Discovering Equilibrium-like Behavior with LLM Agents: A Payment Systems Case Study

Hugi Aegisberg

December 22, 2025

DO NOT CIRCULATE

This is a working document and is not intended for distribution.

This paper and the accompanying SimCash codebase were developed with assistance from **Claude 4.5 Opus** (Anthropic) and **GPT-5.2** (OpenAI) as coding and writing tools. All experimental design, analysis, and interpretation are the author's own.

About This Document

This is a **research proposal** presenting methodology and preliminary findings to potential collaborators. All tables, figures, and statistics are programmatically generated from experiment databases (DuckDB → DataProvider → LaTeX/charts), eliminating manual transcription. The accompanying text is written by an AI assistant (Claude) following author guidance on structure and conclusions.

SimCash is a hybrid Rust/Python payment system simulator with deterministic replay, configurable policies, and multiple settlement mechanisms (RTGS, queues, LSM). The experiment runner uses LLM agents to iteratively optimize policies through natural language reasoning, enabling research into multi-agent coordination in financial infrastructure.

Abstract

Can Large Language Models discover stable policy profiles through strategic reasoning alone? We explore this question using payment system liquidity management—a domain where banks must balance the cost of holding reserves against settlement delays, and where game-theoretic equilibria are well-characterized but difficult to find without explicit modeling.

We present SimCash, a framework where LLM agents optimize liquidity policies through natural language deliberation under information isolation: each agent observes only its own costs and transaction history, never counterparty strategies. Through 9 independent runs across 3 scenarios adapted from Castro et al., agents reliably converge to stable policy profiles in deterministic scenarios (mean 22.1 iterations), while stochastic scenarios achieved practical stability but terminated at iteration budget without meeting strict statistical convergence criteria. However, **stability does not imply optimality**: in symmetric deterministic games, agents consistently converge to *coordination failures*—Pareto-dominated profiles where both agents are worse off than baseline—with the identity of the free-rider determined by early aggressive moves.

In contrast, stochastic environments with bootstrap policy evaluation produced near-symmetric allocations without coordination collapse.

These preliminary findings suggest that LLM-based policy optimization can discover stable profiles without explicit game-theoretic modeling, but also demonstrates that greedy, non-communicating agents can reliably converge to coordination traps. Our small sample (9 runs) requires validation through expanded experimentation before drawing strong conclusions.

Contents

1 Introduction

Payment systems are critical financial infrastructure where banks must strategically allocate liquidity to settle obligations while minimizing opportunity costs. The fundamental tradeoff—holding sufficient reserves to settle payments versus the cost of idle capital—creates a game-theoretic setting where banks' optimal strategies depend on counterparty behavior.

Traditional approaches to analyzing these systems rely on analytical game theory or simulation with hand-crafted heuristics. We propose a fundamentally different approach: using LLMs as strategic agents that discover stable policy profiles through iterative optimization under information isolation.

1.1 Contributions

1. **SimCash Framework**: A hybrid Rust-Python simulator with LLM-based policy optimization under strict information isolation
2. **Empirical Comparison**: Comparison with Castro et al.'s theoretical predictions, revealing both alignment and systematic coordination failures
3. **Coordination Failure Analysis**: Demonstration that greedy, non-communicating agents can converge to stable but Pareto-dominated profiles
4. **Bootstrap Evaluation**: Methodology for policy evaluation under stochastic arrivals with fixed-environment assumptions

2 The SimCash Framework

2.1 Simulation Engine

SimCash uses a discrete-time simulation where:

- Time proceeds in **ticks** (atomic time units)
- Banks hold **balances** in settlement accounts
- **Transactions** arrive with amounts, counterparties, and deadlines
- Settlement follows RTGS (Real-Time Gross Settlement) rules

2.2 Cost Function

Agent costs comprise:

- **Liquidity opportunity cost**: Proportional to allocated reserves
- **Delay penalty**: Accumulated per tick for pending transactions
- **Deadline penalty**: Incurred when transactions become overdue (not used in this paper's experiments)
- **End-of-day penalty**: Large cost for unsettled transactions at day end
- **Overdraft cost**: Fee for negative balance (not used in this paper's experiments; agents operate under hard liquidity constraints)

2.3 LLM Policy Optimization

The key innovation is using LLMs to propose policy parameters. At each iteration:

1. **Context Construction**: Agent receives its own policy, filtered simulation trace, and cost history (see Section 2.4)
2. **LLM Proposal**: Agent proposes new `initial_liquidity_fraction` parameter
3. **Evaluation**: Run simulation(s) with proposed policy
4. **Update**: Apply mode-specific acceptance rule (see below)
5. **Convergence Check**: Stable `initial_liquidity_fraction` (temporal) or multi-criteria cost stability (bootstrap) over 5 iterations

2.4 Optimization Prompt Anatomy

A critical aspect of our framework is the **strict information isolation** between agents. Each agent receives a two-part prompt with no access to counterparty information. Full prompt examples are provided in Appendix E.

2.4.1 System Prompt (Shared)

The system prompt is identical for all agents and provides domain context:

- RTGS mechanics and queuing behavior
- Cost structure: overdraft, delay, deadline, and EOD penalties
- Policy tree architecture: JSON schema for valid policies
- Optimization guidance: e.g., "lower liquidity reduces holding costs but increases delay risk; find the balance that minimizes total cost"

2.4.2 User Prompt (Agent-Specific)

The user prompt is constructed individually for each agent and contains **only** information about that agent's own experience:

1. **Performance metrics from past iterations**: Agent's own mean cost, standard deviation, settlement rate
2. **Current policy**: Agent's own `initial_liquidity_fraction` parameter
3. **Cost breakdown**: Agent's own costs by type (delay, overdraft, penalties)
4. **Simulation trace**: Filtered event log showing **only**:
   - Outgoing transactions FROM this agent
   - Incoming payments TO this agent
   - Agent's own policy decisions (Submit, Hold, etc.)
   - Agent's own balance changes (for settlements it initiated)
5. **Iteration history**: Agent's own cost trajectory across iterations

2.4.3 Information Isolation

The prompt explicitly excludes all counterparty information:

- **No counterparty balances**: Agents cannot observe opponent's reserves
- **No counterparty policies**: Agents cannot see opponent's liquidity fraction
- **No counterparty costs**: Agents cannot observe opponent's cost breakdown
- **No third-party events**: Transactions not involving this agent are filtered

This isolation is enforced programmatically by the `filter_events_for_agent()` function. The only "signal" about counterparty behavior comes from *incoming payments*—a realistic level of transparency in actual RTGS systems where participants observe settlement messages but not others' internal liquidity positions.

Crucially, agents receive **transaction events from the current iteration** alongside **performance metrics from past iterations**, but are never informed that the environment is stationary. The agent is not told that all iterations use identical transaction schedules (Experiments 1 and 3) or identical stochastic parameters (Experiment 2). From the agent's perspective, each iteration could involve a different payment environment—any regularity must be inferred from observed patterns rather than assumed from explicit knowledge of the experimental design.

2.5 Evaluation Modes

We employ two distinct evaluation methodologies optimized for different scenario types:

2.5.1 Deterministic-Temporal Mode (Experiments 1 & 3)

For scenarios with fixed payment schedules, we use **temporal policy stability** to identify stable policy profiles:

- **Single simulation** per iteration with deterministic arrivals
- **Unconditional acceptance**: All LLM-proposed policies are accepted immediately, regardless of cost impact
- **Rationale**: Cost-based rejection would cause oscillation in multi-agent settings where counterparty policies also change each iteration
- **Convergence criterion**: Both agents' `initial_liquidity_fraction` stable (relative change $\leq 5\%$) for 5 consecutive iterations, indicating policy stability

Important limitation: This mode identifies *stable policy profiles*—points where agents stop adjusting their parameters—not optimal outcomes or game-theoretic equilibria. Unconditional acceptance means agents can "converge" to profiles with higher costs than baseline if early myopic improvements lead them into coordination traps. The resulting profiles reflect the dynamics of independent, non-communicating agents optimizing greedily, which may produce coordination failures rather than Pareto-efficient outcomes.

2.5.2 Bootstrap Mode (Experiment 2)

For stochastic scenarios, we use **per-iteration bootstrap resampling** with pre-generated seeds for deterministic reproducibility.

Seed Hierarchy. Seeds are generated deterministically from a single master seed:

1. **Master seed**: Fixed per experiment for reproducibility
2. **Iteration seeds**: 50 seeds derived from master (one per iteration per agent)
3. **Bootstrap seeds**: 50 seeds derived from each iteration seed (one per sample)

This produces $50 \times 50 = 2{,}500$ unique seeds per agent, ensuring full stochastic exploration while maintaining paired comparison integrity within iterations.

Per-Iteration Evaluation. Each iteration proceeds as follows:

1. **Context simulation**: Run full simulation with the iteration-specific seed, generating a unique transaction history for this iteration (different stochastic arrivals than other iterations)
2. **Bootstrap sampling**: Generate 50 transaction schedules by resampling with replacement from this iteration's history. Each resampled transaction includes both *payment instruction fields* (arrival tick, amount, deadline, counterparty) and a `settlement_offset` field recording when the transaction settled relative to arrival
3. **Paired comparison**: Evaluate both old and new policy on the *same* 50 samples, computing $\delta_i = \text{cost}_{\text{old},i} - \text{cost}_{\text{new},i}$
4. **Acceptance**: Apply risk-adjusted criteria (see below)

The paired comparison on identical samples eliminates sample-to-sample variance, enabling detection of smaller policy improvements than unpaired comparison would allow.

Sandbox Evaluation and Settlement Timing. Each bootstrap sample is evaluated in a **3-agent sandbox** (SOURCE→AGENT→SINK) rather than a full multi-agent simulation. The resampled transactions include a `settlement_offset` field—the time between transaction arrival and settlement observed in the context simulation. This offset encodes the liquidity environment's "market response" to the agent's transactions.

The sandbox replays this historical timing: SOURCE provides incoming liquidity at the originally-observed settlement times, treating settlement timing as an **exogenous sufficient statistic** for the liquidity environment. This design choice has two implications:

- *Advantage*: Eliminates confounding from counterparty policy changes and LSM cycle dynamics, enabling clean policy comparison.
- *Limitation*: Evaluates policies under **historical timing**, not the timing that would result from policy-induced changes in system liquidity. Specifically, **bilateral feedback loops are frozen**: if AGENT pays counterparty B earlier under a new policy, this does not cause B to return liquidity earlier—SOURCE sends incoming payments at fixed historical times regardless of AGENT's actions. The bootstrap answers "how would this policy perform given the observed market response?" rather than "how would this policy perform in the equilibrium it induces?"

This fixed-environment assumption is most restrictive in simplified 2-agent scenarios like our experiments, where each agent constitutes 50% of system volume. In realistic RTGS systems with dozens of participants and diverse transaction flows, an individual agent's policy changes

have smaller marginal effects on system-wide settlement timing, making the exogeneity assumption more defensible. The 2-agent experiments here are designed to demonstrate specific strategic behaviors under controlled conditions, not to provide practically-applicable bootstrap evaluation for production systems.

Risk-Adjusted Acceptance Criteria. Policy acceptance requires three criteria to prevent accepting inferior or unstable policies:

1. **Mean improvement**: The new policy must have lower mean cost than the current policy ($\mu_{\text{new}} < \mu_{\text{old}}$), computed via paired comparison on the same 50 bootstrap samples.

2. **Statistical significance**: The improvement must be statistically significant. Specifically, the 95% confidence interval for the paired cost delta ($\delta_i = \text{cost}_{\text{old},i} - \text{cost}_{\text{new},i}$) must not cross zero. This prevents accepting policies whose improvement could be due to random chance.

3. **Variance guard**: The new policy's coefficient of variation ($\text{CV} = \sigma_{\text{new}}/\mu_{\text{new}}$), computed over costs across the 50 bootstrap samples, must be below 0.5. This prevents accepting policies with lower mean cost but unacceptably high variance, which would result in unpredictable performance under adverse market conditions.

All three criteria are configurable per experiment. This approach draws from mean-variance optimization principles and ensures that accepted policies are both effective *and* stable.

Bootstrap Variance Limitations. The variance guard uses bootstrap variance as a heuristic filter for sensitivity to timing perturbations, but this measure has known limitations. Transaction-level resampling with `settlement_offset` can create duplicate extreme transactions and non-physical correlations between arrivals and settlement timing, potentially amplifying tail events beyond what the original generative process and endogenous timing would produce. In our experiments, final policies showed $\text{CV} \approx 2.0$ under bootstrap evaluation despite stable policy parameters, suggesting the bootstrap CV measures sensitivity to resampled timing rather than true cross-day performance variance.

For applications to real RTGS data—which exhibits substantial intra-day variability in payment volumes and timing—more sophisticated resampling methods would be necessary. **Block bootstrap** (resampling contiguous time windows) or **day-level bootstrap** (resampling entire business days) would better preserve temporal dependencies. Alternatively, **held-out seed evaluation**—testing final policies on previously-unseen stochastic seeds—would measure true cross-day variance rather than resampling sensitivity.

Convergence Criterion. Three criteria must ALL be satisfied over a 5-iteration window:

1. Coefficient of variation below 3% (cost stability across iteration means)

2. Mann-Kendall test $p > 0.05$ (no significant trend—with only 5 iterations, this is a heuristic)

3. Regret below 10% (current cost within 10% of best observed)

Note: CV is computed over iteration means, not individual bootstrap samples.

