Estimating Policy Functions in Payment Systems using Reinforcement Learning*

Pablo Samuel Castro
Google Research, Brain Team
Montreal, QC, Canada
psc@google.com

Ajit Desai
Bank of Canada
Ottawa, ON, Canada
adesai@bankofcanada.ca

Han Du
Bank of Canada
Ottawa, ON, Canada
hdu@bankofcanada.ca

Rodney Garratt
University of California Santa Barbara
Santa Barbara, CA, USA
garratt@ucsb.edu

Francisco Rivadeneyra
Bank of Canada
Ottawa, ON, Canada
riva@bankofcanada.ca

Abstract

This paper utilizes reinforcement learning (RL) to approximate the policy rules of banks participating in a high-value payments system. The objective of the agents is to learn a policy function for the choice of amount of liquidity provided to the system at the beginning of the day. Individual choices have complex strategic effects precluding a closed form solution of the optimal policy, except in simple cases. We show that in a simplified two-agent setting, agents using reinforcement learning do learn the optimal policy which minimizes the cost of processing their individual payments. We also show that in more complex settings, both agents learn to reduce their liquidity costs. Our results show the applicability of RL to estimate best-response functions in real-world strategic games.

1 Introduction

High-value payments systems are used to settle transactions between large financial institutions and are considered part of the core national financial infrastructure. For instance, in Canada the Large Value Transfer System (LVTS) processes payment values equivalent to annual GDP every five days. Most high-value payments systems settle in real time and require financial institutions to fund their payments with costly central bank liquidity. Fortunately for banks, they do not have to fund the entire aggregate value of their daily payments. Rather, they can, in part, count on liquidity from incoming payments [1].

Banks make hundreds of payments to each other throughout the day. While banks make some payments related to their own activity, the vast majority of the payments they make reflect underlying customer requests, which are determined from real economic activity. Banks take the payment requests they receive from customers as exogenous. Payment requests from customers must be processed on the day that they are received, but for most payments banks have flexibility in terms of when, during the day, they process payment requests.

*The opinions here are of the authors and do not necessarily reflect the ones of the Bank of Canada or Google. The order of the authors is alphabetical.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Each bank starts the day with an initial liquidity allocation that it arranges with the central bank. As payments come in and out the bank's initial liquidity balance fluctuates. If a bank's liquidity balance falls to zero, it cannot make any payments until new liquidity comes in.This bank suffers a delay cost that is proportional to the period of delay. The delay costs represents customer dissatisfaction with the lack of timely processing of its request. In addition, if at the end of the day, the bank does not have sufficient liquidity to complete its payments, it must borrow additional liquidity from the central bank. The cost of end of day borrowing is greater than the initial liquidity cost.

A key decision that banks make in these systems, is the provision of initial liquidity. This is a strategic decision since the timing of the arrival of incoming liquidity from others depends on how much initial liquidity they post. If others post more liquidity, which means incoming payments are likely to arrive sooner, then a bank can make more of its own payments with recycled liquidity and thus require less of its own liquidity.

In this paper we consider the initial liquidity decision of banks in a specialized large value payment system. Banks receive payment requests from their customers throughout the day. These requests instruct the bank to transfer funds to another bank in the system. The bank executes these requests as so as it can, given available liquidity. Banks avoid delay costs if they post enough liquidity so that they never have to delay payments. But liquidity is costly, so they would like to achieve this goal while posting as little liquidity as possible.

We estimate the best-response functions for liquidity provision in a stylized large value transfer system using machine learning techniques called reinforcement learning (RL). RL is a computational approach to automate learning of sequential decision-making. It emphasizes the learning of an agent by trial and error through the association of actions to states that maximize certain objective in the form of cumulative rewards [2]. RL is particularly suitable for the problem of estimating the policy functions of the participants in a payments system. In RL, the experience provided by trial and error should allow agents to learn to process payments at the lowest possible cost.

We start off with a 2-period model, with known customer payment requests. We then consider a 12-period model in which payment requests are drawn from a distribution of payment profiles that were estimated from the Canadian LVTS data. For the 2-period fixed-payment model, we can derive the best response functions of each bank and compute the Nash equilibrium. This provides a target solution for the RL estimation and allows us to determine unequivocally whether or not the RL procedure works. The 12-period model with randomly drawn payment requests is more realistic; However, we cannot solve it analytically, but, we can solve it using brute-force search, which again provides the target outcomes.

