BANK OF CANADA
BANQUE DU CANADA

Estimating Policy Functions in Payments Systems using Deep Reinforcement Learning

A. Desai[1] H. Du[1] R. Garratt[2] F. Rivadeneyra[1] P. S. Castro[3]

[1]Bank of Canada
[2]University of California Santa Barbara
[3]Google Brain

Motivation

How *should* new HVPS be designed?

- 1. Estimate structural model. 2. Perform counterfactual exercises

How *are* new HVPS designed?

- Empirical evaluation of different design options using historical data

Problem: historical data were generated under different rules, behaviour of participants will surely change

Objective

Approximate the current liquidity management rules of LVTS participants using deep reinforcement learning (DRL).

Objective of the agents is to learn:

- The optimal initial liquidity policy
- The optimal intraday payments policy

Literature

Payments systems theory:

- Bech & Garratt (2003) explore the intraday liquidity management game and equilibria

Agent-based methods:

- Arciero et al. (2009) analyze responses of agents to shocks in RTGS systems
- Galbiati & Soramäki (2011) study an agent-based model of payment systems

Reinforcement learning:

- Bellman (1957) dynamic programming is a direct precursor of RL
- Bertsekas & Tsitsiklis (1996) techniques for approximate DP
- Sutton & Barto (1998) early techniques of reinforcement learning
- Mnih, Volodymyr, et al. (2015) showed human-level control through DRL
- Lots of academic/private sector research: OpenAI, Deep Mind, Google Brain, Mila, etc.

Outline

1. Environment: Payments System
2. Method: Reinforcement Learning
3. Learning Setup & Results
4. Conclusion & Future Work

Environment: Payments System

Environment: RTGS payments system

Real-time gross settlement (RTGS):

- Banks participating in the payments system are required to satisfy exogenous and random payment demands from their clients.
- To make payments, banks require liquidity, which can be allocated from either collateral (costly), or incoming payments (not costly).
- Banks need to choose a policy that balances the cost of initial liquidity and the cost of delay (liquidity-delay tradeoff).
- At the end-of-day, banks are required to satisfy all payment demands. If needed, they can borrow from the central bank at a higher cost.

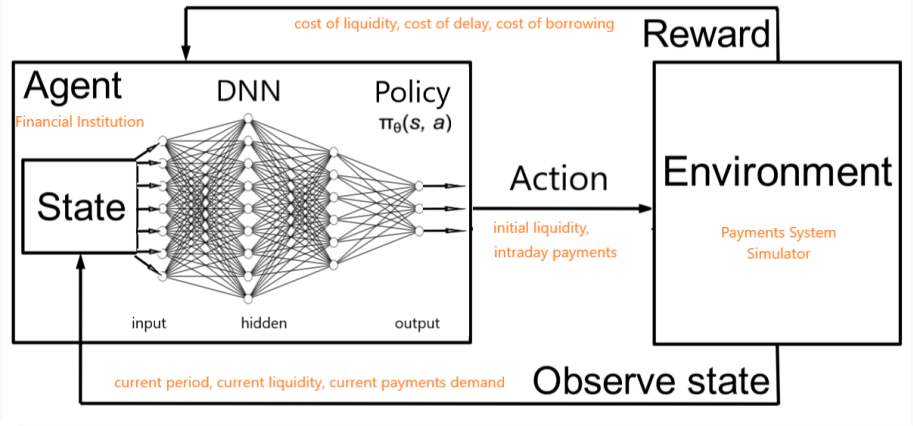
Environment: Timing and definitions

- **Beginning of the day**
  - **Decision**: allocate share $x_0 \in [0, 1]$
  - **Liquidity allocation**: $\ell_0 = x_0 \cdot B$ (B available collateral)
  - **Cost of liquidity**: $r_c \cdot \ell_0$ (r_c is the collateral cost)
- **Intraday periods,** $t = 1, ..., T - 1$:
  - **Payment demand**: P_t (includes delayed payments)
  - **Decision**: send share $x_t \in [0, 1]$
  - **Liquidity constraint**: $P_t x_t \leq \ell_{t-1}$
  - **Liquidity evolves**: $\ell_t = \ell_{t-1} - P_t x_t + R_t$ (R_t received by agent)
  - **Cost of delay**: $\rho_t = P_t(1 - x_t) \cdot r_d$ (r_d is the delay cost)
- **End-of-day borrowing**, $t = T$:
  - **Borrowing from central bank**: ℓ_{cb} (if required to satisfy all P_t)
  - **Cost of borrowing**: $\ell_{cb} \cdot r_b$ (r_b is the borrowing cost)

Method: Reinforcement Learning

Deep Reinforcement Learning: payments system

Deep Reinforcement Learning (DRL): Aims to train agents, parameterized using a deep neural network, to behave optimally in a sequential decision task.