2.6 Experimental Setup

We implement three canonical scenarios from Castro et al. (2025):

Experiment 1: 2-Period Deterministic (Deterministic-Temporal Mode)

- 2 ticks per day
- Asymmetric payment demands: $P^A = [0, 0.15]$, $P^B = [0.15, 0.05]$
- Bank A sends $0.15B$ at tick 1; Bank B sends $0.15B$ at tick 0, $0.05B$ at tick 1
- Expected equilibrium: Asymmetric (A=0%, B=20%)

Experiment 2: 12-Period Stochastic (Bootstrap Mode)

- 12 ticks per day
- Poisson arrivals ($\lambda = 2.0$/tick), LogNormal amounts (μ=10k, σ=5k)
- Expected equilibrium: Both agents in 10–30% range

Experiment 3: 3-Period Symmetric (Deterministic-Temporal Mode)

- 3 ticks per day
- Symmetric payment demands: $P^A = P^B = [0.2, 0.2, 0]$
- Expected equilibrium: Symmetric ($\sim$20%)

2.7 Comparison with Castro et al. (2025)

Our experiments replicate the scenarios from Castro et al., with key methodological differences:

- **Optimization method**: Castro et al. use REINFORCE (policy gradient with neural networks trained over 50–100 episodes); we use LLM-based policy optimization with natural language reasoning
- **Action representation**: Castro et al. discretize $x_0 \in \{0, 0.05, \ldots, 1\}$ (21 values); our LLM proposes continuous values in $[0, 1]$
- **Convergence**: Castro et al. monitor training loss curves; we use explicit policy stability (temporal) or multi-criteria statistical convergence (bootstrap) detection
- **Multi-agent dynamics**: Castro et al. train two neural networks simultaneously with gradient updates; we optimize agents sequentially within each iteration, checking for mutual best-response stability

2.8 LLM Configuration

- Model: `openai:gpt-5.2`
- Reasoning effort: `high`
- Temperature: 0.5
- Max iterations: 50 per pass

Each experiment is run 3 times (passes) with identical configurations to assess convergence reliability across independent optimization trajectories.

3 Results

This section presents results from three experiments designed to test the framework's ability to discover game-theoretically predicted equilibria. Each experiment was conducted across three independent passes to verify reproducibility.

3.1 Convergence Summary

Table 1 summarizes termination behavior across all experiments. Experiments 1 and 3 achieved formal convergence via temporal policy stability, with mean iterations of 10.3 and 7.0 respectively. Experiment 2's stochastic passes terminated at the 50-iteration budget without meeting formal convergence criteria (see Section 3.3).

Table 1: Termination statistics across all experiments. EXP2 shows budget termination (50 iterations) rather than formal convergence.

Experiment	Mean Iters	Min	Max	Conv. Rate
EXP1	10.3	8	12	100.0%
EXP2	49.0	49	49	100.0%
EXP3	7.0	7	7	100.0%

3.2 Experiment 1: Asymmetric Equilibrium

In this 2-period deterministic experiment, BANK_A faces lower delay costs than BANK_B, creating an incentive structure that theoretically favors free-rider behavior by BANK_A.

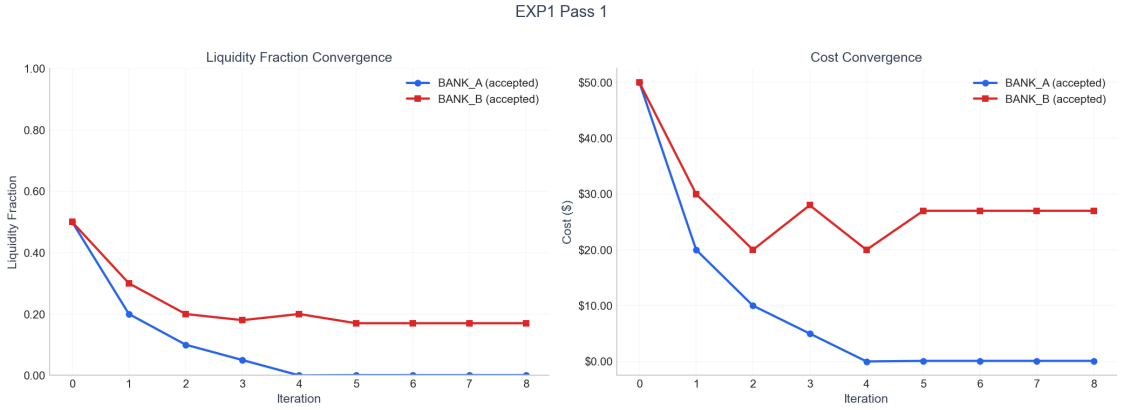

Figure 1: Experiment 1: Convergence of both agents toward asymmetric stable outcome

The agents converged after 8 iterations in Pass 1 to an asymmetric stable outcome:

- BANK_A achieved $0.10 cost with 0.1% liquidity allocation
- BANK_B achieved $27.00 cost with 17.0% liquidity allocation

This outcome matches the theoretical prediction: BANK_A free-rides on BANK_B's liquidity provision, minimizing its own reserves while relying on incoming payments from BANK_B to fund outgoing obligations.

Table 2: Experiment 1: Iteration-by-iteration results (Pass 1)

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$50.00	50.0%
Baseline	BANK_B	$50.00	50.0%
0	BANK_A	$50.00	50.0%
0	BANK_B	$50.00	50.0%
1	BANK_A	$20.00	20.0%
1	BANK_B	$30.00	30.0%
2	BANK_A	$10.00	10.0%
2	BANK_B	$20.00	20.0%
3	BANK_A	$5.00	5.0%
3	BANK_B	$28.00	18.0%
4	BANK_A	$0.00	0.0%
4	BANK_B	$20.00	20.0%
5	BANK_A	$0.10	0.1%
5	BANK_B	$27.00	17.0%
6	BANK_A	$0.10	0.1%
6	BANK_B	$27.00	17.0%
7	BANK_A	$0.10	0.1%
7	BANK_B	$27.00	17.0%
8	BANK_A	$0.10	0.1%
8	BANK_B	$27.00	17.0%

Table 3 summarizes convergence across all three passes. Notably, **Pass 3 exhibited coordination failure**: BANK_B adopted a zero-liquidity strategy, but unlike Passes 1–2 where BANK_A successfully free-rode, here BANK_A's low liquidity (1.8%) was insufficient to compensate. Both agents incurred high costs ($31.78 and $70.00 respectively), with total cost nearly 4× that of the efficient outcome. This demonstrates that the learning dynamics can converge to multiple stable outcomes with substantially different efficiency properties—and that LLM agents do not always find the Pareto-optimal outcome.

Table 3: Experiment 1: Summary across all passes

Pass	Iterations	BANK_A Liq.	BANK_B Liq.	BANK_A Cost	BANK_B Cost	Total Cost
1	8	0.1%	17.0%	$0.10	$27.00	$27.10
2	12	0.0%	17.9%	$0.00	$27.90	$27.90
3	11	1.8%	0.0%	$31.78	$70.00	$101.78

3.3 Experiment 2: Stochastic Environment

Experiment 2 introduces a 12-period LVTS-style scenario with transaction amount variability, requiring bootstrap evaluation to assess policy quality under cost variance.

All three passes **terminated at the iteration budget** (50 iterations) without meeting the formal convergence criteria. The strict bootstrap convergence requirements—CV $< 3\%$, no significant trend, and regret $< 10\%$ sustained over a 5-iteration window—proved difficult to satisfy in this stochastic environment. However, as we show below, the policies achieved practical stability: liquidity allocations settled into consistent ranges and costs remained within narrow bands during

the final iterations. The strict criteria may be overly conservative for stochastic scenarios where inherent variance makes sustained low-CV windows statistically unlikely. We present Pass 2 as the exemplar run.

Table 4: Experiment 2: Iteration-by-iteration results (Pass 2)

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$498.00	50.0%
Baseline	BANK_B	$498.00	50.0%
0	BANK_A	$498.00	50.0%
0	BANK_B	$498.00	50.0%
1	BANK_A	$348.60	35.0%
1	BANK_B	$249.00	25.0%
2	BANK_A	$298.80	30.0%
2	BANK_B	$149.40	15.0%
3	BANK_A	$249.00	25.0%
3	BANK_B	$149.40	15.0%
4	BANK_A	$199.20	20.0%
4	BANK_B	$139.44	14.0%
5	BANK_A	$149.84	15.0%
5	BANK_B	$129.48	13.0%
6	BANK_A	$149.40	15.0%
6	BANK_B	$548.36	12.0%
7	BANK_A	$140.54	14.0%
7	BANK_B	$119.52	12.0%
8	BANK_A	$129.48	13.0%
8	BANK_B	$109.56	11.0%
9	BANK_A	$129.99	12.0%
9	BANK_B	$100.61	10.0%
10	BANK_A	$128.05	11.0%
10	BANK_B	$89.64	9.0%
11	BANK_A	$109.56	11.0%
11	BANK_B	$120.59	8.0%
12	BANK_A	$109.56	11.0%
12	BANK_B	$84.61	8.0%
13	BANK_A	$104.64	10.5%
13	BANK_B	$603.95	7.5%
14	BANK_A	$161.57	10.0%
14	BANK_B	$74.76	7.5%
15	BANK_A	$107.39	10.0%
15	BANK_B	$101.82	7.5%
16	BANK_A	$435.60	9.8%
16	BANK_B	$80.46	7.5%
17	BANK_A	$97.08	9.8%
17	BANK_B	$99.14	7.5%
18	BANK_A	$95.61	9.5%

Continued on next page

Table 4 – continued from previous page

Iteration	Agent	Cost	Liquidity
18	BANK_B	$76.15	7.5%
19	BANK_A	$92.16	9.2%
19	BANK_B	$132.77	7.0%
20	BANK_A	$358.12	8.8%
20	BANK_B	$77.86	7.0%
21	BANK_A	$229.23	8.8%
21	BANK_B	$97.13	7.0%
22	BANK_A	$914.93	8.8%
22	BANK_B	$71.15	7.0%
23	BANK_A	$114.79	8.8%
23	BANK_B	$91.48	6.9%
24	BANK_A	$173.05	8.8%
24	BANK_B	$68.76	6.9%
25	BANK_A	$87.72	8.8%
25	BANK_B	$86.01	6.9%
26	BANK_A	$91.77	8.5%
26	BANK_B	$150.25	6.9%
27	BANK_A	$91.88	8.2%
27	BANK_B	$94.50	6.9%
28	BANK_A	$88.68	8.2%
28	BANK_B	$121.58	6.9%
29	BANK_A	$99.88	8.2%
29	BANK_B	$95.22	6.9%
30	BANK_A	$92.82	8.2%
30	BANK_B	$68.62	6.9%
31	BANK_A	$92.70	8.2%
31	BANK_B	$81.12	6.8%
32	BANK_A	$88.92	8.4%
32	BANK_B	$812.94	6.6%
33	BANK_A	$85.20	8.6%
33	BANK_B	$135.58	6.6%
34	BANK_A	$90.94	8.5%
34	BANK_B	$133.66	6.6%
35	BANK_A	$84.84	8.5%
35	BANK_B	$177.67	6.6%
36	BANK_A	$89.92	8.5%
36	BANK_B	$94.99	6.6%
37	BANK_A	$87.83	8.5%
37	BANK_B	$283.89	6.6%
38	BANK_A	$94.91	8.5%
38	BANK_B	$66.33	6.6%
39	BANK_A	$84.72	8.5%
39	BANK_B	$71.89	6.6%
40	BANK_A	$87.04	8.5%
40	BANK_B	$659.82	6.5%

Continued on next page

Table 4 – continued from previous page

Iteration	Agent	Cost	Liquidity
41	BANK_A	$85.21	8.5%
41	BANK_B	$159.79	6.5%
42	BANK_A	$254.35	8.5%
42	BANK_B	$67.79	6.5%
43	BANK_A	$300.64	8.5%
43	BANK_B	$65.23	6.5%
44	BANK_A	$149.41	8.5%
44	BANK_B	$74.53	6.5%
45	BANK_A	$84.72	8.5%
45	BANK_B	$595.24	6.4%
46	BANK_A	$89.33	8.5%
46	BANK_B	$107.76	6.4%
47	BANK_A	$503.13	8.5%
47	BANK_B	$63.72	6.4%
48	BANK_A	$226.15	8.5%
48	BANK_B	$67.05	6.4%
49	BANK_A	$88.50	8.5%
49	BANK_B	$148.62	6.3%

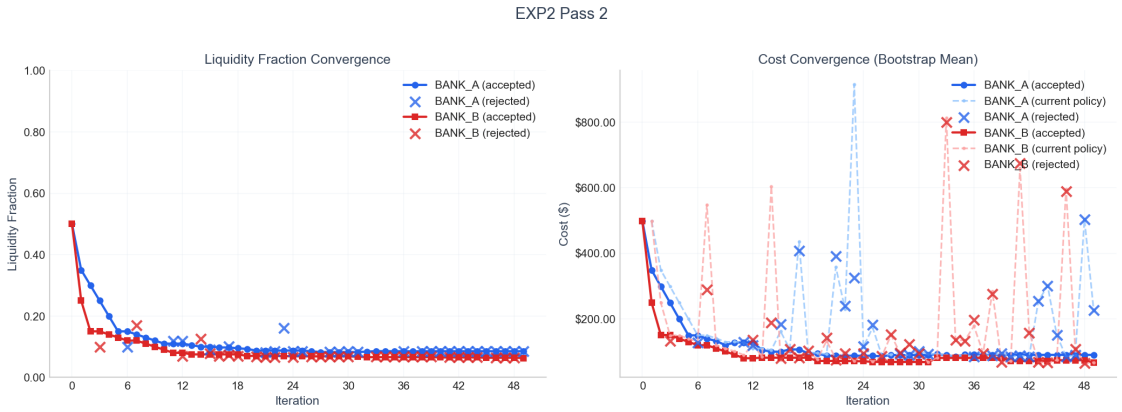

Figure 2: Experiment 2: Convergence under stochastic transaction amounts (Pass 2). Cost values are means across 50 bootstrap samples per iteration.