We demonstrate that RL techniques can be used successfully to replicate equilibrium behaviour. In addition to demonstrating the usefulness of RL in guiding participant behaviour, we observe quantitatively the importance of the relative size of the delay and initial liquidity costs for their initial liquidity choices. Having estimates of the sensitivity of the agent best responses across different levels of delay cost is important because delay cost is unobservable to researchers and policy makers. This paper is related to the expanding literature exploring the connections between economics and reinforcement learning [3, 4, 5, 6, 7, 8, 9].

We proceed as follows. In Section 2 we describe the stylized payment system environment we will use in the training of the RL agents. Section 3 discusses the RL algorithm and the multi-agent learning setup. In Section 4 we turn to the training of the initial liquidity problem . We examine two versions of this problem: one with two intraday periods and one with twelve intraday periods. Section 5 provides some concluding remarks.

2 The Payments System Environment

Our environment is a real-time gross settlement (RTGS) payments system. To settle transactions, participants in the system require liquidity provided by the central bank. Transactions are settled in real-time without netting which means that, for example, two offsetting positions between two banks cannot be cancelled out even if they were submitted to the system in the same instant and for the same amount. Banks participating in the system use it to satisfy the exogenous and random payment demands they receive throughout the day from their clients. Banks can source the liquidity required to process the payments from collateral they post at the central bank, which is costly due

to its opportunity cost, or from incoming payments from other banks, which is not. This creates an incentive to delay processing their own payments to wait for incoming payments. On the other hand, delaying payments is costly to banks if they don't service their clients in a timely fashion. Therefore, the bank's problem is to choose a policy that balances the cost of initial liquidity and the cost of delay by choosing the level of initial liquidity and the timing of their own payments, all this conditional on the other bank's own policies. At the end of the day, banks are required to satisfy all payment demands. If they don't have enough liquidity, they can borrow from the central bank at a cost higher than the cost of collateral at the beginning of the day.

In our environment, each day is an episode e which is divided into intraday periods $t = 0, 1, 2, ..., T$. At the beginning of the day $t = 0$, the agent is faced with making the decision to allocate share $x_0 \in [0, 1]$ of collateral $\mathcal{B}$ for its initial liquidity, $\ell_0 = x_0 \cdot \mathcal{B}$. Subsequently, at each intraday period $t = 1, ..., T - 1$ the agent receives payment demands P_t and has to choose the share $x_t \in [0, 1]$ of requested payments to send in that intraday period. Note, for ease of notation, P_t includes any payment demand that was not sent in the previous intraday period.

The agent's choice is constrained by the available liquidity in that intraday period so that $P_t x_t \leq \ell_{t-1}$. At the end of each intraday period the agent receives payments from other agents which we denote by R_t. Then the available liquidity evolves according to: $\ell_t = \ell_{t-1} - P_t x_t + R_t$. The associated cost with the initial liquidity choice is $r_c \cdot \ell_0$ and the cost of delay is given by $r_d \cdot P_t(1 - x_t)$. If the agent does not have enough liquidity at $T - 1$ to satisfy all the remaining payment demands, the agent can borrow from the central bank at rate r_b which is more expensive than the morning liquidity cost r_c. Table 1 summarizes the timeline and decisions of the agent.

Table 1: Timeline, decisions of the agent and constraints of the environment.

Beginning of the day, $t = 0$	
Available collateral:	$\mathcal{B}$
Initial liquidity decision:	$x_0 \in [0, 1]$
Liquidity allocation:	$\ell_0 = x_0 \cdot \mathcal{B}$
Cost of initial liquidity:	$r_c \cdot \ell_0$
Intraday periods, $t = 1, ..., T - 1$	
Payment demand:	P_t
Decision to send:	$x_t \in [0, 1]$
Liquidity constraint:	$P_t x_t \leq \ell_{t-1}$
Agent receives incoming payments:	R_t
Evolution of liquidity:	$\ell_t = \ell_{t-1} - P_t x_t + R_t$
Cost of delay:	$r_d \cdot P_t(1 - x_t)$
End-of-day borrowing, $t = T$	
Borrowing from central bank (for remaining payments):	ℓ_{cb}
Cost of end-of-day borrowing:	$r_b \cdot \ell_{cb}$

