interaction includes its marginal 95% CI. Retained means are unchanged at 2095.4 (Flat) and 2159.5 (MMDP).

Fine accuracy	Flat	MMDP	Test interaction [95% CI]
75%	2260.9	2375.2	$+50.2$ $[-35.7, 154.5]$
80%	2381.5	2514.0	$+68.4$ $[3.0, 136.2]$
85%	2502.6	2625.2	$+58.6$ $[-39.4, 164.0]$

Interpretation. The results support a finite-budget learning benefit from the evaluated decision interface and show that delivery-week forecast updates can improve learned decisions. A larger benefit of forecast updating under MMDP is not a general conclusion: the test interaction depends on fine-signal accuracy, while the main-setting AUC interaction includes zero. The physical lossless-deferral property and these empirical learning results address distinct questions.

D Synthetic Cash Management

The five-account comparisons examine route restrictions and equal-size controls; the ten-account comparisons examine action abstraction and policy-guided planning. We separate environment specifications, shared methods, benchmark-specific evaluation protocols, and planning-budget accounting.

D.1 Environment

Accounts, routes, and daily stages. Each synthetic environment contains a master account M, an investment account I, and a set $\mathcal{O}$ of operating accounts with uncertain end-of-day cash demand. The five- and ten-account settings have $|\mathcal{O}| = 3$ and 8, respectively. One episode represents one daily decision cycle with two stages. The full early-stage route catalogue contains both directions between M and I, both directions between M and each operating account, and every directed connection between distinct operating accounts. At the stage boundary, the master–operating routes expire. The master–investment routes and inter-operating routes remain available. The full route catalogue therefore contains 14 routes in the five-account setting and 74 in the ten-account setting; respectively, six and sixteen expire, leaving eight and fifty-eight persistent routes.

Flat exposes the complete early-stage route catalogue, whereas the MMDP interface exposes only its expiring routes at that stage. Both expose the same persistent routes after the boundary, with termination available in both stages. Selecting termination ends the current stage without making another transfer. Reaching the stage's operation limit also advances the environment. The limits are ten non-termination operations per stage in the five-account setting and sixteen in the ten-account setting. A zero-amount transfer still counts as an operation. These are decision-operation limits, not subdivisions of physical hours.

Demand and forecast generation. At reset, the generator samples the day's nonnegative base demand, a possible localized demand shock, and the forecasts revealed during the episode. Base demand for operating account i is $d_i^{\mathrm{base}} = \max\{0, \mu_i + \sigma_i \epsilon_i\}$, where $\epsilon_i \sim \mathcal{N}(0,1)$. The initial forecast adds independent zero-mean Gaussian noise with standard deviation $\nu \max\{d_i^{\mathrm{base}}, 1\}$ and is clipped at zero. With probability 0.9, a single operating account receives an additional nonnegative shock; otherwise the shock vector is zero. The master and investment accounts have no exogenous demand. Table 10 lists the configuration parameters, with operating-account vectors ordered by account index.

Stage zero reveals only the initial forecast; the realized shock and updated forecast remain hidden. At the boundary, the updated forecast and shock information are revealed simultaneously with expiration of the early-only routes. The evaluated configurations use zero second-stage reveal noise, so the updated forecast equals the total end-of-day demand. This is a controlled full-revelation endpoint, not a claim that operational forecasts are perfect. Pre-sampling the full episode makes paired evaluation reproducible but does not grant the controller access to unrevealed outcomes.

Observations and transfers. Within each matched comparison, Flat and MMDP receive the same stage indicator, within-stage progress, account balances, current demand forecast, forecast liquidity gaps and surpluses, and revealed-shock fields. The continuous PPO parameterization selects a local route and an amount ratio $\rho \in [0,1]$. Let b_i be the current balance and $\widehat{d}_i$ the current forecast. Define forecast surplus $u_i = (b_i - \widehat{d}_i)^+$ and forecast gap $g_j = (\widehat{d}_j - b_j)^+$, where $x^+ = \max\{x, 0\}$. The route–amount executor computes

$$q_{ij} = \begin{cases} \rho u_i, & j = I, \\ \rho \min\{u_i, g_j\}, & j \neq I, \end{cases} \qquad q_{ij} \leftarrow \min\{q_{ij}, b_i^+\}. \tag{26}$$

The transfer immediately subtracts q_{ij} from the source balance and adds it to the destination balance. The public environments do not introduce separate per-route capacity parameters: executable quantities are constrained by balances and the forecast-based amount rule. Edge-only action-abstraction variants use their separately specified quantity executor; they should not be interpreted as continuous route–amount policies with an identical action parameterization.

Settlement and reward. No demand is settled at the intermediate stage boundary. After the final stage, the total daily demand d_i is deducted once from each account. If b_i^{pre} is the balance immediately before settlement, write $\widetilde{b}_i = b_i^{\mathrm{pre}} - d_i$. The undiscounted episode return is

$$R = \sum_i y_i \widetilde{b}_i^+ - \sum_i p_i (-\widetilde{b}_i)^+ - f \sum_{t \in \mathcal{T}} q_t - c|\mathcal{T}| - c_0 \sum_{t \in \mathcal{T}} \mathbf{1}\{q_t \leq 10^{-8}\}, \tag{27}$$

where $\mathcal{T}$ contains non-termination operations, y_i is the positive-balance yield coefficient, p_i is the shortfall-penalty coefficient, and f is the proportional transfer fee. The per-operation cost is c, with an additional zero-transfer penalty c_0 in the ten-account environment. Termination has no operation fee, although final termination also returns the settlement reward. Transfer fees are charged in the reward and are not deducted a second time from account balances. Investment-shaping rewards are disabled. All monetary quantities and reward coefficients are synthetic benchmark units rather than calibrated production costs.

Coupling and comparison scope. Expiration is explicit, but a persistent transfer generally cannot be deferred without changing its consequences: an earlier transfer changes balances, the quantities executable on other routes, and the resources available before a cutoff. Cash therefore tests a structured approximate restriction rather than the exact-deferral conditions of the theorem. Comparisons within

Table 10: Configurations of the two synthetic Cash environments. Initial balances are sampled uniformly within the listed ranges. Gaussian demand and shock samples are clipped at zero. Account-wise yield and penalty vectors are ordered as master, investment, then operating accounts.

Parameter	Five accounts	Ten accounts
Initial master balance	$[140, 180]$	$[140, 180]$
Initial investment balance	$[20, 40]$	$[30, 60]$
Initial operating balances	$[25, 45]$, $[20, 40]$, $[20, 35]$	$[18, 35]$ for each account
Base-demand means	$(55, 40, 35)$	$(40, 38, 36, 34, 32, 30, 28, 26)$
Base-demand standard deviations	$(10, 8, 7)$	$(8, 8, 7, 7, 6.5, 6.5, 6, 6)$
Initial forecast noise ratio ν	0.20	0.18
Shock probability	0.90	0.90
Shock Gaussian mean / SD	$35/12$	$28/9$
Shock target probabilities	$(.40, .35, .25)$	$(.16, .15, .14, .13,$ $.12, .11, .10, .09)$
Second-stage reveal noise	0	0
Yield coefficients	$(.003, .010, .001, .001, .001)$	.003 master, .010 investment, .001 each operating
Shortfall coefficients	$(4, 2, 10, 8, 8)$	4 master, 2 investment, 9 each operating
Transfer fee f	0.001	0.001
Operation cost c	20	10
Additional zero-transfer cost c_0	0	2

each controller family hold observations, dynamics, reward, execution rules, and evaluation instances fixed while changing the route interface. The five- and ten-account settings also differ in demand, initial liquidity, costs, and training protocols, so their results are not a controlled estimate of the effect of account count alone.