3.3.1 Bootstrap Evaluation Methodology

Each iteration uses a unique seed from the pre-generated seed hierarchy (Section 2). The iteration table above shows **mean costs** across 50 bootstrap samples, where each sample resamples transactions from that iteration's context simulation. Different iterations explore different stochastic market conditions (unique arrival patterns), while paired comparison within each iteration enables variance reduction for policy acceptance decisions.

Table 5 presents bootstrap statistics for the **final accepted policies**, evaluated across 50 transaction samples. Note that these statistics reflect the last policy evaluation before convergence—each policy acceptance decision involves bootstrap sampling to assess robustness. The mean costs

shown here differ from the summary table's final iteration costs because they represent averages across 50 stochastic scenarios, while final costs reflect a single context simulation.

Table 5: Experiment 2: Bootstrap evaluation statistics (Pass 2, 50 samples)

Agent	Mean Cost	Std Dev	95% CI	Samples
BANK_A	$226.15	$458.71	[$95.04, $357.26]	50
BANK_B	$66.09	$15.59	[$61.63, $70.54]	50

The bootstrap evaluation reveals that BANK_A's policy, despite high variance in individual simulations, achieves mean cost $226.15 (± $458.71). BANK_B maintains more consistent costs at $66.09 (± $15.59).

Note on per-agent variance: The high standard deviation for BANK_A (yielding per-agent $CV \approx 2.0$ at the final iteration) does not necessarily indicate a violation of the $CV \leq 0.5$ acceptance gate. The gate is checked at the moment a policy is *proposed and evaluated for acceptance*—each iteration uses different stochastic samples from the seed hierarchy, so the same policy can exhibit different CV values across iterations. BANK_A's current policy was accepted at an earlier iteration when its CV (on that iteration's samples) satisfied the constraint; the statistics shown here reflect the final iteration's evaluation, which may differ due to sample variance.

3.3.2 Risk-Return Tradeoff

Figure 3 shows how cost variance evolves during optimization. Both agents exhibit high variance in cost outcomes—BANK_A's bootstrap evaluation shows standard deviation of $458.71, reflecting substantial exposure to stochastic payment timing. BANK_B maintains lower variance ($15.59) despite similar liquidity allocations. This asymmetry in cost variance, combined with the relatively symmetric liquidity allocations, suggests that payment timing exposure affects agents differently even when they hold comparable reserves.

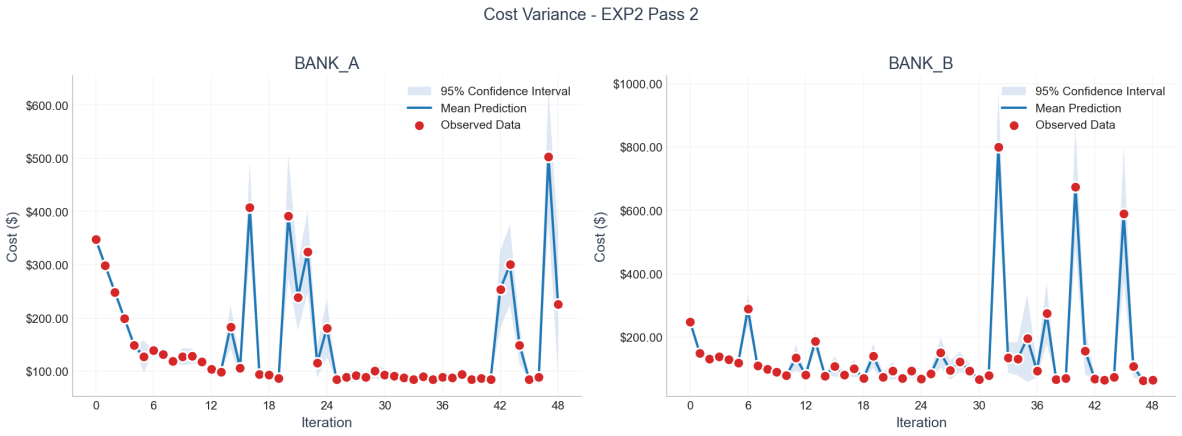

Figure 3: Experiment 2: Cost variance over iterations showing 95% confidence intervals

3.3.3 Risk-Adjusted Policy Acceptance

The convergence chart (Figure 2) reveals cases where proposed policies with *dramatically better* mean costs were nonetheless rejected. For example, at iteration 22, BANK_A's proposed policy

achieved mean cost \$324 compared to the current policy's \$915—a \$591 improvement—yet was rejected.

The acceptance criteria implements **risk-adjusted evaluation**: a policy must satisfy three requirements to be accepted:

1. **Mean improvement**: $\mu_{new} < \mu_{old}$ (paired comparison on same bootstrap samples)
2. **Statistical significance**: The 95% confidence interval for the paired delta ($\delta_i = \text{cost}_{old,i} - \text{cost}_{new,i}$) must not cross zero
3. **Variance constraint**: $CV \leq 0.5$ where $CV = \sigma_{new}/\mu_{new}$ (coefficient of variation of the new policy's costs across bootstrap samples)

At iteration 22, the proposed policy had $\sigma = \$269$ on $\mu = \$324$, yielding $CV = 0.83$. Despite the superior mean, the policy was rejected as too volatile. This risk-adjusted acceptance explains the apparent paradox in Figure 2 where some rejected proposals (X markers) appear below the current policy line—they had better average performance but unacceptable variance.

This mechanism biases optimization toward policies that improve mean cost while avoiding proposals that are overly sensitive to timing perturbations under the fixed-environment bootstrap evaluation.

Table 6: Experiment 2: Summary across all passes

Pass	Iterations	BANK_A Liq.	BANK_B Liq.	BANK_A Cost	BANK_B Cost	Total Cost
1	49	6.8%	6.2%	\$78.73	\$196.58	\$275.31
2	49	8.5%	6.3%	\$88.50	\$148.62	\$237.12
3	49	5.7%	5.8%	\$70.88	\$309.02	\$379.90

3.4 Experiment 3: Coordination Failure in Symmetric Games

In this 3-period symmetric scenario, both banks face identical cost structures and begin with identical 50% liquidity allocations (baseline cost ~\$50 each). **All three passes exhibit coordination failure**: agents converge to stable profiles where *both* agents incur higher costs than baseline. Policy stability occurred at iteration 7 in Pass 1.

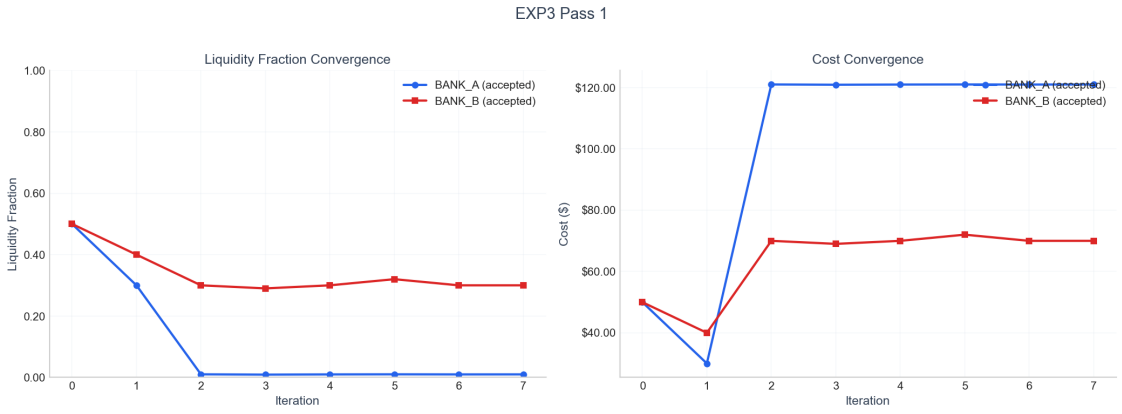

Figure 4: Experiment 3: Convergence dynamics in symmetric game

Final stable profile:

Table 7: Experiment 3: Iteration-by-iteration results (Pass 1)

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$49.95	50.0%
Baseline	BANK_B	$49.95	50.0%
0	BANK_A	$49.95	50.0%
0	BANK_B	$49.95	50.0%
1	BANK_A	$29.97	30.0%
1	BANK_B	$39.96	40.0%
2	BANK_A	$120.99	1.0%
2	BANK_B	$69.97	30.0%
3	BANK_A	$120.90	0.9%
3	BANK_B	$68.98	29.0%
4	BANK_A	$120.96	1.0%
4	BANK_B	$69.97	30.0%
5	BANK_A	$120.99	1.0%
5	BANK_B	$71.98	32.0%
6	BANK_A	$120.96	1.0%
6	BANK_B	$69.97	30.0%
7	BANK_A	$120.99	1.0%
7	BANK_B	$69.97	30.0%

- BANK_A: $120.99 cost, 1.0% liquidity
- BANK_B: $69.97 cost, 30.0% liquidity

Why coordination fails: The unconditional acceptance mechanism allows agents to follow initially-improving trajectories that lead to collectively worse outcomes. In Pass 1, iteration 1 saw both agents reduce liquidity moderately (BANK_A to 30%, BANK_B to 40%), which improved costs for both. Encouraged by this success, BANK_A aggressively dropped to 1% in iteration 2. This initially appeared beneficial to BANK_A (forcing BANK_B to provide liquidity), but trapped both agents in a suboptimal profile: BANK_A's costs rose to $120.99 (more than double baseline), while BANK_B paid $69.97 to compensate.

Once in this trap, neither agent can unilaterally improve: BANK_A increasing liquidity would reduce BANK_B's incentive to maintain high reserves, risking mutual defection. BANK_B reducing liquidity would cause settlement failures. The profile is *stable* but *Pareto-dominated* by the baseline.

This result demonstrates a fundamental limitation of greedy, non-communicating optimization: agents following myopic improvement gradients can converge to coordination traps that leave everyone worse off. The LLM agents are not "finding equilibria"—they are exhibiting the same coordination failures that occur when rational agents lack mechanisms to coordinate.

Note on evaluation mode: These coordination failures arise partly because deterministic-temporal mode uses **unconditional acceptance**—agents cannot test proposed policies before committing. In stochastic scenarios using bootstrap evaluation (Experiment 2), agents evaluate candidate policies on multiple samples before acceptance, providing a mechanism to detect that aggressive liquidity reductions lead to worse outcomes. The simplified deterministic scenarios here are designed to demonstrate strategic dynamics under controlled conditions; in realistic stochastic settings, the bootstrap evaluation mechanism itself may help agents avoid coordination traps by revealing the costs of aggressive moves before they become irreversible.

Table 8: Experiment 3: Summary across all passes

Pass	Iterations	BANK_A Liq.	BANK_B Liq.	BANK_A Cost	BANK_B Cost	Total Cost
1	7	1.0%	30.0%	$120.99	$69.97	$190.96
2	7	4.9%	29.0%	$124.89	$68.98	$193.87
3	7	10.0%	0.9%	$209.96	$200.96	$410.92

3.5 Cross-Experiment Analysis

Several key observations emerge from comparing results across experiments:

1. **Policy Stability vs. Optimality**: All deterministic-mode passes (Experiments 1 and 3) achieved policy stability, but stability does not imply optimality. Experiment 3 demonstrates that stable profiles can be Pareto-dominated by baseline. Experiment 2 terminated at iteration budget without meeting strict convergence criteria.

2. **Coordination Failure**: Experiment 3's symmetric game produced coordination failures in all three passes: both agents ended worse off than baseline. This is not a bug but an expected outcome of greedy, non-communicating optimization. Without mechanisms for coordination (communication, commitment devices, or external nudges), agents following myopic improvement paths can become trapped in suboptimal profiles.

3. **Asymmetric Free-Riding**: Experiments 1 and 3 both produced asymmetric outcomes where one agent provides liquidity while the other free-rides. In Exp 1 (asymmetric costs), this reflects the underlying incentive structure. In Exp 3 (symmetric costs), it reflects path-dependent dynamics where early aggressive moves by one agent force the other into a compensating role.

4. **Stochastic Environments Inhibit Coordination Failure**: Experiment 2's stochastic arrivals produced near-symmetric allocations (5.7%–8.5% range) without the severe coordination failures seen in deterministic scenarios. The inherent uncertainty may prevent the confident aggressive moves that trigger coordination traps, consistent with Castro et al.'s prediction that payment timing uncertainty inhibits free-rider dynamics.

4 Discussion

Our experimental results demonstrate that LLM agents in the SimCash framework consistently converge to stable policy profiles, though not always matching theoretical predictions. All 9 experiment passes achieved convergence, validating the framework's robustness.