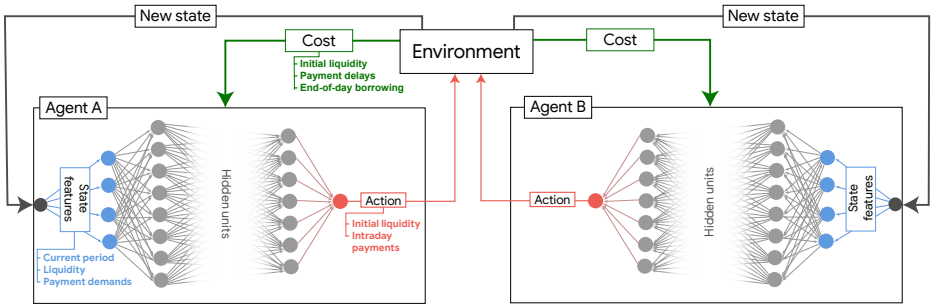
Our model of the environment abstracts from two important dimensions of payment systems: the indivisibility of payments and the interbank market for liquidity. Typically, due to legal restrictions, a payment request can only be fulfilled by the bank in its entirety or not at all. In other words, the payment amount cannot be fulfilled by splitting the payment into smaller amounts. In our learning setup, however, agents are presented with continuous payment demands at each intraday period, and are assumed to pay the largest fraction $x_t \in [0, 1]$ they can afford given available liquidity.Divisible payments may be a reasonable approximation of reality when aggregate payment requests are large relative the size of individual payments. The second aspect we abstract from is that banks participating in actual payment systems can typically source liquidity in an interbank market, and not only from the central bank or from incoming payments. Including these two dimensions of the environment along with more than two agents are avenues of future research.

3 RTGS as a multi-agent reinforcement learning problem

Reinforcement learning methods are used for optimizing the behaviour of an agent in an uncertain environment. The agent improves its behaviour by interacting with the environment, via the selection of *actions* (from a set of possible actions $\mathcal{A}$) in discrete time steps, and transitioning between *states* of the environment (from a set of possible states $\mathcal{S}$). Specifically, while at state $s_t \in \mathcal{S}$ at time step t, the agent selects action $a_t \in \mathcal{A}$; the environment responds with a new state $s_{t+1} \in \mathcal{S}$ and a numerical cost r_t. The goal of the agent is to discover a *policy* $\pi : \mathcal{S} \to \mathcal{A}$ mapping states to actions such that following this policy will minimize the cumulative sum of costs incurred. Traditionally, there is a single agent being trained; in our current work, we train two agents concurrently in the same environment, which results in more complex learning dynamics.

Figure 1 depicts a learning setup of a simplified payment system. Two agents in the payments system interact with the same environment by each making independent action choices. Based on the actions chosen by each agent, the environment will return a cost to each, as well as a new state. The state for each agent could consists of the current period, the agent's current liquidity, and the payment demands. Although the same learning algorithm is used for both agents, each agent executes this algorithm independently and maintains its own policy.

Figure 1: Reinforcement learning in the context of a payments system, where each agent (a separate bank) maintains its own policy, represented as a neural network (in grey), to make the action choices (in red), conditioned on the current state (in blue). The environment provides the agents with the cost (comprised of the initial liquidity, delay and end-of-day borrowing costs in green).



Given the multiple complexities involved with the setting just described, we aim to isolate some of them so as to better explore the learning dynamics and more carefully develop algorithms for handling the complete, independent multi-agent setting. Our strategy for training the agents is to separately tackle the the initial liquidity choice and the intraday payments decision. This strategy allows us to verify that our learning algorithm is performing as expected. The training of the agents proceeds as follows.

For the initial liquidity decision, the agent will be trained while the intraday decision is fixed to send all payments. The optimal initial liquidity policy, given the intraday policy, is the one that minimizes the cost of processing all payments. Intuitively, when the agent behaves optimally in its intraday decision, making the initial liquidity choice requires estimating its own payment demand profile and the payments it expects to receive.

This is akin to the well-known bandit problems [2], with the added complexity that the transition dynamics for each agent is affected by the learning process of the other agent. Under this lens, this bandit setting allows us to evaluate the learning process in the independent multi-agent setting without the compounding effects present when dealing with sequential decision problems.

The learning task is episodic, in our setup a day is divided into multiple intraday periods, which we take to be one hour each. Each episode starts with the beginning of the payment cycle and ends at the end of the payment cycle. In the last intraday period, banks are required to satisfy all payment demands. If they do not have enough liquidity, they will automatically borrow the shortfall amount from the central bank. The central bank is not strategic and will always lend the necessary amount at the fixed borrowing cost.

We simplify further by limiting our training to only two agents simultaneously, as shown in Figure 1. Both agents are transferring payments to each other through the payments system. The payments demand is exogenous and received throughout the day from their clients. To a certain extent, training only two agents is not a big limitation of our exercise if, in reality, banks use only one policy function to manage their payments irrespective of which agent they are sending to and if they do not distinguish from which agent they have received a payment from. Therefore a similar training setup would be agents learning a policy to process payment demands to many other agents as long as the learning agent does not know or use information specific to the recipient.