D.2 Five-Account Methods and Training

This subsection specifies the five-account controller implementations, training settings, route-mask construction, and serial timing method.

D.2.1 Controllers and PPO Training

Matched interfaces and PPO training. Flat and MMDP use the same 28-dimensional observation, two continuous route–ratio outputs, immediate amount executor, and reward function. Flat exposes 14 routes at stage zero and eight at stage one; MMDP exposes the six expiring routes at stage zero and the same eight persistent routes at stage one. Termination is available under both interfaces. Actor and critic each have two 128-unit hidden layers with Tanh activations, giving 40837 policy parameters. PPO uses learning rate 3×10^{-4}, rollout length 256, batch size 64, ten optimization epochs, $\gamma = 0.99$, GAE parameter 0.95, clipping parameter 0.2, entropy coefficient zero, value coefficient 0.5, and gradient-norm cap 0.5. The nominal budget is 500000 transitions; completing the final rollout gives an endpoint of 500224 transitions. Every independent test uses the final policy, without validation-based checkpoint selection.

We use ten paired training seeds, 42–51, retaining all runs. The extension from five to ten seeds was specified after earlier results were known; final checkpoints and evaluation rules were fixed before the added test results were accessed.

Random policy. Random samples uniformly from the available routes plus termination and independently samples a transfer ratio uniformly on $[0, 1]$. Each evaluated environment uses ten paired action-sampling replicates per interface. A separate PCG64 generator is keyed by environment, replicate, stage, and within-stage step, without consuming environment randomness. The stage-zero termination probability is $1/15$ under Flat and $1/7$ under MMDP; both are $1/9$ at stage one. Conse-

quently, the Random comparison includes changes in route identity, action count, and termination probability, rather than isolating expiration semantics alone.

Rolling-horizon MILP controller. We evaluate the forecast-based rolling-horizon MILP controller described in the main text. At each decision, it plans transfers from current balances, visible forecasts, public model coefficients, and routes admitted by the interface. Stage-zero planning demand adds the public expected localized shock to the visible forecast; stage-one planning uses the currently revealed forecast. The controller does not access unrevealed cash flows, future forecasts, or the environment's random-generator state.

The MILP has continuous transfer amounts, binary route-use variables, and positive and negative terminal balances. Its objective combines terminal yields and shortfall penalties with proportional transfer fees and operation costs. Constraints impose balance conservation, route-amount limits, shared source-surplus and target-gap budgets, and the remaining operation budget. We solve it with HiGHS (Huangfu & Hall, 2018) through `scipy.optimize.milp` (SciPy 1.15.3), using presolve and a zero relative-gap target. The controller first selects the largest positive planned transfer, breaking ties lexicographically by route, and clips its amount using the executor's limits. If the selected transfer has no executable amount above the tolerance, it chooses STOP without scanning the remaining planned transfers; otherwise, it executes the clipped transfer and replans. Solver optimality concerns this deterministic surrogate rather than the stochastic sequential problem.

Serial timing protocol. Controller timing uses an Apple M4 Max, one CPU worker, batch size one, and single-thread settings for PyTorch, BLAS, and HiGHS. The timing subset contains 20 fixed instances sampled without replacement from the 200-instance five-account test bank. Models are loaded before timing, controller blocks are warmed up, and three rounds randomize block and instance order. For each controller–interface–replicate–instance combination, the median episode time across rounds is averaged equally over instances and controller replicates.

Controller time includes action selection and wrapper processing: policy inference for learned controllers, action sampling for Random, and public-state construction, formulation, and solving for MILP. It excludes model loading, reset, actual environment advancement, auditing, and file writing. These measurements describe serial controller computation, not parallel service latency or training cost; episode times also depend on the number of real decisions.

D.2.2 Route-Mask Construction and Training

Design and matched quantities. This mechanism follow-up was designed after the earlier five-account controller and ten-account mask results were known. All new route subsets, training seeds, test instances, and statistical rules were fixed before observing this experiment's results. We retain the PPO, observation, reward, and executor described above and vary only the ordered stage-zero route list. True denotes the six-route expiration-based MMDP mask. The design includes five uniformly sampled controls, U1–U5, and five direct-funding-coverage controls, C1–C5. The main text compares the funding-preserving controls; this appendix reports both groups and the original primary contrast. True and every alternative expose six routes plus STOP, with identical local selector bins, STOP decoding interval, and amount-ratio decoding. All settings expose the same eight persistent routes at stage one. The masks remain fixed throughout training and testing.

Direct-funding controls. Each C mask retains the three routes from the master account to the operating accounts and selects three additional routes from the other eleven. Excluding True leaves $\binom{11}{3} - 1 = 164$ candidates. Five distinct C masks are drawn using lexicographically ordered candidate lists and NumPy 2.0.1 PCG64 with seeds 982201–982205, rejecting duplicates only. No balance, forecast, cost score, trajectory, or return is used to choose masks, and topology diagnostics do not trigger redrawing.

The coverage-preserving controls retain direct funding rather than merely graph connectivity because the original gap-limited executor does not guarantee multi-hop forwarding. They prevent missing direct access to an operating account from being the sole explanation for the comparison, but do not preserve investment opportunities, reverse flow, balances, or attainable policy value.

Uniform controls. After fixing C1–C5, the U candidate pool contains all six-route subsets except True and these five C masks, leaving $\binom{14}{6} - 6 = 2997$ candidates. U1–U5 are sampled uniformly from this lexicographically ordered pool using PCG64 seeds 982101–982105, rejecting only previously selected U masks. These controls match route count but need not preserve direct funding coverage; topology diagnostics do not trigger redrawing.

Complete route sets. Let M, I, and O_1, O_2, O_3 denote the master, investment, and operating accounts. The fixed route catalogue, indexed from zero, is

$$\begin{aligned}(e_0, e_1, e_2, e_3) &= (M{\to}I, I{\to}M, M{\to}O_1, O_1{\to}M),\\ (e_4, e_5, e_6, e_7) &= (M{\to}O_2, O_2{\to}M, M{\to}O_3, O_3{\to}M),\\ (e_8, e_9, e_{10}) &= (O_1{\to}O_2, O_2{\to}O_1, O_1{\to}O_3),\\ (e_{11}, e_{12}, e_{13}) &= (O_3{\to}O_1, O_2{\to}O_3, O_3{\to}O_2).\end{aligned}$$

The common stage-one list is $(0, 1, 8, 9, 10, 11, 12, 13)$. Within each stage-zero subset, the original catalogue order determines the local action mapping. Table 12 reports all ten fixed control subsets and their final-policy results.

Mask-policy training. The Flat and True endpoint policies from the controller comparison are reused. For each alternative mask, we train ten policies with seeds 42–51, giving 100 mask policies and 120 final policies in total. All endpoints are at 500224 transitions; no validation selection, early stopping, warm-starting, or Search is used. Training streams follow the original seed rule and are paired by episode ordinal, not by the transition index of differently sized episodes.

D.3 Five-Account Evaluation and Complete Results

D.3.1 Common Evaluation Protocol and Controller Results

Independent-test evaluation. The independent-test bank contains 200 shared instances with environment seeds 980000–980199. Each newly constructed wrapper is explicitly reset before scoring, retaining reset ordinal two. Final PPO policies are evaluated deterministically, without validation-based checkpoint selection. All controllers receive matched exogenous realizations and report the native undiscounted sum of all step rewards in Eq. (27). PPO averages instances within policy before forming ten paired training-seed differences. Random averages ten action-sampling replicates within instance before forming paired instance differences; MILP is deterministic and evaluated once per instance. Mask averages the five C controls before contrasts are formed. The timing subset contains 20 instances rather than the full 200-instance return bank.