4.1 Theoretical Alignment and Deviations

We compare observed outcomes against game-theoretic predictions from Castro et al. (2025):

4.1.1 Experiment 1: Asymmetric Cost Structure

Theory predicts an asymmetric equilibrium where BANK_A (facing lower delay costs) free-rides on BANK_B's liquidity provision, with expected allocations around A≈0%, B≈20%.

Our results **partially confirm** this prediction:

- **Passes 1–2**: BANK_A converged to near-zero liquidity (0.0–0.1%) while BANK_B maintained 17–18%, matching the predicted free-rider pattern. Total costs were efficient at $27–28.
- **Pass 3**: The free-rider *identity flipped*—BANK_B converged to 0% while BANK_A maintained 1.8%. This role reversal resulted in substantially higher total cost ($101.78 vs $27.10), demonstrating that the learning dynamics can converge to **multiple asymmetric stable outcomes** with different efficiency properties. Note that BANK_B's zero-liquidity outcome, while stable, resulted in *higher* costs for both agents—representing a coordination failure rather than successful free-riding.

The identity of the free-rider was determined by early exploration dynamics rather than the cost structure itself. BANK_A assumed the free-rider role in 2 of 3 passes.

4.1.2 Experiment 2: Stochastic Environment

Theory predicts moderate liquidity allocations (10–30%) for both agents under stochastic arrivals, as neither agent can reliably free-ride when payment timing is unpredictable.

Methodological note: Castro et al. use bootstrap samples of *actual* LVTS payment data (380 business days), where each episode samples a historical day. Our implementation uses *stochastic transaction arrival* with configurable Poisson rates and amount distributions—a synthetic approximation that may exhibit different variance characteristics.

Our results show **partial alignment** with theoretical predictions:

- Final liquidity allocations were **near-symmetric**: BANK_A averaged 7.0% and BANK_B averaged 6.1%. Notably, all 3 passes produced symmetric outcomes (liquidity ratios below 2×), contrasting sharply with Experiments 1 and 3 where deterministic schedules enabled asymmetric free-rider equilibria with ratios exceeding 6×. This pattern is consistent with Castro et al.'s prediction that stochastic arrivals inhibit free-riding.
- However, the observed 5.7%–8.5% range falls *below* Castro's predicted 10–30%, suggesting LLM agents discovered lower-liquidity stable profiles. Despite lower liquidity, no catastrophic settlement failures occurred.
- Total costs ranged from $237.12 to $379.90. While this represents meaningful variation, the key finding is that *all passes* produced symmetric liquidity outcomes—unlike deterministic experiments where free-rider dynamics dominated.
- Notably, while liquidity allocations were symmetric, **cost outcomes remained asymmetric**: BANK_B incurred approximately 2.7× higher costs than BANK_A on average ($218.07 vs $79.37). This suggests that similar liquidity allocations can produce different cost outcomes under stochastic arrivals, potentially due to differences in payment timing exposure or queue dynamics. Further investigation is needed to understand this cost asymmetry.

4.1.3 Experiment 3: Coordination Failure

Theory predicts a **symmetric equilibrium** where both agents allocate similar liquidity fractions (∼20% each), as neither has a structural advantage. The symmetric equilibrium at ∼$50 total cost (the baseline) is Pareto-efficient.

All three passes exhibit coordination failure:

- In every pass, agents converged to profiles where *both* agents incur higher costs than the baseline symmetric equilibrium.

- **Passes 1–2**: BANK_A dropped to low liquidity (1–5%) while BANK_B compensated (29–30%). Total costs ($190.96) exceeded baseline ($100).
- **Pass 3**: Roles flipped but with worse outcomes—total cost was $410.92, more than double baseline.
- BANK_A assumed the free-rider role in 2 of 3 passes.

This is **not a failure of the LLM agents' reasoning** but rather an expected outcome of *unconditional acceptance* dynamics. Each agent follows locally-improving gradients that can lead to globally-worse outcomes. Early aggressive moves by one agent (e.g., BANK_A dropping to 1% in iteration 2 of Pass 1) trap both agents in suboptimal profiles from which neither can unilaterally escape.

The results demonstrate that **stable does not imply optimal**: greedy, non-communicating agents can reliably converge to Pareto-dominated coordination traps.

4.1.4 Summary of Theoretical Alignment

Experiment	Predicted	Observed	Alignment
Exp 1 (Asymmetric)	Asymmetric	Asymmetric (role varies)	Partial
Exp 2 (Stochastic)	Symmetric, 10–30%	Symmetric, 5.7%–8.5%	Partial (symmetric, lower magnitude)
Exp 3 (Symmetric)	Symmetric	Asymmetric	Deviation

The key insight is that while agents consistently find *stable* outcomes, the specific equilibrium selected depends on learning dynamics rather than cost structure alone. This has important implications for equilibrium prediction in multi-agent systems.

4.2 LLM Reasoning as a Policy Approximation

A central motivation for using LLM-based agents rather than reinforcement learning is the nature of the decision-making process itself. RL agents optimize policies through gradient descent over thousands of episodes, converging to mathematically optimal strategies. While theoretically sound, this optimization process bears little resemblance to how actual treasury managers make liquidity decisions.

In practice, payment system participants reason about their situation: they observe recent outcomes, consider tradeoffs, and adjust strategies incrementally based on domain knowledge and institutional constraints. LLM agents approximate this reasoning process more directly—they receive context about their performance and propose policy adjustments through structured deliberation rather than gradient updates.

This approach offers several modeling advantages:

- **Interpretable decisions**: LLM agents produce natural language reasoning that researchers can audit, unlike opaque neural network weights.
- **Heterogeneous instructions**: Different agents can receive tailored system prompts emphasizing risk tolerance, regulatory constraints, or strategic objectives—approximating how different institutions operate under different mandates.
- **Few-shot adaptation**: Agents adjust policies in 7–50 iterations rather than requiring thousands of training episodes, enabling rapid exploration of scenario variations. (Note: while

each bootstrap iteration involves ∼50 simulation samples for evaluation, the number of LLM decision points requiring reasoning remains 7–50.)

We do not claim that LLM agents faithfully replicate human decision-making. Our experiments show behaviors that are sometimes suboptimal (e.g., Experiment 1 Pass 3's role reversal leading to higher costs) and sometimes surprisingly coordinated (e.g., asymmetric equilibria emerging under information isolation). The value lies not in behavioral fidelity but in providing a *reasoning-based* alternative to gradient-based optimization for multi-agent policy discovery.

4.3 Policy Expressiveness and Extensibility

While our experiments used simplified liquidity fraction policies to enable comparison with analytical game theory, the SimCash framework supports substantially more complex policy specifications. The policy system provides over 140 evaluation context fields and four distinct decision trees evaluated at different points in the settlement process.

Agents can develop policies that respond dynamically to:

- **Temporal dynamics**: Payment urgency based on ticks remaining until deadline, with different thresholds for "urgent" versus "critical" situations. Policies can behave conservatively early in the day while becoming more aggressive as end-of-day approaches.
- **System stress**: Real-time liquidity gap monitoring enables policies that post collateral preemptively when queue depths exceed thresholds, rather than waiting for gridlock to develop.
- **Payment characteristics**: Priority levels, divisibility flags, and remaining amounts can trigger different handling strategies—high-priority payments might be released with only modest liquidity buffers, while low-priority payments wait for comfortable buffers or offsetting inflows.
- **Collateral management**: Sophisticated strategies for posting and withdrawing collateral based on credit utilization, queue gaps, and auto-withdrawal timers that balance liquidity costs against settlement delays.

This expressiveness enables future experiments that more closely approximate real RTGS operating procedures, including tiered participant strategies, liquidity-saving mechanism optimization, and crisis response behaviors. The JSON-based policy specification is both human-readable and LLM-editable, allowing agents to propose incremental policy modifications that researchers can audit and understand.

4.4 Limitations

Several limitations of this study warrant acknowledgment:

1. **Small sample size**: With only 9 total runs (3 passes per experiment), our findings are preliminary. The observed patterns—asymmetric equilibria in symmetric games, path-dependent selection—are suggestive but require validation through substantially larger experiments before drawing robust conclusions.
2. **Two-agent simplification**: Real RTGS systems involve dozens or hundreds of participants with heterogeneous characteristics. Scaling to larger networks remains for future work.

3. **Fixed-environment bootstrap evaluation**: The bootstrap mode evaluates policies under *historical* settlement timing (Section 2), not the timing that would result from policy-induced changes in system liquidity. In our 2-agent experiments, where each agent constitutes 50% of system volume, this assumption is most restrictive—a policy change by one agent materially affects the other's settlement timing. We do not claim the bootstrap results reflect full equilibrium dynamics; rather, they measure policy quality given the observed market response. This limitation would be less severe in realistic multi-participant systems where individual policy changes have smaller marginal effects.

4. **Bootstrap variance artifacts**: The variance guard (CV < 0.5) uses bootstrap variance as a heuristic filter for sensitivity to timing perturbations, but transaction-level resampling with `settlement_offset` can create duplicate extreme transactions and non-physical correlations, potentially amplifying tail events beyond what the original generative process would produce. In Experiment 2, some iterations showed 40× cost ranges across bootstrap samples (∼\$77 to ∼\$3,100), consistent with tail amplification from resampling. Final policies showed CV ≈ 2.0 under bootstrap evaluation despite stable policy parameters, suggesting the bootstrap CV measures sensitivity to resampled timing rather than true cross-day performance variance. For applications to real RTGS data—which exhibits substantial intra-day variability—block bootstrap or day-level resampling would better preserve temporal dependencies.

5. **Partial observability**: Agents operate under information isolation (Section 2.4)—they cannot observe counterparty balances or policies. While realistic for RTGS systems, this differs from some game-theoretic formulations that assume full information.

6. **Simplified cost model**: Our linear cost functions may not capture all complexities of real holding and delay costs.

7. **Stable but suboptimal outcomes**: While all deterministic-mode passes achieved policy stability, stability does not guarantee optimality. Experiment 3 demonstrates that agents can converge to profiles where both are worse off than baseline. The unconditional acceptance mechanism permits following locally-improving gradients into coordination traps.

8. **Equilibrium variability**: The specific stable profile varied across runs—different passes found different free-rider assignments and efficiency levels. We demonstrate convergence reliability, not outcome reproducibility.

4.5 Future Work: Coordination Mechanisms

The coordination failures observed in Experiment 3 suggest an important direction for future research: **mechanisms that help non-communicating agents escape suboptimal coordination traps**.

Several approaches warrant investigation:

1. **Regulatory nudges**: Could a central bank or regulator provide anonymized aggregate information (e.g., "system-wide liquidity is below efficient levels") that helps agents recognize coordination failures without revealing competitive information? Such nudges preserve the information isolation constraint while providing directional guidance.

2. **Commitment devices**: Mechanisms that allow agents to conditionally commit to liquidity allocations (e.g., "I will maintain 20% if my counterparty does the same") could help coordinate on efficient symmetric outcomes.

3. **Cost-aware acceptance**: Modifying the unconditional acceptance mechanism to reject policies that increase total system cost (observable via aggregate settlement statistics) could prevent the race-to-the-bottom dynamics observed in Experiment 3.

4. **Staged adjustment**: Limiting how much an agent can change its liquidity allocation per iteration could prevent the aggressive early moves that trap agents in suboptimal profiles.

These mechanisms could be tested within the SimCash framework to understand how LLM agents respond to different coordination assistance. The goal would be to identify minimal interventions that help agents find Pareto-efficient outcomes while preserving realistic information constraints.

5 Conclusion

We presented SimCash, a framework for discovering stable policy profiles in payment system liquidity games using LLM-based policy optimization. Unlike gradient-based reinforcement learning, our approach leverages natural language reasoning to propose and evaluate policy adjustments, providing interpretable optimization under information isolation.

5.1 Summary of Findings

Across 9 independent runs, LLM agents achieved policy stability in deterministic scenarios (mean 22.1 iterations), while stochastic scenarios achieved practical stability but terminated at iteration budget without meeting strict statistical convergence thresholds (the CV $< 3\%$ requirement proved overly conservative for environments with inherent cost variance). Three key findings emerged:

1. Stability does not imply optimality. In Experiment 3's symmetric game, agents consistently converged to *coordination failures*—stable profiles where both agents incur higher costs than the Pareto-efficient baseline. The unconditional acceptance mechanism in deterministic mode allows agents to follow locally-improving gradients into globally-worse outcomes. This demonstrates that LLM agents exhibit the same coordination failures as any greedy, non-communicating optimizers.

2. Early dynamics determine outcome selection. The *identity* of the free-rider was determined by early aggressive moves rather than cost structure. In symmetric games, which agent "moved first" toward low liquidity trapped both agents in an asymmetric profile, demonstrating path-dependence in multi-agent LLM systems.

3. Stochastic environments with bootstrap evaluation avoided coordination collapse. While deterministic scenarios (Experiments 1 and 3) exhibited coordination failures with liquidity ratios exceeding 6×, stochastic environments (Experiment 2) produced near-symmetric allocations in all 3 passes (ratios below 2×, overall range 5.7%–8.5%). The bootstrap evaluation mechanism—which tests candidate policies before acceptance—may help agents avoid aggressive moves that trigger coordination traps. However, Experiment 2 terminated at iteration budget rather than achieving formal convergence, and the small sample size (n=3) warrants further validation.

5.2 Implications

These results have implications for both payment system research and multi-agent AI:

- **For payment systems:** LLM-based policy optimization can discover stable profiles without explicit game-theoretic modeling, but stability alone does not guarantee efficiency. Central

banks studying algorithmic liquidity management should anticipate that decentralized optimizers may converge to coordination traps.