Deep Reinforcement Learning: vanilla policy gradient

The key idea: Push up the probabilities of an agent actions that lead to higher reward, and bring down the probabilities of an agent actions that lead to lower reward, until you arrive at the optimal policy.

- Let $J(\pi_\theta)$ denote the expected finite-horizon undiscounted cost of the policy π_θ. The gradient of the cost function:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \mathcal{R}(s_t, a_t),$$

  where $\mathcal{R}(s_t, a_t)$ denote the cost of executing action a_t from state s_t, period t and τ denote the trajectory, a sequence of state-action-reward(cost) tuples.

- The policy parameters θ are updated after each episode (e):

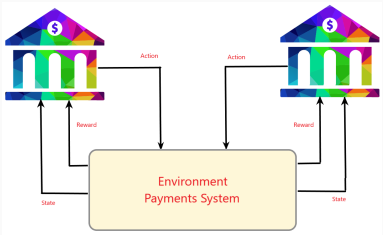
$$\theta_{e+1} = \theta_e + \alpha \nabla_\theta J(\pi_\theta).$$

Learning Setup & Results

Learning intraday payment and initial liquidity

Strategy: is to tackle two simplified learning problems for which we know the optimal policies **if each agent had complete knowledge** of the environment:

1. **Intraday payment decision**: given inexpensive and ample initial liquidity, an agent must learn to send as much as it can in every period in order to reduce the cost of processing all payments.
2. **Initial liquidity decision**: given the above rule for the intraday decision, an agent must learn to choose an initial liquidity that minimizes the cost of processing all payments.

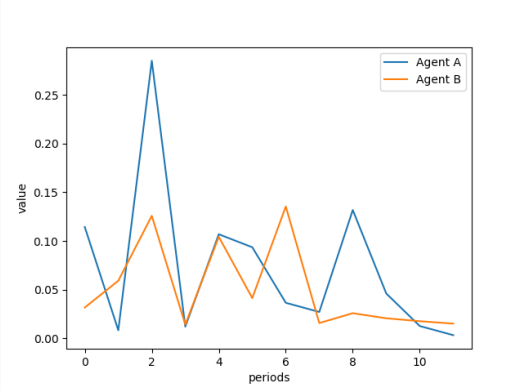


Intraday payment: learning setup

- Both agents are trained simultaneously over multiple episodes using VPG method
- Each episode is divided into twelve periods ($T = 12$)
- For each episode the same payments demand profile is used
- Enough initial liquidity is provided to settle all payments (i.e., $\ell_0 \geq \sum_{t=1}^{T} P_t$)
- There is no cost associated with the initial liquidity allocation (i.e., $r_c = 0$)
- The state space s_t is the vector of current period t, initial liquidity ℓ_0, current period payments demand P_t and current period liquidity ℓ_t. $s_t = \{t, \ell_0, P_t, \ell_t\}$
- The action space a_t is the payment choice, it is the fraction x_t of payments demand P_t in period t. $x_t = \{0,\ 0.1,\ 0.2,\ ...,\ 1\}$
- Per-episode total cost:

$$\mathcal{R} = \sum_{t=1}^{T-1} P_t(1 - x_t) \cdot r_d$$

Intraday payment: payments demand and learning curves



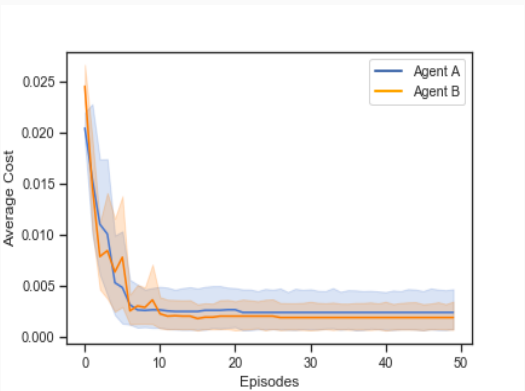


Figure 1: (Left) Per-period payments demand (12-periods) as a percentage of collateral in the LVTS. (Right) Average cost over training episodes for 50-samples with 95% confidence interval.