Complete controller comparison. Table 11 collects the returns, paired interface and mask contrasts, and serial controller times underlying Table 1. All prescribed training seeds and action-sampling replicates remain in their corresponding averages. MMDP improves PPO and Random mean returns, while MILP obtains the same transfers and returns under all seven route interfaces.

Table 11: Complete five-account controller comparisons on 200 shared test instances. Mask is the equal average of C1–C5. Parenthesized controller times use 20 shared instances on a single-thread CPU. Paired 95% CIs resample PPO training seeds or Random/MILP test instances; timing and return statistics have different sampling units.

	Interface comparison			Mask comparison	
Controller	Flat (ms/episode)	MMDP (ms/episode)	MMDP − Flat [95% CI]	Mask (ms/episode)	MMDP − Mask [95% CI]
PPO	−552.003 (0.581)	−434.278 (0.543)	+117.725 [85.237, 151.115]	−486.853 (0.610)	+52.575 [41.175, 64.955]
Random	−789.646 (0.190)	−720.130 (0.160)	+69.515 [64.246, 74.890]	−721.888 (0.160)	+1.758 [−1.107, 4.687]
MILP	−330.286 (2.255)	−330.286 (1.988)	0.000 [0.000, 0.000]	−330.286 (2.082)	0.000 [0.000, 0.000]

D.3.2 Validation Learning Dynamics

Validation curves and Random references. The learning-curve matrix evaluates the twenty Flat/MMDP runs at one hundred scheduled checkpoints, from 5000 through 500000 transitions in 5000-transition increments. Fixed-bank curve scoring is distinct from the continuing ten-episode evaluation callback used during training. Training seed i uses the fixed environment bank $B_i = \{i+1000, \ldots, i+1019\}$, with a newly constructed wrapper explicitly reset before each deterministic evaluation (reset ordinal two). The banks are matched within each Flat/MMDP pair and overlap across training seeds. The released offline scoring entry point evaluates saved scheduled checkpoints without selecting a policy; the final independent-test endpoint remains at 500224 transitions. The learning plots describe across-seed validation dynamics, not independent-test performance.

The Random validation references in Figure 4a use the same documented banks, environment configuration, wrappers, and reset convention as PPO. The ten banks contain 29 unique environment seeds, 1042–1070. Random uses ten policy-RNG replicates per interface and unique instance, giving $2 \times 29 \times 10 = 580$ evaluations. We first average replicates within each instance, then average the twenty instances within each bank, and finally average the ten bank means equally. This preserves the PPO curves' instance weights rather than uniformly averaging the 29-instance union. These overlapping bank means are not independent training repetitions. Historical aggregate PPO curves do not permit retrospective episode-level fingerprint verification; the Random references match their documented evaluation protocol. The descriptive validation means are -761.180 for Random-Flat and -700.457 for Random-MMDP.

D.3.3 Individual Route-Mask Results and Interpretation

Table 12: Complete route-mask inventory and test returns. U controls match route count; C controls also preserve direct funding coverage, counted out of three master-to-operating routes. Means average 200 instances within training seed and then ten seeds; SD is across the ten seed means. This is the common bank used in Table 1.

Setting	Stage-zero route indices	Funding coverage	Mean	Seed SD
Flat	$0, \ldots, 13$	$3/3$	-552.003	59.071
True	$2, 3, 4, 5, 6, 7$	$3/3$	-434.278	10.145
U1	$1, 2, 5, 6, 8, 12$	$2/3$	-611.724	54.034
U2	$6, 7, 9, 10, 12, 13$	$1/3$	-631.880	54.546
U3	$1, 3, 4, 7, 9, 10$	$1/3$	-617.332	0.000
U4	$1, 3, 4, 10, 11, 12$	$1/3$	-617.332	0.000
U5	$0, 3, 5, 9, 11, 13$	$0/3$	-623.047	9.202
C1	$0, 2, 4, 5, 6, 13$	$3/3$	-510.815	60.256
C2	$0, 2, 3, 4, 6, 7$	$3/3$	-521.022	38.892
C3	$2, 3, 4, 6, 11, 13$	$3/3$	-450.455	9.952
C4	$2, 4, 6, 8, 12, 13$	$3/3$	-461.660	17.019
C5	$0, 2, 4, 6, 9, 11$	$3/3$	-490.316	83.555

Test coverage. The complete results include all 120 PPO endpoint policies and 24000 test episodes from Flat, True, U1–U5, and C1–C5. The main-text Flat, True, and C1–C5 comparison is a 70-policy, 14000-episode subset of this study. All endpoints were fixed before the common test results were accessed.

Estimands and inference. Let $R_{i,m}$ be the mean test return for training seed i and interface or mask m. The PPO interface contrast is $R_{i,\mathrm{True}} - R_{i,\mathrm{Flat}}$, and the Mask contrast is $R_{i,\mathrm{True}} - \frac{1}{5}\sum_{k=1}^{5} R_{i,C_k}$. Mask averaging precedes inference across the ten paired training seeds. Both marginal 95% intervals resample complete paired training-seed rows; the Mask interval uses 100000 percentile-bootstrap resamples with PCG64 analysis seed 982301. Neither episodes nor individual mask policies are treated as independent training repetitions. The True-minus-C comparison remains a pre-specified secondary contrast; focusing the main text on C does not change that designation.

The original sole primary contrast is $R_{i,\mathrm{True}} - \frac{1}{10}\sum_{k=1}^{5}(R_{i,U_k} + R_{i,C_k})$. Its mean is 119.280 (paired-bootstrap 95% CI $[112.272, 126.801]$), with positive differences in all ten training-seed pairs. The interval uses 100000 complete-row percentile-bootstrap resamples with PCG64 seed 982301. A two-sided sign-flip test enumerating all 2^{10} assignments, including ties, gives $p = 0.001953125$, conditional on sign-exchangeability of the paired differences. The primary is tested separately; the original Holm family comprises the three secondary contrasts True-minus-Flat, True-minus-mean(U), and True-minus-mean(C).

Inference is conditional on the fixed masks and test bank; it does not establish uniform superiority over all possible route masks.

Interpretation and implementation scope. True has a higher mean return than the average of the funding-preserving controls. This supports the importance of retained route identities beyond their count for this PPO implementation. It does not separately identify expiration semantics from account roles or network topology. The continuous local selector maps the same output coordinate to different physical routes across masks, so this is not a fixed-global-logit mask ablation. Equal STOP decoding intervals also do not force equal learned termination probabilities. Flat has fourteen routes rather than six, and its contrast additionally changes selector binning.

The frozen generator assigns the master account zero forecast demand, and the gap-limited executor preserves nonnegative balances before settlement. Consequently, operating-to-master and investment-to-master selections have zero preterminal transfer amounts but remain legal actions incurring the non-STOP penalty. These routes were neither removed nor repaired. In particular, True's reverse routes must not be interpreted as effective cash-recycling paths. The C comparison controls direct funding coverage, not the attainable value of the remaining route set.

Integrity. All 120 policy runs and 24000 test episodes are retained. Shared latent instances, deterministic replay, and return reconstruction were verified.

Sensitivity to immediate termination. On the common test bank used in Table 1, Flat policies with training seeds 46–48 immediately terminate both stages on all 200 instances; no MMDP policy exhibits this behavior on every instance. All ten pairs remain in the main results. A post-hoc sensitivity analysis excluding these three matched pairs yields a mean MMDP-minus-Flat difference of 92.052 (paired-bootstrap 95% CI $[65.345, 127.692]$), positive in all seven remaining pairs. The interval uses 100000 percentile-bootstrap resamples of the seven paired training-seed rows with PCG64 seed 2026092201, conditional on the fixed test bank. This behavior-conditioned analysis is not a corrected primary estimate, but shows that the positive gap is not confined to the three always-terminating Flat policies. Among the funding-preserving controls, C1 and C2 at seed 45 and C5 at seed 47 also immediately terminate both stages on every instance; all remain in the reported mask averages.