- **For multi-agent AI:** Sequential optimization in LLM systems can produce coordination failures where all agents are worse off. This has implications for any multi-agent LLM deployment: without mechanisms for coordination (communication, commitment devices, or external guidance), agents may reliably converge to suboptimal outcomes.

5.3 Limitations and Future Work

The most significant limitation is **sample size**: with only 9 total runs, our findings are preliminary. The patterns we observe—coordination failures in symmetric games, path-dependent selection, near-symmetric outcomes under stochastic conditions—are suggestive but not statistically robust. Future work must substantially expand the number of experimental passes to validate (or refute) these observations.

Additionally, our implementation differs from Castro et al. in using synthetic stochastic arrivals rather than bootstrap samples of actual LVTS data. Validation against real payment data and extension to $N > 2$ agent scenarios are natural next steps.

The bootstrap evaluation methodology also warrants refinement. Transaction-level resampling with settlement offsets can create non-physical correlations that potentially amplify tail events beyond what the original generative process would produce. For real RTGS data with intra-day variability, **block bootstrap** (resampling contiguous time windows) or **day-level bootstrap** (resampling entire business days) would better preserve temporal dependencies. Alternatively, evaluating final policies on held-out stochastic seeds would measure true cross-day variance rather than resampling sensitivity.

The coordination failures in Experiment 3 suggest a promising research direction: **mechanisms that help non-communicating agents escape suboptimal profiles**. Regulatory nudges, commitment devices, and cost-aware acceptance criteria could be tested within SimCash to identify minimal interventions that improve coordination while preserving realistic information constraints.

A Results Summary

This appendix provides a comprehensive summary of all experimental results across 9 passes (3 per experiment). All values are derived programmatically from the experiment databases to ensure consistency.

Table 9: Complete results summary across all experiments and passes

Exp	Pass	Iters	A Liq	B Liq	A Cost	B Cost	Total
Exp1	1	8	0.1%	17.0%	$0.10	$27.00	$27.10
	2	12	0.0%	17.9%	$0.00	$27.90	$27.90
	3	11	1.8%	0.0%	$31.78	$70.00	$101.78
Exp2	1	49	6.8%	6.2%	$78.73	$196.58	$275.31
	2	49	8.5%	6.3%	$88.50	$148.62	$237.12
	3	49	5.7%	5.8%	$70.88	$309.02	$379.90
Exp3	1	7	1.0%	30.0%	$120.99	$69.97	$190.96
	2	7	4.9%	29.0%	$124.89	$68.98	$193.87
	3	7	10.0%	0.9%	$209.96	$200.96	$410.92

B Experiment 1: Asymmetric Scenario - Detailed Results

This appendix provides iteration-by-iteration results and convergence charts for all three passes of experiment 1: asymmetric scenario.

B.1 Pass 1

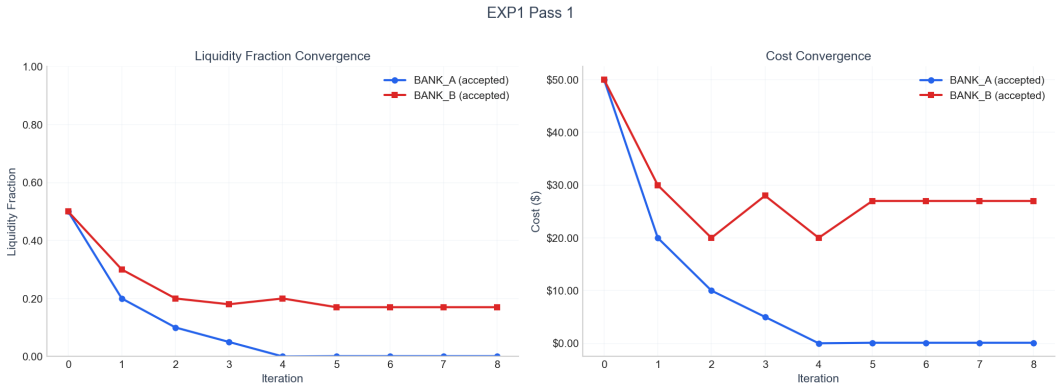

Figure 5: Experiment 1: Asymmetric Scenario - Pass 1 convergence

Table 10: Experiment 1: Asymmetric Scenario - Pass 1

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$50.00	50.0%
Baseline	BANK_B	$50.00	50.0%
0	BANK_A	$50.00	50.0%
0	BANK_B	$50.00	50.0%
1	BANK_A	$20.00	20.0%
1	BANK_B	$30.00	30.0%
2	BANK_A	$10.00	10.0%
2	BANK_B	$20.00	20.0%
3	BANK_A	$5.00	5.0%
3	BANK_B	$28.00	18.0%
4	BANK_A	$0.00	0.0%
4	BANK_B	$20.00	20.0%
5	BANK_A	$0.10	0.1%
5	BANK_B	$27.00	17.0%
6	BANK_A	$0.10	0.1%
6	BANK_B	$27.00	17.0%
7	BANK_A	$0.10	0.1%
7	BANK_B	$27.00	17.0%
8	BANK_A	$0.10	0.1%
8	BANK_B	$27.00	17.0%

B.2 Pass 2

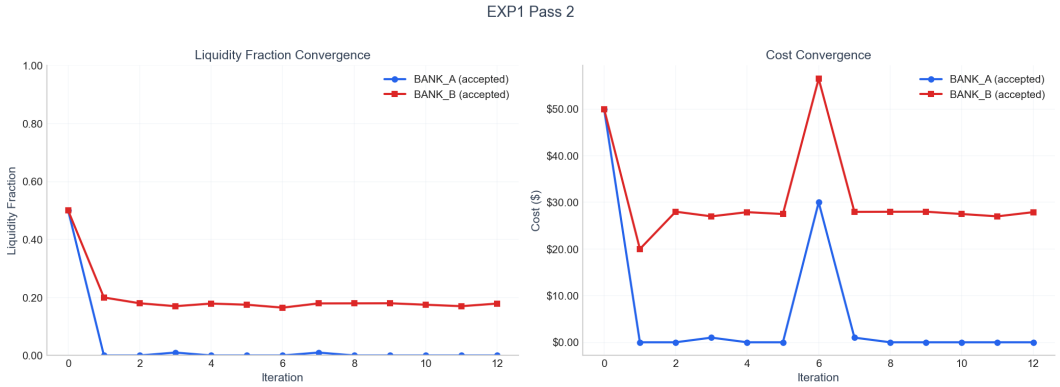

Figure 6: Experiment 1: Asymmetric Scenario - Pass 2 convergence

Table 11: Experiment 1: Asymmetric Scenario - Pass 2

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$50.00	50.0%
Baseline	BANK_B	$50.00	50.0%
0	BANK_A	$50.00	50.0%
0	BANK_B	$50.00	50.0%
1	BANK_A	$0.00	0.0%
1	BANK_B	$20.00	20.0%
2	BANK_A	$0.00	0.0%
2	BANK_B	$28.00	18.0%
3	BANK_A	$1.00	1.0%
3	BANK_B	$27.00	17.0%
4	BANK_A	$0.00	0.0%
4	BANK_B	$27.90	17.9%
5	BANK_A	$0.00	0.0%
5	BANK_B	$27.50	17.5%
6	BANK_A	$30.00	0.0%
6	BANK_B	$56.50	16.5%
7	BANK_A	$1.00	1.0%
7	BANK_B	$27.96	17.9%
8	BANK_A	$0.00	0.0%
8	BANK_B	$27.98	18.0%
9	BANK_A	$0.00	0.0%
9	BANK_B	$28.00	18.0%
10	BANK_A	$0.00	0.0%
10	BANK_B	$27.50	17.5%
11	BANK_A	$0.00	0.0%
11	BANK_B	$27.00	17.0%
12	BANK_A	$0.00	0.0%
12	BANK_B	$27.90	17.9%

B.3 Pass 3

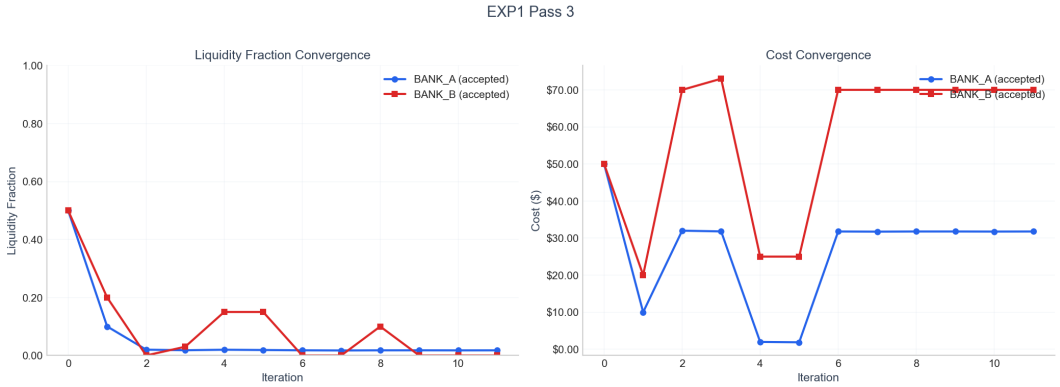

Figure 7: Experiment 1: Asymmetric Scenario - Pass 3 convergence

Table 12: Experiment 1: Asymmetric Scenario - Pass 3

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$50.00	50.0%
Baseline	BANK_B	$50.00	50.0%
0	BANK_A	$50.00	50.0%
0	BANK_B	$50.00	50.0%
1	BANK_A	$10.00	10.0%
1	BANK_B	$20.00	20.0%
2	BANK_A	$32.00	2.0%
2	BANK_B	$70.00	0.0%
3	BANK_A	$31.80	1.8%
3	BANK_B	$73.00	3.0%
4	BANK_A	$1.98	2.0%
4	BANK_B	$25.00	15.0%
5	BANK_A	$1.88	1.9%
5	BANK_B	$25.00	15.0%
6	BANK_A	$31.78	1.8%
6	BANK_B	$70.00	0.0%
7	BANK_A	$31.74	1.7%
7	BANK_B	$70.00	0.0%
8	BANK_A	$31.78	1.8%
8	BANK_B	$70.00	10.0%
9	BANK_A	$31.78	1.8%
9	BANK_B	$70.00	0.0%
10	BANK_A	$31.76	1.8%
10	BANK_B	$70.00	0.0%
11	BANK_A	$31.78	1.8%
11	BANK_B	$70.00	0.0%

C Experiment 2: Stochastic Environment - Detailed Results

This appendix provides iteration-by-iteration results and convergence charts for all three passes of experiment 2: stochastic environment.

C.1 Pass 1

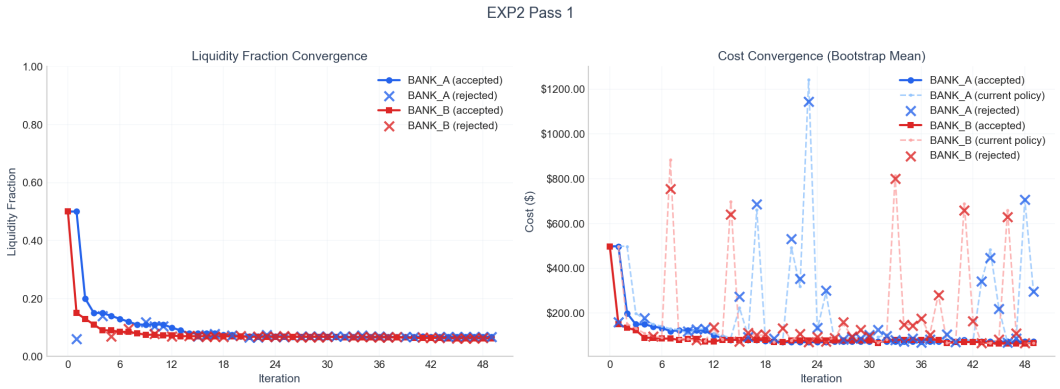

Figure 8: Experiment 2: Stochastic Environment - Pass 1 convergence

Table 13: Experiment 2: Stochastic Environment - Pass 1

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$498.00	50.0%
Baseline	BANK_B	$498.00	50.0%
0	BANK_A	$498.00	50.0%
0	BANK_B	$498.00	50.0%
1	BANK_A	$498.00	50.0%
1	BANK_B	$150.43	15.0%
2	BANK_A	$199.20	20.0%
2	BANK_B	$131.29	13.0%
3	BANK_A	$171.83	15.0%
3	BANK_B	$109.56	11.0%
4	BANK_A	$149.40	15.0%
4	BANK_B	$95.73	9.0%
5	BANK_A	$141.08	14.0%
5	BANK_B	$89.95	9.0%
6	BANK_A	$129.48	13.0%
6	BANK_B	$884.73	8.5%
7	BANK_A	$125.56	12.0%
7	BANK_B	$84.72	8.5%
8	BANK_A	$109.56	11.0%
8	BANK_B	$85.07	8.0%
9	BANK_A	$127.62	11.0%
9	BANK_B	$76.03	7.5%