Intraday payment: agent choices

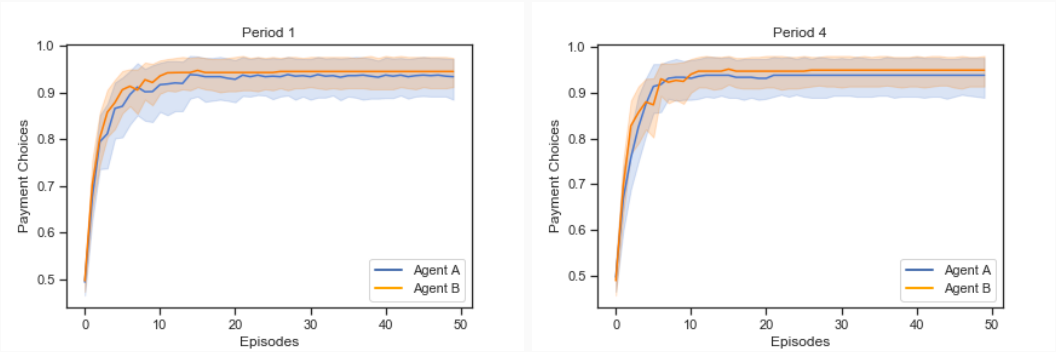


Figure 2: Average payment choices (x_t) in periods 1 and 4 for 50-samples with 95% confidence interval bands. Similar results were observed in all other periods.

Initial liquidity: learning setup

- Both agents are trained simultaneously over multiple episodes using VPG method.
- Each episode has the same payment demands (fraction of discretized action space)
- The intraday payment policy is to send as much as possible:
  If $P_t \leq \ell_{t-1}$ send P_t otherwise send payments equal to ℓ_{t-1}.
- The state is the vector of intraday payments demands (same for each episode).
- Action space is the liquidity choice, it is the fraction x_0 of the available collateral B. $x_0 = \{0,\ 0.05,\ 0.1,\ ...,\ 1\}$. Recall $\ell_0 = x_0 \cdot B$.
- Per-episode cost:

$$\mathcal{R} = r_c \ell_0 + \sum_{t=1}^{T-1} P_t(1 - x_t) \cdot r_d + r_b \ell_{cb},$$

We consider two cases: (1) two intraday periods, (2) 12 intraday periods.

We use: liquidity cost $r_c = 0.05$, delay cost $r_d = 0.01$, and borrowing cost $r_b = 0.1$.

Initial liquidity: problem with two intraday periods

Payment $P^A = [1/19,\ 2/19],\ P^B = [0,\ 3/19]$, Cost $\mathcal{R} = r_c \ell_0 + P_1(1 - x_1) r_d + r_b \ell_{cb}$

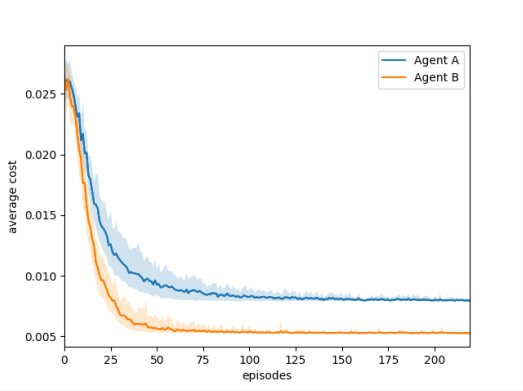


Figure 3: Average cost over training episodes for 50-samples which converges to the optimal for both agents ($\mathcal{R}_A^* = r_c \ell_0^A = 0.00789$, and $\mathcal{R}_B^* = r_c \ell_0^B = 0.00526$). Recall $r_b > r_c$

Initial liquidity: agents' choices

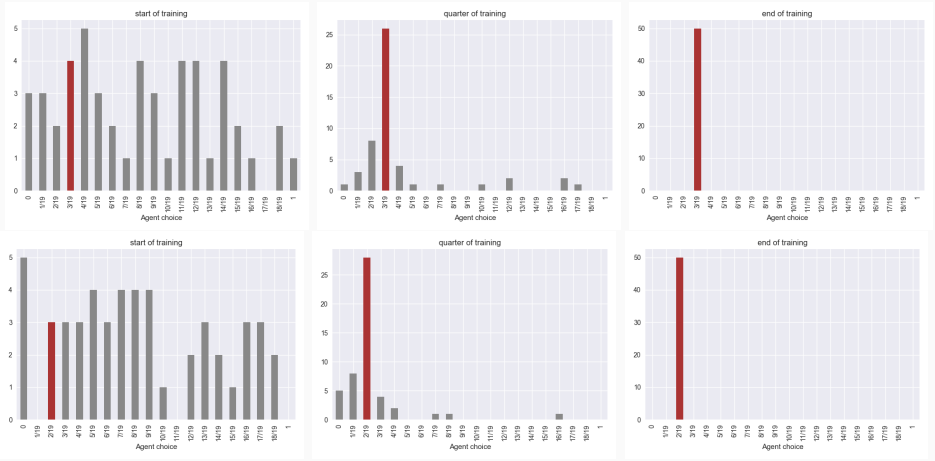


Figure 4: Agent A (top row) and B (bottom row): histograms of choice of initial liquidity at the start (left), quarter (mid) and end (right) of the training (for 50 independent exercises).