D.3.4 Random Interface and Mask Comparisons

Alignment and replication. Table 1 uses the route-mask bank, environment seeds 980000–980199, for every column. We denote the MMDP mask by True and the five equal-size masks preserving all three master-to-operating funding routes by C1–C5. Mask is the equal average of C1–C5, not the best-performing mask. A separately locked Random follow-up evaluates Flat, True, and C1–C5 with ten action-sampling replicates per instance, yielding $7 \times 200 \times 10 = 14000$ episodes. It uses the Random rule described above together with the unchanged executor, ordered masks, and reset convention, without training or loading PPO policies. The ten replicates represent action-sampling randomness, not training seeds. True versus C matches route count and per-decision STOP probability, but not necessarily episode length or termination timing.

Inference for Random. Random returns are first averaged over replicates within each instance and then over the five C masks where applicable. Its two pre-specified contrasts, True minus Flat and True minus mean(C), use 100000 paired environment-bootstrap resamples and 100000-draw two-sided Monte Carlo sign-flip tests, with PCG64 seeds 2026091702 and 2026091703, respectively. The tests include ties and the $(b+1)/(B+1)$ correction, require sign symmetry under the null, and use Holm adjustment over these two contrasts only. The mean gaps are 69.515 with marginal 95%

CI $[64.246, 74.890]$ and 1.758 with CI $[-1.107, 4.687]$; Holm-adjusted p-values are 0.000020 and 0.239348. Positive instance counts are 196/200 and 101/200. PPO intervals instead resample paired training-seed rows; they are not replaced by environment-level inference.

Provenance and scope. The Random follow-up was designed after prior results were known and reuses the PPO test bank; it is not a new-bank replication. All planned episodes are retained. The inconclusive Random mask contrast does not establish equivalence or a statistically resolved difference from PPO's interface response. Neither comparison isolates expiration semantics from route roles, topology, and execution.

D.3.5 Forecast-Based Optimization Reference and Runtime

Five-account comparison. Flat admits every currently available route; MMDP admits the stage-local routes; Mask averages the five fixed funding-preserving masks C1–C5. The forecast processing, MILP, action sequencing, executor, and reward are unchanged. The MILP controller is evaluated once on each of the 200 common test instances, with environment seeds 980000–980199 and matched exogenous realizations. Difference intervals use 100000 paired percentile-bootstrap resamples of instances with PCG64 seed 2026092302, conditional on the fixed masks. These intervals quantify instance variation; PPO intervals instead resample training seeds.

The MILP controller obtains mean return -330.286 under Flat, MMDP, and every funding-preserving mask. Its semantic actions, executed amounts, and balance paths coincide across all seven route lists: at stage zero it executes only the three direct funding routes common to every list, and the stage-one lists are identical. Both MMDP-minus-Flat and MMDP-minus-Mask return differences are therefore zero, with intervals rounding to $[0.000, 0.000]$. The executor clips stage-zero funding amounts because planning uses shock-adjusted demand while execution uses the visible forecast gap; this occurs identically across interfaces. This result describes the frozen controller on this test bank and does not imply that the interfaces are generally equivalent.

Five-account controller timing. PPO includes all ten training seeds, Random includes ten action-sampling replicates per instance, and MILP is evaluated once per instance per round, giving 8820 timed episodes under the shared serial protocol. Mask averages the five C masks equally. Table 1 reports mean time per episode alongside returns; timing uses the 20-instance subset, whereas return means use all 200 instances. Timed executions reproduce their corresponding archived returns, episode lengths, and instance fingerprints.

D.4 Ten-Account Methods and Training

The ten-account controllers use the same environment, observation fields, and reward within each interface comparison. PPO is evaluated with and without AA; Random and MILP provide non-learning references. Search is an execution-time procedure applied to frozen PPO policies rather than part of training.

Shared training protocol. We compare Flat, MMDP, Flat-AA, and MMDP-AA using ten matched training seeds, 42–51, with forty final policies in total. All receive the same 53-dimensional observation, including the stage indicator, and use separate policy and value networks with two 256-unit hidden layers and Tanh activations. PPO uses a constant learning rate of 3×10^{-4}, rollout length 512, minibatch size 128, ten optimization epochs, discount factor 0.99, GAE parameter 0.95, clipping parameter 0.2, entropy coefficient 0.001, value-loss coefficient 0.5, and maximum gradient norm 0.5. Advantages are normalized; observations and rewards are not normalized. Policies use orthogonal initialization and Adam with $\epsilon = 10^{-5}$ and coefficients $(0.9, 0.999)$, with one training environment per policy and CPU float32 execution.

The nominal training budget is two million transitions. Completing the final rollout and PPO update gives an actual endpoint of 2,000,384 transitions for every policy. Evaluation uses these final updated policies, not validation-selected checkpoints or the callback checkpoint at exactly 2,000,000 transitions. No Search or distillation is used in this training cohort.

Action representations and execution. Basic policies output a continuous route selector and amount ratio; AA uses categorical route slots and the forecast-based quantity in Eq. (18) without a learned ratio. A permitted route with zero executable amount remains a transfer action: it consumes a step, incurs the base environment's penalties, and follows the usual boundary rules rather than automatically selecting STOP. This `zero_passthrough` rule applies during both Raw execution and Search rollouts.

STOP aliases and observed behavior. Flat-AA has 75 categorical output slots: 74 route slots and one STOP slot at the early stage, and 58 route slots and 17 STOP aliases at the late stage. MMDP-AA has 59 output slots: 16 route slots and 43 STOP aliases at the early stage, and 58 route slots and one STOP slot at the late stage. These counts describe the encoding, not the learned action probabilities. The comparison therefore includes both route availability and categorical representation; matched hidden-layer widths and PPO settings do not equalize output dimensions, parameter counts, or effective entropy regularization. Cross-AA comparisons measure complete implementations rather than an isolated executor effect.

A post-hoc analysis of the saved Raw evaluation records finds no ten-account policy that immediately terminates both stages on every test instance. However, Flat-AA and MMDP-AA execute averages of 19.173 and 10.023 non-STOP actions per episode, including 14.718 and 4.651 zero-amount transfers, respectively. These quantities average within policy over the shared test instances and then equally over the ten training seeds. Zero-amount transfers remain non-STOP actions under the evaluated executor.

Optimization and timing methods. The rolling-horizon MILP uses current balances, visible forecasts, public model coefficients, and interface-admitted routes. Stage-zero demand includes the public expected localized shock; stage one uses the revealed forecast. Continuous amounts and binary route-use decisions minimize transfer costs and terminal shortfall costs net of yields, subject to balance conservation, amount limits, shared source and target budgets, and the remaining operation budget. HiGHS through `scipy.optimize.milp` (SciPy 1.15.3) solves the deterministic surrogate with presolve and a zero relative-gap target. The controller first selects the largest positive planned transfer, breaking ties lexicographically by route, and clips its amount using the executor's limits. If the selected transfer has no executable amount above the tolerance, it chooses STOP without scanning the remaining planned transfers; otherwise, it executes the clipped transfer and replans. It does not read unrevealed outcomes or environment randomness.