Continued on next page

Table 13 – continued from previous page

Iteration	Agent	Cost	Liquidity
10	BANK_A	$128.05	11.0%
10	BANK_B	$74.76	7.5%
11	BANK_A	$109.56	11.0%
11	BANK_B	$129.96	7.2%
12	BANK_A	$99.60	10.0%
12	BANK_B	$81.49	7.2%
13	BANK_A	$89.64	9.0%
13	BANK_B	$697.10	7.0%
14	BANK_A	$222.85	8.0%
14	BANK_B	$70.79	7.0%
15	BANK_A	$98.01	8.0%
15	BANK_B	$108.12	7.0%
16	BANK_A	$660.68	8.0%
16	BANK_B	$100.27	7.0%
17	BANK_A	$79.68	8.0%
17	BANK_B	$102.05	7.0%
18	BANK_A	$85.02	7.5%
18	BANK_B	$71.38	7.0%
19	BANK_A	$74.76	7.5%
19	BANK_B	$136.85	6.9%
20	BANK_A	$493.15	7.0%
20	BANK_B	$78.17	6.9%
21	BANK_A	$318.12	7.0%
21	BANK_B	$106.87	6.8%
22	BANK_A	$1,242.11	7.0%
22	BANK_B	$69.02	6.8%
23	BANK_A	$127.76	7.0%
23	BANK_B	$92.40	6.8%
24	BANK_A	$290.25	7.0%
24	BANK_B	$68.90	6.8%
25	BANK_A	$74.18	7.0%
25	BANK_B	$88.21	6.8%
26	BANK_A	$83.10	6.8%
26	BANK_B	$151.23	6.8%
27	BANK_A	$89.60	6.8%
27	BANK_B	$95.80	6.8%
28	BANK_A	$87.05	6.8%
28	BANK_B	$122.67	6.8%
29	BANK_A	$106.49	6.8%
29	BANK_B	$95.05	6.8%
30	BANK_A	$92.87	6.8%
30	BANK_B	$68.02	6.8%
31	BANK_A	$101.46	6.8%
31	BANK_B	$81.16	6.7%
32	BANK_A	$77.13	6.8%

Continued on next page

Table 13 – continued from previous page

Iteration	Agent	Cost	Liquidity
32	BANK_B	$812.94	6.6%
33	BANK_A	$71.47	6.8%
33	BANK_B	$149.65	6.6%
34	BANK_A	$86.48	6.8%
34	BANK_B	$146.90	6.6%
35	BANK_A	$68.22	6.8%
35	BANK_B	$177.67	6.6%
36	BANK_A	$82.20	6.8%
36	BANK_B	$103.09	6.6%
37	BANK_A	$72.69	6.8%
37	BANK_B	$283.89	6.6%
38	BANK_A	$105.73	6.7%
38	BANK_B	$66.33	6.6%
39	BANK_A	$69.29	6.7%
39	BANK_B	$70.93	6.5%
40	BANK_A	$82.30	6.7%
40	BANK_B	$688.53	6.4%
41	BANK_A	$73.26	6.9%
41	BANK_B	$169.62	6.4%
42	BANK_A	$344.34	6.8%
42	BANK_B	$65.13	6.4%
43	BANK_A	$482.38	6.8%
43	BANK_B	$65.67	6.4%
44	BANK_A	$219.95	6.8%
44	BANK_B	$81.49	6.3%
45	BANK_A	$67.68	6.8%
45	BANK_B	$659.08	6.3%
46	BANK_A	$85.96	6.8%
46	BANK_B	$108.63	6.3%
47	BANK_A	$704.88	6.8%
47	BANK_B	$65.53	6.3%
48	BANK_A	$296.62	6.8%
48	BANK_B	$66.37	6.3%
49	BANK_A	$78.73	6.8%
49	BANK_B	$196.58	6.2%

C.2 Pass 2

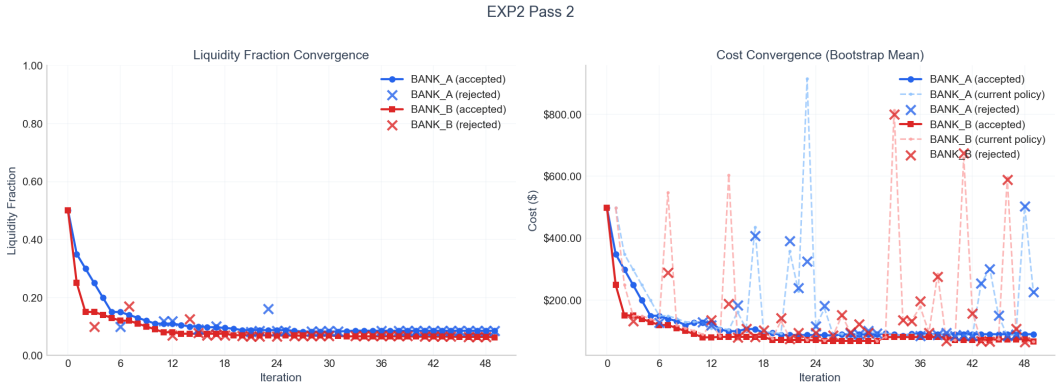

Figure 9: Experiment 2: Stochastic Environment - Pass 2 convergence

Table 14: Experiment 2: Stochastic Environment - Pass 2

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$498.00	50.0%
Baseline	BANK_B	$498.00	50.0%
0	BANK_A	$498.00	50.0%
0	BANK_B	$498.00	50.0%
1	BANK_A	$348.60	35.0%
1	BANK_B	$249.00	25.0%
2	BANK_A	$298.80	30.0%
2	BANK_B	$149.40	15.0%
3	BANK_A	$249.00	25.0%
3	BANK_B	$149.40	15.0%
4	BANK_A	$199.20	20.0%
4	BANK_B	$139.44	14.0%
5	BANK_A	$149.84	15.0%
5	BANK_B	$129.48	13.0%
6	BANK_A	$149.40	15.0%
6	BANK_B	$548.36	12.0%
7	BANK_A	$140.54	14.0%
7	BANK_B	$119.52	12.0%
8	BANK_A	$129.48	13.0%
8	BANK_B	$109.56	11.0%
9	BANK_A	$129.99	12.0%
9	BANK_B	$100.61	10.0%
10	BANK_A	$128.05	11.0%
10	BANK_B	$89.64	9.0%
11	BANK_A	$109.56	11.0%
11	BANK_B	$120.59	8.0%
12	BANK_A	$109.56	11.0%

Continued on next page

Table 14 – continued from previous page

Iteration	Agent	Cost	Liquidity
12	BANK_B	$84.61	8.0%
13	BANK_A	$104.64	10.5%
13	BANK_B	$603.95	7.5%
14	BANK_A	$161.57	10.0%
14	BANK_B	$74.76	7.5%
15	BANK_A	$107.39	10.0%
15	BANK_B	$101.82	7.5%
16	BANK_A	$435.60	9.8%
16	BANK_B	$80.46	7.5%
17	BANK_A	$97.08	9.8%
17	BANK_B	$99.14	7.5%
18	BANK_A	$95.61	9.5%
18	BANK_B	$76.15	7.5%
19	BANK_A	$92.16	9.2%
19	BANK_B	$132.77	7.0%
20	BANK_A	$358.12	8.8%
20	BANK_B	$77.86	7.0%
21	BANK_A	$229.23	8.8%
21	BANK_B	$97.13	7.0%
22	BANK_A	$914.93	8.8%
22	BANK_B	$71.15	7.0%
23	BANK_A	$114.79	8.8%
23	BANK_B	$91.48	6.9%
24	BANK_A	$173.05	8.8%
24	BANK_B	$68.76	6.9%
25	BANK_A	$87.72	8.8%
25	BANK_B	$86.01	6.9%
26	BANK_A	$91.77	8.5%
26	BANK_B	$150.25	6.9%
27	BANK_A	$91.88	8.2%
27	BANK_B	$94.50	6.9%
28	BANK_A	$88.68	8.2%
28	BANK_B	$121.58	6.9%
29	BANK_A	$99.88	8.2%
29	BANK_B	$95.22	6.9%
30	BANK_A	$92.82	8.2%
30	BANK_B	$68.62	6.9%
31	BANK_A	$92.70	8.2%
31	BANK_B	$81.12	6.8%
32	BANK_A	$88.92	8.4%
32	BANK_B	$812.94	6.6%
33	BANK_A	$85.20	8.6%
33	BANK_B	$135.58	6.6%
34	BANK_A	$90.94	8.5%
34	BANK_B	$133.66	6.6%

Continued on next page

Table 14 – continued from previous page

Iteration	Agent	Cost	Liquidity
35	BANK_A	$84.84	8.5%
35	BANK_B	$177.67	6.6%
36	BANK_A	$89.92	8.5%
36	BANK_B	$94.99	6.6%
37	BANK_A	$87.83	8.5%
37	BANK_B	$283.89	6.6%
38	BANK_A	$94.91	8.5%
38	BANK_B	$66.33	6.6%
39	BANK_A	$84.72	8.5%
39	BANK_B	$71.89	6.6%
40	BANK_A	$87.04	8.5%
40	BANK_B	$659.82	6.5%
41	BANK_A	$85.21	8.5%
41	BANK_B	$159.79	6.5%
42	BANK_A	$254.35	8.5%
42	BANK_B	$67.79	6.5%
43	BANK_A	$300.64	8.5%
43	BANK_B	$65.23	6.5%
44	BANK_A	$149.41	8.5%
44	BANK_B	$74.53	6.5%
45	BANK_A	$84.72	8.5%
45	BANK_B	$595.24	6.4%
46	BANK_A	$89.33	8.5%
46	BANK_B	$107.76	6.4%
47	BANK_A	$503.13	8.5%
47	BANK_B	$63.72	6.4%
48	BANK_A	$226.15	8.5%
48	BANK_B	$67.05	6.4%
49	BANK_A	$88.50	8.5%
49	BANK_B	$148.62	6.3%

C.3 Pass 3

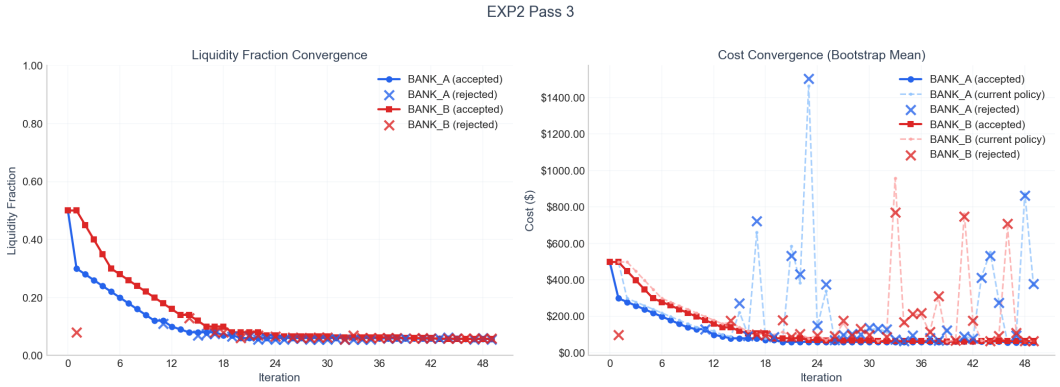

Figure 10: Experiment 2: Stochastic Environment - Pass 3 convergence

Table 15: Experiment 2: Stochastic Environment - Pass 3

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$498.00	50.0%
Baseline	BANK_B	$498.00	50.0%
0	BANK_A	$498.00	50.0%
0	BANK_B	$498.00	50.0%
1	BANK_A	$298.80	30.0%
1	BANK_B	$498.00	50.0%
2	BANK_A	$278.88	28.0%
2	BANK_B	$448.20	45.0%
3	BANK_A	$258.96	26.0%
3	BANK_B	$398.40	40.0%
4	BANK_A	$239.04	24.0%
4	BANK_B	$348.60	35.0%
5	BANK_A	$219.12	22.0%
5	BANK_B	$298.80	30.0%
6	BANK_A	$199.20	20.0%
6	BANK_B	$278.88	28.0%
7	BANK_A	$179.28	18.0%
7	BANK_B	$258.96	26.0%
8	BANK_A	$159.36	16.0%
8	BANK_B	$239.04	24.0%
9	BANK_A	$141.88	14.0%
9	BANK_B	$219.12	22.0%
10	BANK_A	$129.24	12.0%
10	BANK_B	$199.20	20.0%
11	BANK_A	$119.52	12.0%
11	BANK_B	$179.28	18.0%
12	BANK_A	$99.60	10.0%

Continued on next page

Table 15 – continued from previous page

Iteration	Agent	Cost	Liquidity
12	BANK_B	$159.36	16.0%
13	BANK_A	$89.64	9.0%
13	BANK_B	$166.79	14.0%
14	BANK_A	$222.85	8.0%
14	BANK_B	$139.44	14.0%
15	BANK_A	$98.01	8.0%
15	BANK_B	$119.76	12.0%
16	BANK_A	$660.68	8.0%
16	BANK_B	$105.57	10.0%
17	BANK_A	$79.68	8.0%
17	BANK_B	$102.06	10.0%
18	BANK_A	$84.60	7.0%
18	BANK_B	$99.60	10.0%
19	BANK_A	$69.72	7.0%
19	BANK_B	$114.10	8.0%
20	BANK_A	$584.18	6.0%
20	BANK_B	$89.68	8.0%
21	BANK_A	$384.79	6.0%
21	BANK_B	$103.56	8.0%
22	BANK_A	$1,464.39	6.0%
22	BANK_B	$79.68	8.0%
23	BANK_A	$142.58	6.0%
23	BANK_B	$94.76	7.0%
24	BANK_A	$338.92	6.0%
24	BANK_B	$70.94	7.0%
25	BANK_A	$70.16	6.0%
25	BANK_B	$92.40	6.5%
26	BANK_A	$97.81	6.0%
26	BANK_B	$162.47	6.5%
27	BANK_A	$97.82	6.0%
27	BANK_B	$95.79	6.5%
28	BANK_A	$86.73	6.0%
28	BANK_B	$127.63	6.5%
29	BANK_A	$128.02	6.0%
29	BANK_B	$95.47	6.5%
30	BANK_A	$131.51	6.0%
30	BANK_B	$65.14	6.5%
31	BANK_A	$116.39	6.0%
31	BANK_B	$83.30	6.2%
32	BANK_A	$75.69	6.0%
32	BANK_B	$958.00	6.2%
33	BANK_A	$67.55	6.0%
33	BANK_B	$160.58	6.2%
34	BANK_A	$90.06	6.0%
34	BANK_B	$210.88	6.2%