Initial liquidity: problem with 12-intraday periods

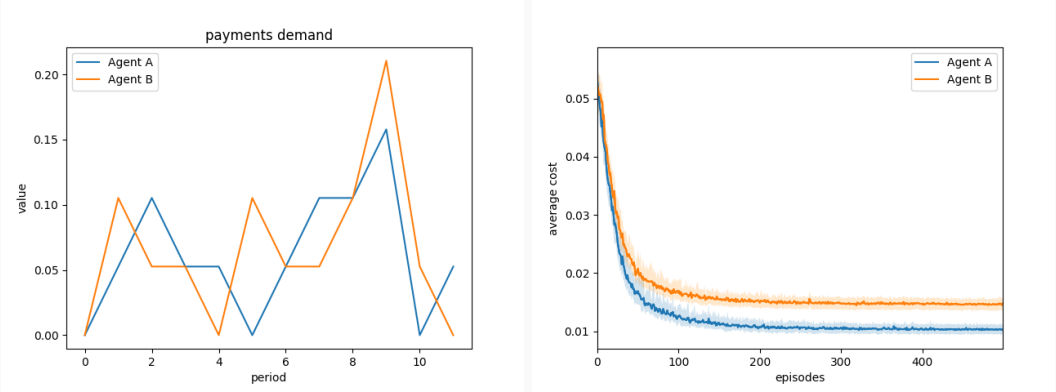


Figure 5: (left) Payments demand. (right) Average reward over training episodes.

Initial liquidity: agents' choices

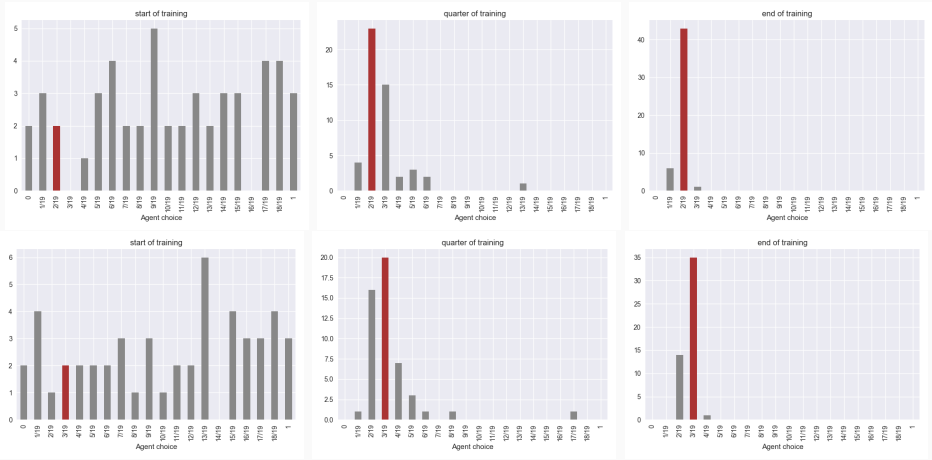


Figure 6: Agent A (top row) and B (bottom row): histograms of choice of initial liquidity at the start (left), quarter (mid) and end (right) of the training (for 50 independent exercises).

Conclusion & Future Work

Conclusions

- Our results demonstrate the applicability of DRL to real-world strategic games
- In the simplified two-player settings, RL agents demonstrate optimal learning behaviour in both intraday payments decision and initial liquidity choices problems
- For some settings, it is hard to verify if agents have learned optimal behaviour

Next Steps

1. Combine initial liquidity and intraday payments decision problems
2. Introduce dynamics: learning over multiple days
3. Introduce non-divisible urgent payments with high delay cost

Thank You!

Questions, comments, suggestions?

Deep Reinforcement Learning: REINFORCE pseudocode

s = [period $\in \mathbb{R}$, initial liquidity $\in \mathbb{R}$, liquidity $\in \mathbb{R}$, payments demand $\in \mathbb{R}$]

a = [initial liquidity $\in \mathbb{R}$, intraday paymets $\mathbb{R}$]

---

1. **initialize** θ (each agents policy parameters) and retrieve initial state s
2. **for** each each episode (day) d = 1, 2, ... **do**
   - i. **draw** payments demands (urgent, no-urgent)
   - ii. **for** $k = 1, ..., K$ (collect trajectories)
     - **for** t = 1, ..., T ($\tau^k = \{s_1^k, a_1^k, r_1^k, ..., a_T^k, r_T^k, s_T^k\}$)
       - a. sample action a_t^k from π_θ for each agent
       - b. perform action a_t^k, updating state $s^{k'}$ and get reward r_t^k
   - iii. **compute gradient** $\nabla_\theta J(\theta) = \frac{1}{k} \sum_k^k \sum_t^T \nabla_\theta \log \pi_\theta(a_t|s_t)\ r(t)$
   - iv. **update** $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$