Serial timing uses an Apple M4 Max, one CPU worker, batch size one, and single-thread PyTorch, BLAS, and HiGHS. Models are preloaded, controller blocks warmed up, and three rounds randomize block and instance order on 20 fixed test instances. Repetition medians are averaged equally over instances and controller replicates. Timing includes policy inference, wrapper processing, Random sampling, or MILP construction and solving, as applicable. Search additionally includes public-state construction and hypothetical environment advancement. Loading, reset, actual environment advancement, auditing, and file writing are excluded; this measures serial computation rather than parallel service latency.

D.5 Ten-Account Evaluation, Search, and Planning Budgets

This subsection documents the ten-account learning comparison in Figure 4b, the Raw/Search evaluation in Table 3, and the Random, MILP, and runtime references. The learned-policy comparisons reuse forty final policies on 200 common test instances; Random and MILP use the same instance bank but not those learned policies.

D.5.1 Learning with Action Abstraction

Return-component accounting. The observed Raw MMDP-AA advantage includes both fewer penalized actions and a higher terminal yield-minus-shortfall outcome. This is trajectory accounting, not a causal decomposition: actions, balances, and terminal outcomes change jointly.

Validation learning curves. For training seed i, each validation point averages twenty complete episodes with environment seeds $i + 1000 + j$, $j = 0, \ldots, 19$. Banks remain fixed during training

and are shared across families, although banks for different training seeds may overlap. Wrapper construction performs the first reset and an explicit reset starts the scored second episode; all validation runs retain this reset convention. Validation returns use per-step float32 accumulation. The observed grid is every 5,000 transitions from 5,000 through 2,000,000, giving 400 points per policy and 16,000 points across the forty policies. Curves show means and sample standard deviations across ten training-seed means, without smoothing, best-so-far transformations, or an invented time-zero observation. The two Random references shown with the learning curves use the corresponding fixed validation banks and are not independent-test baselines.

The two-million-transition budget was chosen after inspecting earlier validation curves. The cohort therefore does not establish convergence or a blind comparison of training budgets.

Independent test and statistical units. All forty final policies are evaluated on 200 shared test instances, with environment seeds 994000–994199 and reset ordinal two, giving 8,000 episodes. This bank was held out from training and validation. Evaluation uses deterministic decoded policy actions and the native sum of all step rewards, not terminal reward alone. Returns are first averaged over the 200 instances within each policy, then contrasted within each of the ten matched training seeds. Uncertainty concerns training randomness conditional on this fixed test bank; the 2,000 episodes per family are not treated as 2,000 independent training repetitions.

The four primary raw-policy contrasts are listed in Table 14. Marginal percentile-bootstrap 95% intervals use 100,000 whole-training-seed resamples, shared across contrasts, with PCG64 seed 2026091409 and linear percentile interpolation. Exact two-sided paired sign-flip tests enumerate all $2^{10} = 1024$ sign assignments, including ties with absolute numerical tolerance 10^{-12}. Their inferential interpretation requires paired sign-exchangeability or symmetry. Holm adjustment covers these four raw-policy tests; the displayed confidence intervals are individually unadjusted.

Table 13: Final raw-policy results on 200 shared independent-test instances. SD is across ten training-seed mean returns. Output dimensions and parameter counts disclose implementation differences despite shared hidden-layer widths and PPO settings.

Family	Output dimension	Parameters	Mean return	SD
Flat	2	160005	-712.639	91.812
MMDP	2	160005	-567.546	148.361
Flat-AA	75	178764	-546.925	27.923
MMDP-AA	59	174652	-388.297	11.805

Table 14: Four primary raw-policy contrasts. Confidence intervals are marginal paired-bootstrap 95% intervals; Holm adjustment covers the four tests in this table. Positive pairs count training seeds with a positive difference.

Contrast	Mean gap	95% CI	Holm p	Positive pairs
MMDP − Flat	145.092	$[45.270, 228.854]$	.017578	9/10
MMDP-AA − Flat-AA	158.628	$[143.495, 173.972]$	.007813	10/10
Flat-AA − Flat	165.714	$[108.700, 217.625]$	.007813	10/10
MMDP-AA − MMDP	179.249	$[121.004, 276.249]$	.007813	10/10

AA improves both policy families in this cohort, and MMDP-AA also exceeds Flat-AA. All training seeds are retained, including the negative non-AA interface difference for seed 48.

D.5.2 Non-Learning Random Interface Reference

The Random controller uniformly samples one currently exposed semantic route or STOP, then independently samples a transfer ratio uniformly on $[0, 1]$. This post-result fixed-controller follow-up uses the same 200 independent-test instances (environment seeds 994000–994199, reset ordinal two) as the ten-account PPO and MILP comparisons. Ten action-sampling replicates per instance and interface yield 4000 complete episodes; these replicates are not training seeds. Within each instance–replicate pair, Flat and MMDP start from the same action-RNG seed, although their streams

may subsequently diverge when episode lengths differ. No policy is trained or selected for this comparison.

Returns sum all native step rewards. We first average the ten replicates within each instance and interface, then form 200 paired MMDP-minus-Flat instance differences. A 100000-resample percentile bootstrap over instances (PCG64 seed 2026092402) yields the primary 95% CI, conditional on the sampled action-RNG replicates. Random-Flat and Random-MMDP have mean returns of -983.908 and -855.267, respectively, giving a gap of $+128.641$ (95% CI $[121.287, 136.123]$).

The stage-zero STOP probability changes from $1/75$ under Flat to $1/17$ under MMDP, along with the exposed routes. The result therefore measures this complete interface response for a non-learning controller; it does not isolate expiration semantics. Its primary interval resamples test instances, whereas the PPO intervals resample training seeds, so their widths do not quantify the same source of variation.

D.5.3 Policy-Guided Search with Frozen Policies

Policies, instances, and planning caps. Search uses the same forty final policies and 200 instances as Raw, retaining its 8,000 episode returns. The three per-decision caps are $1\times$, $2\times$, and $4\times$ the base budget; $4\times$ is primary. Policy parameters remain frozen, with no fine-tuning, checkpoint selection, or distillation. The Search protocol was locked after earlier Raw and Search results were known but before the present Search outcomes were accessed; the bank is held out from training, not previously unseen by the investigators.

Public-information planning worlds. The planner constructs deterministic forecast-mean scenarios using only the current public observation and known environment parameters. At stage zero, it uses one no-shock world and eight target-shock worlds, with the public mean positive shock magnitude and occurrence/target probabilities. The visible forecast supplies a point estimate of base demand. At stage one, it uses one world whose terminal outflow equals the currently visible updated forecast. The controller does not copy the real episode's unrevealed shock, future forecast, terminal outflow, or random-generator state. Hidden trajectory fingerprints are used only by the auditor. This is a forecast-mean approximation, not exact posterior integration or a scenario-sampled stochastic optimizer.

Unified candidate construction. Within the non-AA pair and within the AA pair, candidate generation, pruning, ordering, scoring, and cap admission use the same program. Legal route sets and the frozen policy outputs still differ. Table 15 gives the common candidate upper bounds. These are separate from the per-decision planning-transition cap and do not require the planner to fill every candidate slot.

Table 15: Candidate rules shared within each interface pair. AA uses its existing amount executor; the continuous amount grid is not applied to AA.

Interface pair	Amount candidates	Candidate upper bound
Flat / MMDP	$0.25, 0.5, 0.75, 1.0$ and policy ratio	32
Flat-AA / MMDP-AA	Existing AA executor	16

Non-AA candidates are formed from legal routes and the amount-ratio grid, rounding ratios to six decimal places, clipping, and removing duplicates. If the full pool exceeds the limit, the original prefilter retains the policy route and its immediate local-index neighbors; a STOP proposal instead uses the first three routes. Consequently, the nominal cap of 32 commonly yields at most 16 proposals, including STOP. The neighborhood is not expanded to fill unused slots. AA candidates use probability-ranked legal routes, reserve a legal policy proposal, and include one semantic STOP. Original slot decoding and tie behavior are preserved; probabilities of STOP aliases are not merged before policy argmax. The runtime rejects invalid or missing categorical scores rather than silently changing to a fallback proposal rule.