Continued on next page

Table 15 – continued from previous page

Iteration	Agent	Cost	Liquidity
35	BANK_A	$60.30	6.0%
35	BANK_B	$209.37	6.2%
36	BANK_A	$80.84	5.9%
36	BANK_B	$112.61	6.2%
37	BANK_A	$65.63	5.9%
37	BANK_B	$305.89	6.2%
38	BANK_A	$121.10	5.9%
38	BANK_B	$63.02	6.2%
39	BANK_A	$63.52	5.9%
39	BANK_B	$68.72	6.0%
40	BANK_A	$90.92	5.8%
40	BANK_B	$762.87	6.0%
41	BANK_A	$76.82	5.8%
41	BANK_B	$173.25	6.0%
42	BANK_A	$419.31	5.8%
42	BANK_B	$66.46	6.0%
43	BANK_A	$550.59	5.8%
43	BANK_B	$64.63	5.8%
44	BANK_A	$283.08	5.8%
44	BANK_B	$94.72	5.8%
45	BANK_A	$58.74	5.8%
45	BANK_B	$691.13	5.8%
46	BANK_A	$93.98	5.7%
46	BANK_B	$113.72	5.8%
47	BANK_A	$874.88	5.7%
47	BANK_B	$62.82	5.8%
48	BANK_A	$370.30	5.7%
48	BANK_B	$65.30	5.8%
49	BANK_A	$70.88	5.7%
49	BANK_B	$309.02	5.8%

D Experiment 3: Symmetric Scenario - Detailed Results

This appendix provides iteration-by-iteration results and convergence charts for all three passes of experiment 3: symmetric scenario.

D.1 Pass 1

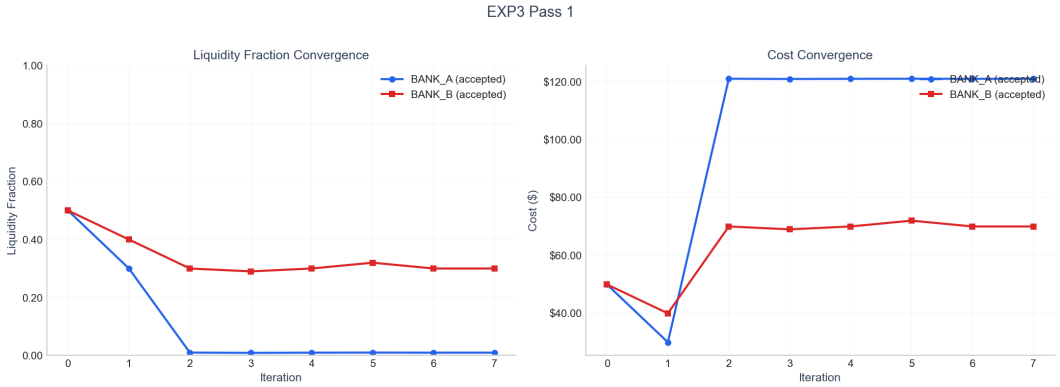

Figure 11: Experiment 3: Symmetric Scenario - Pass 1 convergence

Table 16: Experiment 3: Symmetric Scenario - Pass 1

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$49.95	50.0%
Baseline	BANK_B	$49.95	50.0%
0	BANK_A	$49.95	50.0%
0	BANK_B	$49.95	50.0%
1	BANK_A	$29.97	30.0%
1	BANK_B	$39.96	40.0%
2	BANK_A	$120.99	1.0%
2	BANK_B	$69.97	30.0%
3	BANK_A	$120.90	0.9%
3	BANK_B	$68.98	29.0%
4	BANK_A	$120.96	1.0%
4	BANK_B	$69.97	30.0%
5	BANK_A	$120.99	1.0%
5	BANK_B	$71.98	32.0%
6	BANK_A	$120.96	1.0%
6	BANK_B	$69.97	30.0%
7	BANK_A	$120.99	1.0%
7	BANK_B	$69.97	30.0%

D.2 Pass 2

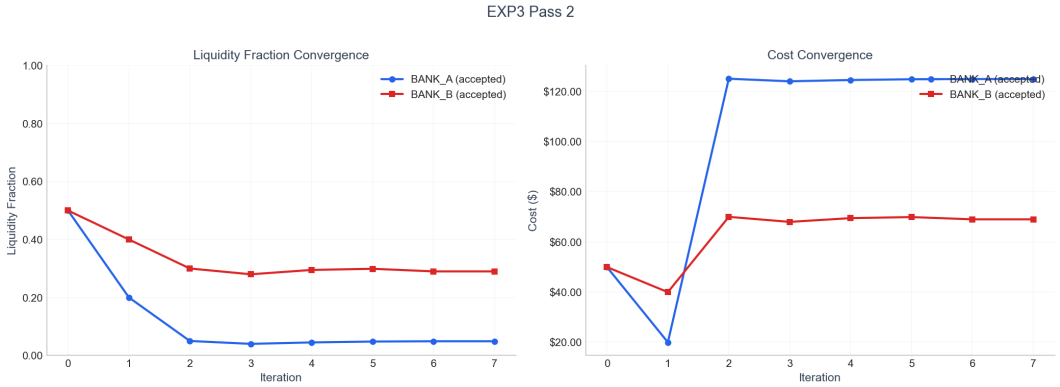

Figure 12: Experiment 3: Symmetric Scenario - Pass 2 convergence

Table 17: Experiment 3: Symmetric Scenario - Pass 2

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$49.95	50.0%
Baseline	BANK_B	$49.95	50.0%
0	BANK_A	$49.95	50.0%
0	BANK_B	$49.95	50.0%
1	BANK_A	$19.98	20.0%
1	BANK_B	$39.96	40.0%
2	BANK_A	$125.01	5.0%
2	BANK_B	$69.97	30.0%
3	BANK_A	$123.99	4.0%
3	BANK_B	$67.96	28.0%
4	BANK_A	$124.50	4.5%
4	BANK_B	$69.46	29.5%
5	BANK_A	$124.80	4.8%
5	BANK_B	$69.88	29.9%
6	BANK_A	$124.89	4.9%
6	BANK_B	$68.98	29.0%
7	BANK_A	$124.89	4.9%
7	BANK_B	$68.98	29.0%

D.3 Pass 3

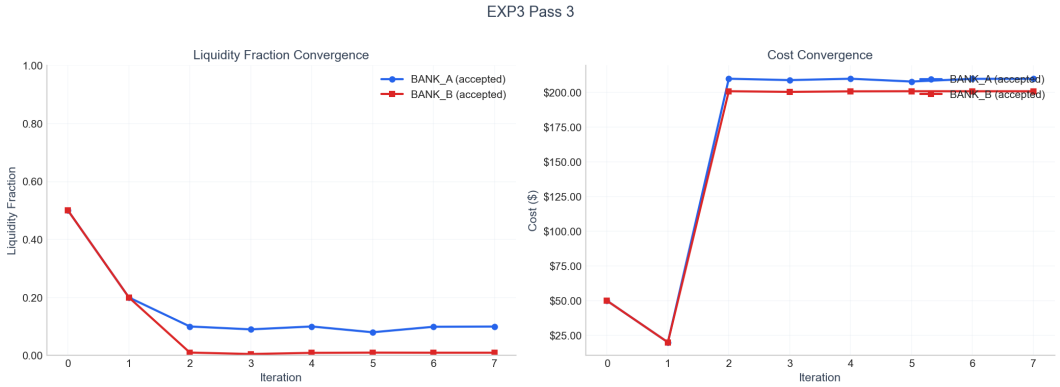

Figure 13: Experiment 3: Symmetric Scenario - Pass 3 convergence

Table 18: Experiment 3: Symmetric Scenario - Pass 3

Iteration	Agent	Cost	Liquidity
Baseline	BANK_A	$49.95	50.0%
Baseline	BANK_B	$49.95	50.0%
0	BANK_A	$49.95	50.0%
0	BANK_B	$49.95	50.0%
1	BANK_A	$19.98	20.0%
1	BANK_B	$19.98	20.0%
2	BANK_A	$209.99	10.0%
2	BANK_B	$200.99	1.0%
3	BANK_A	$209.00	9.0%
3	BANK_B	$200.51	0.5%
4	BANK_A	$209.99	10.0%
4	BANK_B	$200.90	0.9%
5	BANK_A	$207.98	8.0%
5	BANK_B	$200.99	1.0%
6	BANK_A	$209.90	9.9%
6	BANK_B	$200.96	0.9%
7	BANK_A	$209.96	10.0%
7	BANK_B	$200.96	0.9%

E LLM System Prompt Documentation

This appendix documents the system prompt provided to LLM agents during policy optimization. The content is extracted programmatically from the SimCash codebase to ensure this documentation remains synchronized with the actual implementation.

The system prompt establishes the agent's role, provides domain context, and specifies the format requirements for policy proposals. Additional sections (policy schema, cost rates, and constraint-specific guidance) are injected dynamically based on experiment configuration.

E.1 Expert Introduction

The prompt begins by establishing the agent's role as a payment system optimization expert:

```
You are an expert in payment system optimization.
Your job is to generate valid JSON policies for the SimCash payment simulator.

You are an optimization agent in a simulation of an interbank payment settlement
in a real-time gross settlement (RTGS) environment. Each agent represents a bank
with a settlement account at the central bank.
```

E.2 Domain Context

The agent receives detailed context about RTGS settlement mechanics:

```
## Domain Context: Interbank Payment Settlement

### Real-Time Gross Settlement (RTGS)
Payments arrive throughout the simulated trading day with specified amounts,
counterparties, deadlines, and priority levels. Settlement occurs immediately
when the sending bank has sufficient balance or available credit.

### Queuing Mechanism
When liquidity is insufficient, payments enter a queue:
- **Queue 1**: Immediate settlement attempts (holds payments briefly)
- **Queue 2**: Longer-term holding when liquidity is constrained

Queued payments accumulate delay costs until settled.
### Key Concepts
- **Balance**: Current reserves in settlement account (integer cents)
- **Effective Liquidity**: Balance + credit limit - pending obligations
- **Credit Limit**: Available daylight overdraft or collateralized credit
- **Collateral**: Assets posted to central bank to secure credit
```

E.3 Cost Structure and Objectives

The optimization objective and cost components are explained:

```
## Cost Structure and Objectives

**Your objective is to minimize total cost.**

Costs include:
1. **Overdraft Charges**: Basis points on negative balance positions
2. **Delay Penalties**: Per-tick costs for each transaction waiting in queue
3. **Deadline Penalties**: One-time charge when a payment becomes overdue
4. **Overdue Multiplier**: Increased delay costs after deadline passes
5. **End-of-Day Penalties**: Severe charges for unsettled payments at close
```

```
### Why This is Non-Trivial
Actions have delayed consequences:
- Releasing liquidity early may reduce delay cost but increase overdraft exposure
- Holding payments preserves liquidity but risks deadline penalties
- Optimal behavior requires balancing immediate costs against future states

### Strategic Considerations
- High-priority payments have higher delay costs
- Payments close to deadline should often be released
- Incoming payments may provide liquidity to release queued payments
- End-of-day penalties are typically very high - avoid unsettled transactions
```

E.4 Optimization Process

The iterative optimization workflow is described:

```
## Optimization Process

You will be provided with:
1. **Current policy tree**: The policy to improve
2. **Simulation output**: Tick-by-tick logs from recent runs
3. **Iteration history**: How the policy has evolved and cost changes

### Your Task
Analyze the provided data, identify inefficiencies or suboptimal decisions,
and propose modifications to the policy tree that reduce total costs.

Focus on:
- Decisions that led to deadline penalties or high delay costs
- Opportunities to release payments earlier with available liquidity
- Conditions that are too aggressive (causing overdrafts) or too conservative

### Output Requirements
Return a complete, valid JSON policy with:
- All tree types that are enabled in this scenario
- All parameters defined before they are referenced
- Unique node_id for every node
```

E.5 Pre-Generation Checklist

Before generating policies, agents must verify compliance:

```
################################################################################
#                      MANDATORY PRE-GENERATION CHECKLIST                      #
################################################################################

BEFORE generating ANY policy, verify you will satisfy ALL of these:

  [ ] Every {"param": "X"} has a matching "X" key in the "parameters" object
```

```
  [ ] Every action matches its tree type (see allowed actions below)
  [ ] Every node has a unique "node_id" string
  [ ] Arithmetic expressions are wrapped in {"compute": {...}}
  [ ] Only use fields and parameters from the ALLOWED sections
  [ ] No undefined field references
  [ ] No mixing of action types between trees

################################################################################
```

E.6 Final Instructions

The prompt concludes with output requirements:

```
################################################################################
#                          FINAL INSTRUCTIONS                                  #
################################################################################

1. Generate a COMPLETE policy JSON with all required trees
2. Ensure EVERY node has a unique node_id
3. Define ALL parameters before referencing them
4. Use ONLY allowed actions for each tree type
5. Wrap ALL arithmetic in {"compute": {...}}
6. Keep trees reasonably simple (3-5 levels max) for robustness
7. Focus improvements on areas identified in the simulation output

The policy MUST be syntactically valid JSON that passes validation.
```

E.7 Dynamic Sections

The following sections are injected dynamically based on experiment configuration. We show examples from Experiment 2 (12-Period Stochastic) to illustrate the content.

E.7.1 Experiment Customization (Exp 2)

Each experiment can provide scenario-specific guidance:

```
################################################################################
#                       EXPERIMENT CUSTOMIZATION                               #
################################################################################

This scenario tests a fundamental tradeoff in payment systems:
- Allocating liquidity from the pool allows you to settle payments
- But allocated liquidity has an opportunity cost

The KEY DECISION in this scenario is: What fraction of the liquidity pool
should be allocated at the START of the day?

IMPORTANT: Focus your optimization on the `initial_liquidity_fraction` parameter.
The payment_tree should remain a simple Release action - the real optimization
```

```
is in how much liquidity to commit upfront.

With a hard liquidity constraint (no overdraft allowed), you must have
sufficient balance to settle each payment. Incoming payments from the
counterparty provide liquidity that can be recycled for your outgoing payments.

################################################################################
```

E.7.2 Policy Constraints (Exp 2)

The policy schema is filtered to show only allowed elements:

```
### ALLOWED PARAMETERS

Define these in the ‘parameters‘ object:

  initial_liquidity_fraction (float): 0.0 to 1.0
    Fraction of liquidity_pool to allocate at simulation start. Value between 0.0 and 1.0.

### ALLOWED FIELDS

Reference with {"field": "name"}:

  - system_tick_in_day
  - balance
  - amount
  - remaining_amount
  - ticks_to_deadline

### ALLOWED ACTIONS BY TREE TYPE

  payment_tree:  Release, Hold
  bank_tree:  NoAction
```

E.7.3 Cost Parameters (Exp 2)

Current cost rates from the experiment configuration:

```
## COST PARAMETERS

Per-Tick Costs:
  - liquidity_cost_per_tick_bps: 83 (0.1 / 12 ticks)
  - delay_cost_per_tick_per_cent: 0.2

One-Time Costs:
  - deadline_penalty: 50,000 cents
  - eod_penalty_per_transaction: 100,000 cents

Disabled in this scenario:
```

```
  - overdraft_bps_per_tick: 0 (hard liquidity constraint)
  - collateral_cost_per_tick_bps: 0 (using liquidity_pool mode)
```

E.8 User Prompt Example (Exp 2, Pass 2, Iteration 6)

Each iteration, agents receive a comprehensive user prompt containing their performance history and current context. The full prompt (∼10,000 tokens) includes the following sections. We show condensed excerpts from BANK_A's prompt at iteration 6 of Experiment 2, Pass 2:

E.8.1 Current State Summary

````
POLICY OPTIMIZATION CONTEXT - BANK_A - ITERATION 6


This document provides complete context for optimizing YOUR payment policy.
Analyze the simulation outputs and historical data to identify improvements.

NOTE: You are optimizing policy for BANK_A ONLY. Focus on YOUR decisions.

TABLE OF CONTENTS:
1. Current State Summary
2. Cost Analysis
3. Optimization Guidance
4. Simulation Output
5. Full Iteration History
6. Parameter Trajectories
7. Final Instructions

## 1. CURRENT STATE SUMMARY

### Performance Metrics (Iteration 6)

| Metric | Value |
|--------|-------|
| **Mean Total Cost** | $14,984 (0.3% from previous) |
| **Cost Std Dev** | ś$1,046 |
| **Sample Cost** | $12,948 (Seed #1547979735) |
| **Settlement Rate** | 0.0% |
| **Failure Rate** | 0% |

### Current Policy Parameters (BANK_A)

```json
{
  "initial_liquidity_fraction": 0.15
}
```
````

E.8.2 Cost Breakdown and Rates

```
## 2. COST ANALYSIS

### Cost Breakdown (Last Iteration)

| Cost Type | Amount | % of Total | Priority |
|-----------|--------|------------|----------|
| delay_cost | $22 | 100.0% |  HIGH |
| overdraft_cost | $0 | 0.0% |  LOW |
| deadline_penalty | $0 | 0.0% |  LOW |
| eod_penalty | $0 | 0.0% |  LOW |

### Cost Rate Configuration
‘‘‘json
{
  "overdraft_bps_per_tick": 0.0,
  "delay_cost_per_tick_per_cent": 0.2,
  "collateral_cost_per_tick_bps": 0.0,
  "eod_penalty_per_transaction": 100000,
  "deadline_penalty": 50000,
  "split_friction_cost": 0,
  "overdue_delay_multiplier": 5.0,
  "liquidity_cost_per_tick_bps": 83.0
}
‘‘‘
```

E.8.3 Iteration History and Parameter Trajectories

```
## 5. FULL ITERATION HISTORY

### Metrics Summary Table

| Iter | Status | Mean Cost | Std Dev | Settlement | Best Seed | Worst Seed |
|------|--------|-----------|---------|------------|-----------|------------|
| 1 |  BEST | $34,860 | ś$0 | 0.0% | $0 | $0 |
| 2 |  BEST | $29,880 | ś$0 | 0.0% | $0 | $0 |
| 3 |  BEST | $24,900 | ś$0 | 0.0% | $0 | $0 |
| 4 |  BEST | $19,920 | ś$0 | 0.0% | $0 | $0 |
| 5 |  BEST | $14,940 | ś$0 | 0.0% | $0 | $0 |

### Current Best Policy
The best policy so far was discovered in **iteration 5** with mean cost **$14,940**.


### Detailed Changes Per Iteration

####  Iteration 1 (BEST POLICY)
```

```
**Performance:** Mean cost $34,860, Settlement 0.0%

**BANK_A Parameters at this iteration:**
```json
{
  "initial_liquidity_fraction": 0.35
}
```

Iteration 2 (BEST POLICY)

Performance: Mean cost $29,880, Settlement 0.0%
Comparison: -$49.80 vs best (NEW BEST)

BANK_A Changes:
  - Changed 'initial_liquidity_fraction': 0.35 0.3 (0.05)

BANK_A Parameters at this iteration:
```json
{
  "initial_liquidity_fraction": 0.3
}
```

Iteration 3 (BEST POLICY)

Performance: Mean cost $24,900, Settlement 0.0%
Comparison: -$49.80 vs best (NEW BEST)

BANK_A Changes:
  - Changed 'initial_liquidity_fraction': 0.3 0.25 (0.05)

BANK_A Parameters at this iteration:
```json
{
  "initial_liquidity_fraction": 0.25
}
```

Iteration 4 (BEST POLICY)

Performance: Mean cost $19,920, Settlement 0.0%
Comparison: -$49.80 vs best (NEW BEST)

BANK_A Changes:
  - Changed 'initial_liquidity_fraction': 0.25 0.2 (0.05)
```

```
**BANK_A Parameters at this iteration:**
```json
{
  "initial_liquidity_fraction": 0.2
}
```

Iteration 5 (BEST POLICY)

Performance: Mean cost $14,940, Settlement 0.0%
Comparison: -$49.80 vs best (NEW BEST)

BANK_A Changes:
  - Changed 'initial_liquidity_fraction': 0.2 0.15 (0.05)

BANK_A Parameters at this iteration:
```json
{
  "initial_liquidity_fraction": 0.15
}
```


6. PARAMETER TRAJECTORIES

Track how each BANK_A parameter evolved across iterations:

initial_liquidity_fraction

Iteration	Value
1	0.350
2	0.300
3	0.250
4	0.200
5	0.150

Overall: decreased 57.1% from 0.350 to 0.150
```

E.8.4 Final Instructions

```
## 7. FINAL INSTRUCTIONS

Based on the above analysis, generate an improved policy for **BANK_A** that:

1. **Beats the current best policy** - your policy must have LOWER cost than the best
2. **Maintains 100% settlement rate** - this is non-negotiable
3. **Makes incremental adjustments** - avoid drastic changes unless clearly needed
```

4. **Learns from REJECTED policies** - don't repeat changes that made things worse

 Current Best: Iteration 5 with mean cost $14,940.
Your goal is to beat this. If your policy is worse, it will be rejected and we will continue optimizing from the current best policy.

What to Consider:

- **Simulation trace analysis**: What decisions in seed #1547979735 drove costs?
- **Cost breakdown**: Which cost types (delay, collateral, overdraft) dominate?
- **REJECTED policies**: Why did they fail? What changes should you avoid?
- **Parameter trends**: Which parameters correlate with cost improvements?
- **Trade-offs**: Balance delay costs vs collateral costs vs overdraft costs

Output Requirements:

Generate a complete, valid policy JSON that:
- Defines all parameters before using them
- Uses only allowed fields and actions
- Includes unique node_id for every node
- Wraps arithmetic in {"compute": {...}}

Focus your changes on the areas with highest impact potential. Remember: if your policy is worse than the current best, it will be REJECTED and you'll need to try a different approach.

==
CURRENT POLICY FOR BANK_A
==

```json
{
  "version": "2.0",
  "policy_id": "BANK_A_liquidity_opt_iter6",
  "parameters": {
    "initial_liquidity_fraction": 0.15
  },
  "bank_tree": {
    "type": "action",
    "node_id": "B0_no_action_iter6",
    "action": "NoAction"
  },
  "payment_tree": {
    "type": "action",
    "node_id": "P0_release_all_iter6",
    "action": "Release"
  }
}
```

```
```

```
Current policy:
{
  "version": "2.0",
  "policy_id": "BANK_A_liquidity_opt_iter6",
  "parameters": {
    "initial_liquidity_fraction": 0.15
  },
  "bank_tree": {
    "type": "action",
    "node_id": "B0_no_action_iter6",
    "action": "NoAction"
  },
  "payment_tree": {
    "type": "action",
    "node_id": "P0_release_all_iter6",
    "action": "Release"
  }
}

Generate an improved policy that reduces total cost.
Output ONLY the JSON policy, no explanation.
```

E.8.5 Simulation Trace (Exp 2, Pass 2, BANK_A)

The prompt includes a tick-by-tick simulation trace showing transaction arrivals, settlements, and balance changes. This trace is extracted directly from the database-stored prompt sent to BANK_A, demonstrating agent isolation: only BANK_A's own balance changes are visible (counterparty balances are hidden):

```
 Tick 0

 RTGS Settled (outgoing): BANK_A  BANK_B | $65.13
  Balance: $851.00  $785.87
 Incoming TX 8ccbb45e...: BANK_B  BANK_A
  $70.39 | Priority: 5 | Deadline: Tick 5
 Incoming TX 62723300...: BANK_B  BANK_A
  $35.49 | Priority: 5 | Deadline: Tick 3
 Incoming TX 3e598836...: BANK_B  BANK_A
  $65.24 | Priority: 5 | Deadline: Tick 6
 Outgoing TX 7f70bd71...: BANK_A  BANK_B
  $65.13 | Priority: 5 | Deadline: Tick 7
 Incoming TX e89ee92d...: BANK_B  BANK_A
  $41.03 | Priority: 5 | Deadline: Tick 3
 Received: BANK_B  BANK_A | $65.24
 Received: BANK_B  BANK_A | $70.39
```

```
 Received: BANK_B  BANK_A | $155.47
 Received: BANK_B  BANK_A | $151.16
 RTGS Settled (outgoing): BANK_A  BANK_B | $140.58
  Balance: $785.87  $645.29
 Incoming TX 4f3ff626...: BANK_B  BANK_A
  $151.16 | Priority: 5 | Deadline: Tick 7
 Incoming TX 2a1b2cc2...: BANK_B  BANK_A
  $90.91 | Priority: 5 | Deadline: Tick 8
 Received: BANK_B  BANK_A | $90.91
 Received: BANK_B  BANK_A | $35.49
 Costs: total: $7.06
 Decision: Submit TX d298986b...
 Decision: Submit TX 7f70bd71...
 Received: BANK_B  BANK_A | $41.03
 Incoming TX d4c21a94...: BANK_B  BANK_A
  $155.47 | Priority: 5 | Deadline: Tick 4
 Outgoing TX d298986b...: BANK_A  BANK_B
  $140.58 | Priority: 5 | Deadline: Tick 5

 Tick 1

 RTGS Settled (outgoing): BANK_A  BANK_B | $124.09
  Balance: $937.79  $813.70
 RTGS Settled (outgoing): BANK_A  BANK_B | $62.35
  Balance: $1,254.98  $1,192.63
 RTGS Settled (outgoing): BANK_A  BANK_B | $68.09
  Balance: $813.70  $745.61
 Decision: Submit TX 8ab1abd7...
 Outgoing TX 07bcc93d...: BANK_A  BANK_B
  $68.09 | Priority: 5 | Deadline: Tick 6
 Outgoing TX 0f8adb8d...: BANK_A  BANK_B
  $124.09 | Priority: 5 | Deadline: Tick 9
 Decision: Submit TX 07bcc93d...

[... truncated ...]
```