Scoring and budget admission. For each candidate, the planner executes its first action and follows the deterministic frozen policy to termination in every hypothetical world. Scores combine the full

undiscounted sums of step rewards with the world weights, using the original wrapper and executor. The retained policy proposal is scheduled first and STOP second, unless STOP is itself the policy proposal. Before admitting a candidate, the planner checks whether the work already consumed plus the number of worlds times the maximum remaining episode length fits within the cap. Each admitted candidate must complete its entire world bundle; no partial score or terminal-value bootstrap is used. Actual simulated transitions are charged after execution. Exact score ties retain the original STOP-first semantic rule. Only the chosen first action is executed in the real environment, followed by replanning at the next observation.

Primary estimands and inference. The two primary contrasts, both at the $4\times$ budget, are H1: MMDP-AA Search minus Flat-AA Search, and H2: MMDP-AA Search minus MMDP-AA Raw. Episodes are averaged within policy before forming ten matched training-seed differences. Marginal paired-bootstrap 95% intervals use 100,000 whole-seed resamples, shared across all families, caps, and contrasts, with PCG64 seed 2026091501 and linear percentile interpolation. Exact two-sided sign-flip tests enumerate all 1024 assignments with inclusive ties and tolerance 10^{-12}; the same sign-exchangeability qualification as above applies. Holm adjustment covers only H1 and H2, not the separate four-test raw-policy family. No multiplicity adjustment is applied to the reported intervals, and no secondary p-values are added.

H1 is 80.193 with 95% CI $[71.324, 89.623]$; H2 is 94.611 with CI $[88.645, 100.735]$. Both are positive in all ten training-seed pairs, with exact $p = 0.001953125$ and Holm-adjusted $p = 0.00390625$. The uncertainty is conditional on the shared 200-instance bank.

Search results. Table 16 reports all four families' primary-budget returns and gains. Search improves all four families, with positive within-seed gains in every training-seed pair. MMDP-AA retains the highest mean Search return among these four learned policy families. The non-AA MMDP-minus-Flat Search gap is 150.108 (95% CI $[82.028, 214.718]$). The AA interface gap is 80.193, compared with 158.628 under Raw; the observed mean gap therefore narrows with Search.

Table 16: Complete ten-account Raw and Search results at the primary $4\times$ budget on 200 shared instances, using the same frozen policies. Raw and Search brackets are marginal bootstrap 95% CIs over ten training seeds; Search-gain brackets use paired within-seed differences. These are the primary-budget results underlying Table 3.

Policy	Raw	Search	Search gain
Flat	-712.64 [-765.65,-654.86]	-604.80 [-656.28,-547.31]	$+107.84$ [94.73,120.96]
MMDP	-567.55 [-665.37,-509.30]	-454.69 [-517.07,-414.68]	$+112.85$ [87.24,151.90]
Flat-AA	-546.93 [-564.09,-531.37]	-373.88 [-381.99,-365.32]	$+173.05$ [156.05,191.69]
MMDP-AA	-388.30 [-395.33,-381.45]	-293.69 [-298.21,-289.54]	$+94.61$ [88.65,100.74]

D.5.4 Optimization Reference and Serial Runtime

Optimization comparison. We apply the shared forecast-based rolling-horizon MILP to the ten-account environment. Within this environment, Flat and MMDP use the same forecast processing, MILP formulation, sequencing rule, executor, and reward. Both are evaluated on the 200 common test instances with environment seeds 994000–994199 and matched exogenous realizations. A 100000-resample paired percentile bootstrap over instances (PCG64 seed 2026092301) gives the interval for the interface difference.

Mean return is -216.312 for MILP-Flat and -206.290 for MILP-MMDP, a paired improvement of 10.022 (95% CI $[6.579, 13.853]$). The paired differences are positive, zero, and negative on 45, 143, and 12 instances, respectively. Both MILP means exceed those of the evaluated learned controllers on this bank, including their Search-assisted versions. This is a comparison with one deterministic rolling-horizon controller, not a bound on stochastic optimality.

Serial controller runtime. The frozen policies, MILP controller, and Search implementations are measured on 20 common test instances under the shared serial CPU protocol. Search uses the primary cap of 2304 simulated transitions per decision; its timing includes public-state construction,

hypothetical rollouts, and action selection. Episode times are averaged equally over instances and ten training seeds for learned controllers, or over instances alone for MILP. Per-decision values pool visited decisions after taking repetition medians.

Table 17: Ten-account serial CPU controller time in milliseconds on 20 shared test instances. Raw denotes direct policy execution; Search uses the primary planning cap. Values follow the aggregation described above. Lower is better.

	Per decision		Per episode	
Controller	Flat	MMDP	Flat	MMDP
PPO Raw	0.082	0.081	0.888	1.535
PPO-AA Raw	0.083	0.086	1.720	1.058
PPO + Search	96.620	109.764	740.795	1131.344
PPO-AA + Search	136.311	88.059	1965.039	1019.153
MILP	2.366	0.834	25.094	8.941

The MILP controller has higher mean test return than the learned controllers and higher measured controller time than direct policy execution; serial Search is more computationally expensive than MILP. Under MILP, the MMDP interface has both a higher mean return and a lower mean episode time than Flat. These measurements provide a common single-thread CPU reference, not an optimized deployment-latency comparison. Search can exploit parallel continuation-rollout evaluation and batched policy inference, with achievable acceleration depending on the simulator and hardware. Multicore and GPU configurations are not benchmarked here, so the observed serial runtime ordering should not be extrapolated to those configurations or to the separate production controller. End-to-end service latency and training cost are outside this timing protocol.

D.5.5 Planning-Budget Sensitivity and Computational Accounting

Budget definition. Let $B_0 = 576$ be the base cap on simulated transitions per real decision. The evaluated Search budgets are $B = kB_0$ for $k \in \{1, 2, 4\}$; Raw invokes no Search. A simulated transition is one environment step taken while evaluating a candidate continuation in a hypothetical world, not a training transition, a complete rollout, or a real transfer. At stage zero, the planner uses nine worlds and a conservative maximum remaining length of 32 transitions per world. Two complete worst-case candidate bundles therefore require at most $2 \times 9 \times 32 = 576$ transitions. Together with policy-first and STOP-second scheduling, the base cap permits both retained proposals to be scored completely. The larger caps double and quadruple this allowance; these are protocol settings, not theoretically optimal budgets.

Complete budget grid. Raw execution and all three Search caps retain the same policy cohort and test instances. The assembled comparison contains sixteen family-by-budget settings, 160 setting-by-training-seed endpoints, and 32,000 episode rows, including the 8,000 reused raw rows. Table 18 reports all mean returns. The figure uses equally spaced condition labels Raw, $1\times$, $2\times$, and $4\times$. Their positions are not a linear scale of realized computation; connecting lines only guide the eye between evaluated conditions. Shading is the sample SD across ten training-seed means, not episode-level SD or a confidence interval.

Table 18: Complete mean returns on the common 200-instance bank. Raw uses no planning; Search budgets are multiples of $B_0 = 576$ simulated transitions per real decision, with $4\times$ primary. Values average instances within policy and then the ten training-seed means.

Family	Raw	$1\times$	$2\times$	$4\times$
Flat	-712.639	-615.221	-607.033	-604.802
MMDP	-567.546	-466.195	-455.131	-454.694
Flat-AA	-546.925	-369.563	-371.722	-373.879
MMDP-AA	-388.297	-293.021	-293.701	-293.686

Both AA families obtain substantial Search gains at $1\times$, but their mean returns do not improve monotonically at larger budgets. The planner optimizes scores in approximate forecast-based worlds rather than the subsequently realized test return, so evaluating more candidates does not guarantee better realized performance. The present results do not identify a causal explanation for the small budget differences or establish equivalence, convergence, or a saturation threshold. The $4\times$ budget remains primary; the smaller budget is not promoted after observing its results. Non-AA mean returns improve across the evaluated budgets, also as a descriptive rather than universal pattern.

Realized planning work. Table 19 reports realized simulated planning transitions per episode. The caps apply to each real decision, whereas the table aggregates work over the full episode. Raw uses no Search, so its zero entries do not imply zero controller cost.

Table 19: Mean realized simulated planning transitions per episode on the shared test bank, rounded to the nearest transition. Parentheses in the column headings give the per-decision cap. Values average test instances within each policy and then the ten training-seed means.

Policy	Raw	$1\times$ (576)	$2\times$ (1152)	$4\times$ (2304)
Flat	0	1,983	4,384	6,764
MMDP	0	3,622	8,167	11,521
Flat-AA	0	2,743	10,937	20,937
MMDP-AA	0	3,066	7,460	9,829

Equal per-decision caps do not imply equal realized episode-level work. At the primary $4\times$ cap, MMDP-AA uses approximately 53% fewer simulated transitions per episode than Flat-AA while attaining a higher mean return (Table 18). This difference is not uniform across budgets: at $1\times$, MMDP-AA uses slightly more transitions than Flat-AA. These are descriptive comparisons of the complete policy–Search implementations, not a compute-matched test or an isolated effect of AA.

Design scope and integrity. The current matrix uses common final policies, unified candidate rules within each interface pair, and the same test bank. It supports comparisons among these complete implementations, not a single-module causal attribution. All prescribed policies, training seeds, and caps are retained; audits verified instance matching, full-return reconstruction, unchanged checkpoints, and complete-bundle accounting.

E Production-Scale Cash Management

The production case combines a production-derived simulator, historical backtesting, and human-reviewed deployment observations. We separate the environment and simulator protocol from model scope and the definitions of the economic and operational outcomes.

E.1 Environment and Simulator Evaluation Protocol

Setting and evidence sources. The production-derived simulator represents a 27-account cash-management network over 24 hourly decision stages per day. It is distinct from the public two-stage synthetic benchmarks. The operational network contains heterogeneous funding, operating, and interest-bearing accounts, with account-specific liquidity requirements and route-dependent transfer constraints. Account identities, institution-specific route tables, and proprietary operational data are not disclosed.

Hourly decision timeline. Each trajectory represents one business day. At the beginning of each hourly stage, realized external inflows and outflows are posted to account balances, forecasts are refreshed, and network availability is updated. Multiple transfers may be executed within a stage; an individual transfer is not itself an hourly stage. The controller may also end a stage without transferring funds. In the simulator, an End action advances to the next hour; operationally, this corresponds to waiting for the next decision window. External cash flows therefore affect liquidity throughout the day rather than being settled only at the terminal stage.

Information and action opportunities. The observation reflects current balances, realized cash flows, available forecasts, and current operational constraints. Each hourly boundary brings updated information and the permanent expiry of at least one domain-action opportunity. Some routes permit physical execution only later in the day, but the corresponding operations can be reserved or committed earlier. The semantic opportunity catalogue thus distinguishes earlier commitments for later execution from routes immediately executable at the current hour; balance changes determine current executable quantities. Temporary route closures handled through operator rejection are outside the modeled interface. Shared balances and feasibility constraints couple transfers, so the production comparison evaluates a structured restriction rather than establishing the theorem's exact-deferral conditions throughout the network.

Transfers and daily outcomes. Transfer decisions specify a route and an amount subject to the simulator's balance and route constraints. External cash flows are accounted for hourly, while the daily return is settled at the end of the day. Evaluation records liquidity shortfalls and overnight deposit income, with overnight deposit interest forming the principal yield component. These daily outcomes should be distinguished from the amount-based recommendation-adoption metric used after deployment.

Policy families and action abstraction. The simulator comparison evaluates Flat and MMDP interfaces with PPO-AA and GRPO-AA, both with and without Search-assisted execution. Under AA, the policy selects a transfer route and the executor determines the transfer amount from the current operational state and constraints. The operational executor and planner differ from their synthetic-benchmark counterparts. The comparisons concern the complete policy, interface, and execution implementations; they do not isolate a single optimizer or executor component.

Group-relative policy training. GRPO-AA adapts group-relative policy optimization (Shao et al., 2024) to simulated transfer trajectories. Each group contains trajectories sampled from the same initial condition. Their returns are used to construct group-relative advantages for updating the policy without a separately learned value baseline. The grouping therefore compares alternative trajectories under a common starting condition rather than directly comparing returns from unrelated days. Group-relative training is distinct from the additional planning performed by Search at execution time.

Search-assisted execution. Search holds policy parameters fixed and evaluates candidate transfer plans through continuation rollouts using the single cash-flow forecast available at the current decision. Rollout randomness comes from sampling subsequent policy actions, not from sampling multiple future cash-flow scenarios. Returns guide selection of the next action; after execution, the controller replans using newly observed balances, flows, and forecasts. It does not read the evaluation instance's unrevealed future cash flows. Realized flows are available to the simulator or retrospective evaluator for scoring, not to the controller in advance. This information boundary applies to simulator evaluation, historical backtesting, and deployment, without requiring an unchanged checkpoint throughout deployment.

Training and test instances. Operational source data are separated before synthetic training and test instances are generated with added stochastic variation. The resulting training and test sets have a 5:1 size ratio. All eight evaluated pipelines use the same 1,000 test instances, each representing a complete daily trajectory. The source-data separation precedes instance generation rather than splitting generated perturbations of the same source records across training and testing.

Evaluation and reporting. The production-simulator comparison uses five training seeds per method. Each policy's return averages the 1,000 common test instances, and Figure 5b reports the mean of the five policy-level averages. Only aggregate means are available for reporting; uncertainty intervals and significance claims are therefore not supplied. Search-assisted execution is evaluated separately from direct policy execution. Table 20 records the eight means underlying the figure.

Disclosure and interpretation. The simulator and underlying operational records are not publicly released because of confidentiality restrictions. The disclosed protocol therefore does not make these production results fully independently reproducible.

Table 20: Production-derived simulator mean test returns over five training seeds per method on 1,000 common instances, corresponding to Figure 5b. Raw denotes execution without Search. Values are descriptive means, without confidence intervals or significance claims. Higher return is better.

	Raw		Search	
Pipeline	Flat	MMDP	Flat	MMDP
PPO-AA	−3973.3	−3338.0	−1208.0	−852.8
GRPO-AA	−4718.5	−2265.3	−1459.1	−624.8

E.2 Model Scope and Evaluation Definitions

Model scope and evidence sources. The simulator means describe the five-seed evaluation models, not the separate approximately 100-million-parameter operational model. The historical backtest evaluates the estimated economic performance of the MMDP-GRPO-AA controller with Search under historical operational conditions. The subsequent deployment comparison measures human operators' adoption of recommendations from the operational model; it is not a direct deployment test of the simulator-evaluation checkpoints. These analyses have different models, outcomes, and replication units and do not constitute a single causal evaluation.

Computational scope and deployment. The MILP reference evaluated on the two-stage synthetic benchmarks is not a production-scale optimization baseline. The operational system instead requires repeated replanning as balances, forecasts, and route availability change across 24 hourly stages. Its Search-assisted MMDP-GRPO-AA controller evaluates continuation rollouts in parallel while keeping policy parameters fixed during execution. The production backtest compares this controller with the executed incumbent workflow, including human adjustments, rather than with the synthetic MILP reference.

E.2.1 Pre-Deployment Historical Backtest

Controller and baseline. The pre-deployment observation window spans six calendar months and includes 94 eligible business days, excluding weekends and applicable holidays, including extended holiday closures. Calendar month identities are withheld. The Search-assisted MMDP-GRPO-AA controller starts from the same daily opening balances as the historical workflow, retains the historical external inflows and withdrawals, and replaces internal transfer decisions. Its resulting balances are propagated hour by hour through daily settlement rather than reset to historical balances. The baseline is the actually executed workflow, combining incumbent optimization-model recommendations with human adjustments, not the optimization model alone. The controller sees external flows only as they occur; Search uses the contemporaneous forecast, not later realized flows. The backtest metric below is distinct from the simulator's numerical return.

Net-income metric and monthly summaries. For workflow k, define daily net income as $Y^k_{m,d} = I^k_{m,d} - C^k_{m,d}$, where I is overnight deposit interest and C comprises shortfall financing costs and penalties. Other income and cost components are omitted because their magnitudes are small relative to these two components in the evaluated operation. The daily relative improvement, expressed in percent, is

$$g_{m,d} = 100\,\frac{Y^{\mathrm{MMDP}}_{m,d} - Y^{\mathrm{historical}}_{m,d}}{Y^{\mathrm{historical}}_{m,d}}. \tag{28}$$

Historical net income is positive on the evaluated days, so no zero- or negative-denominator adjustment is needed. For month m, n_m is the evaluated-day count, $\widehat{\mu}_m$ averages the daily percentages equally, and e_m is the standard error of that mean, not the standard deviation of daily outcomes. Monthly means, standard errors, and intervals use the same dates. The reported monthly 95% intervals use $\widehat{\mu}_m \pm t_{0.975,n_m-1}e_m$; the supplied intervals are retained rather than recomputed from rounded inputs. This metric is neither a percentage reduction in liquidity shortfalls nor realized post-deployment revenue growth.

Table 21: Pre-deployment backtest summaries. Months are ordered chronologically. Improvements and confidence limits are in percent; standard errors are in percentage points. Monthly intervals are retained from the operational summaries; they are not recomputed from the rounded means and SEs. The full-period estimate and approximate interval are computed from the displayed summaries as described below.

Period	Days	Mean (%)	SE (pp)	95% CI (%)
Month 1	20	1.07	2.01	$[-3.12, 5.27]$
Month 2	13	7.97	3.21	$[0.97, 14.98]$
Month 3	14	9.69	3.01	$[3.17, 16.21]$
Month 4	8	9.99	3.06	$[2.74, 17.23]$
Month 5	18	10.27	9.99	$[-10.82, 31.37]$
Month 6	21	11.89	2.85	$[5.95, 17.84]$
Full period	94	8.25	2.17	$[3.80, 12.70]$

Aggregation. The full-period estimate weights monthly means by the number of evaluated days:

$$N = \sum_{m=1}^{6} n_m = 94, \qquad w_m = \frac{n_m}{N}, \qquad \widehat{\mu} = \sum_{m=1}^{6} w_m \widehat{\mu}_m = 8.246. \tag{29}$$

This is the mean daily relative net-income improvement across evaluated days. It is neither an equally weighted mean across months nor the relative improvement in total net income aggregated over the observation window.

Approximate uncertainty. Under the working assumption that the monthly mean estimates are independent, the standard error of the weighted estimate is

$$e = \left(\sum_{m=1}^{6} w_m^2 e_m^2 \right)^{1/2} = 2.171 \tag{30}$$

percentage points. Assigning $n_m - 1$ degrees of freedom to each monthly variance estimate gives the Welch–Satterthwaite approximation

$$\nu_{\text{eff}} = \frac{\left(\sum_{m=1}^{6} w_m^2 e_m^2 \right)^2}{\sum_{m=1}^{6} w_m^4 e_m^4 / (n_m - 1)} \approx 27.61. \tag{31}$$

The resulting approximate 95% interval is

$$\widehat{\mu} \pm t_{0.975, \nu_{\text{eff}}} e = [3.796, 12.696]\%. \tag{32}$$

This interval is reconstructed from rounded monthly summaries rather than a day-level bootstrap. It is not a training-seed confidence interval. It relies on the supplied standard errors and the stated independence and degrees-of-freedom assumptions, without an additional correction for temporal dependence or cross-month covariance. Monthly intervals are not adjusted for multiple comparisons; in particular, those for Months 1 and 5 include zero.

E.2.2 Human-Reviewed Deployment

Operational comparison. The incumbent workflow operated on the same 94 eligible days used for the backtest. Following the system-wide transition, operators reviewed recommendations from the MMDP-GRPO-AA controller with Search, accepting them or replacing them with manually determined transfers. Post-deployment summaries cover 86 eligible days across three complete months and a fourth month observed through the manuscript's reporting cutoff; calendar month identities are withheld. The accounting definition and human-review procedure are unchanged between periods. No account-level treatment/control split was used because decisions couple liquidity across the network; the comparison is observational rather than randomized.

Amount-based adoption. For day d, let A_d be the executed transfer amount attributable to accepted algorithmic recommendations, and let H_d be the manually determined executed amount, including revisions after rejected recommendations and independently added manual transfers. The daily adoption rate is

$$a_d = \frac{A_d}{A_d + H_d}, \qquad A_d + H_d > 0. \tag{33}$$

Here H_d uses the final executed manual amount, not the rejected recommendation's amount. Each transfer is counted once, not separately at its sending and receiving accounts. Canceled recommendations and approved but unexecuted transfers contribute no executed amount. Recommendations are accepted or rejected; partial acceptance is not used in this workflow. All observed days have positive total transfer amount. The denominator therefore covers actual algorithm-adopted and manually determined transfers, not the total originally recommended amount; adoption is not an acceptance rate by recommendation count.

Table 22: Mean daily amount-based adoption before and after deployment. Pre-deployment observations cover the same eligible days as the backtest. Post-deployment Month 4 is observed only through the reporting cutoff. Calendar identities are withheld. Period means weight monthly means by eligible-day counts.

Workflow	Period	Days	Mean adoption (%)
Incumbent optimization			
+ human review	Month 1	20	19.43
	Month 2	13	12.29
	Month 3	14	15.35
	Month 4	8	20.75
	Month 5	18	16.46
	Month 6	21	48.71
	Full pre-deployment period	94	23.92
MMDP-GRPO-AA + Search			
+ human review	Month 1	23	73.07
	Month 2	26	74.39
	Month 3	26	72.77
	Month 4 (partial)	11	69.11
	Observed post-deployment period	86	72.87

Aggregation and interpretation. For each observation period, the reported adoption mean is

$$\overline{a} = \frac{\sum_m n_m \overline{a}_m}{\sum_m n_m}, \tag{34}$$

where $\overline{a}_m$ is the monthly mean daily rate. Aggregation uses the supplied unrounded monthly values. The pre- and post-deployment means are 23.919% and 72.873%, respectively, a descriptive difference of 48.954 percentage points. The difference is reported in percentage points, not as a relative percentage gain. These are averages of daily amount-based rates, not pooled amount shares $\sum_d A_d / \sum_d (A_d + H_d)$. The partial month contributes only its 11 observed eligible days and is not extrapolated to a full month.

Higher adoption indicates a larger share of executed transfer amount following accepted algorithmic recommendations, not realized financial benefit. Operating conditions and human behavior can also change across periods, so this observational comparison does not identify a causal deployment effect; the economic estimate comes from the historical backtest.