4 Initial Liquidity Policy Estimation

The learning setup for the initial liquidity policy is as follows:

- **Number of intraday periods:** We will consider two scenarios, one in which each episode is divided into only two intraday periods, $T = 2$, and one in which the episode is divided into twelve intraday periods, $T = 12$. The reason for testing the first scenario is that for this case we have the analytical solution to the policy and therefore we can verify if the learning algorithm is performing as expected.
- **Agents:** Agents A and B are trained simultaneously over multiple episodes (depending on the exercise).
- **Payments demand:** The payments demand profile is drawn from the LVTS data shown in Figure 2.
- **Intraday payment policy:** In the current environment there is no benefit in delaying payments, therefore the optimal intraday policy is to always send all payments. Therefore, both agents intraday policy is to send all the payments if possible or the entire value of liquidity in that intraday period, i.e., if $P_t < \ell_{t-1}$ send $x_t = 1$ otherwise they chose x_t such that payments sent are equal to ℓ_{t-1}.
- **State space:** In each episode the agent observes the entire vector of its intraday payments.
- **Action space:** The action space is the liquidity choice as the fraction x_0 of the available collateral $\mathcal{B}$. We discretize the unit interval for x_0 into 21 evenly-spaced fractions: $\{0,\ 0.05,\ 0.1,\ 0.15,\ ...,\ 1\}$. Recall also that $\ell_0 = x_0 \cdot \mathcal{B}$.
- **Cost:** The total processing cost per episode is:

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

  where r_c is the cost of initial liquidity, r_d is the intraday delay cost and r_b is the end-of-day borrowing cost. The last two terms are included because the agent might incur delay costs and a borrowing cost if its chosen initial liquidity was not sufficient to cover all payments. Note that in the last period, T, the agent borrows an amount equal to its shortage to satisfy that period's payments demand. Therefore, no delay cost occurs in the last period.

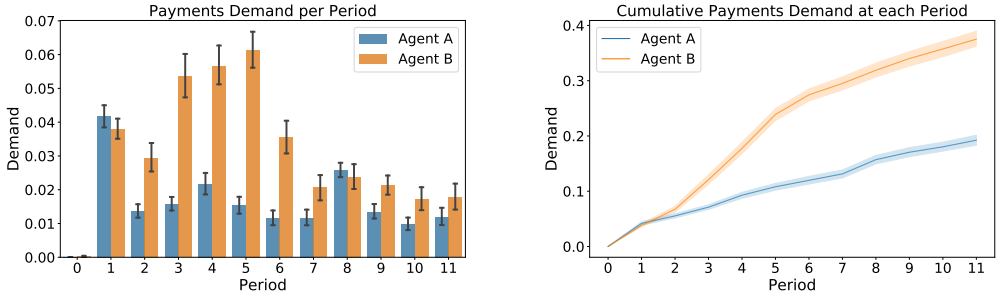
4.1 Data and parameters

As payment demands we use transactions observed between two actual participants in the Canadian Large Value Transfer System (LVTS). Each episode is a daily LVTS payments cycle that runs, typically, from 6 a.m. to 6 p.m. We divide the episodes into 12 hourly periods. We sum the value of transactions between these two participants for each hour; to standardize them, we divide the payment values by the collateral each agent pledged to the system on each particular day. We selected a sample of 380 business days between January 02, 2018 and August 30, 2019. The resulting payment demands are displayed in Figure 2. Agent B's average payment demands and range of payment values are higher than that of agent A's. Agent B's demand peaks in the earlier part of the day before slowing down in the last few periods. To perform the model training and evaluation, we randomly split the sample, reserving 90% for training and 10% for testing.

For initial liquidity decision problem, we consider two scenarios. The first scenario is a problem with only two intraday periods, for which we have the analytical solution of the optimal policy. The

second scenario is a problem with 12 intraday periods. For both scenarios we use the following costs: for the initial liquidity cost $r_c = 0.1$, for the per-period delay cost $r_d = 0.2$, and for the end-of-day borrowing cost $r_b = 0.4$. Compared to the initial liquidity cost, the high end-of-day cost should discourage agents from borrowing at the end of the day instead of allocating sufficient liquidity at the beginning of the day.

Figure 2: Data used for training and testing, showing the hourly sum of payment flows between two participants of the Canadian LVTS. Left plot are the are the hourly averages and standard deviation over the sample period. Right plot is the cumulative payments value. Sample is between January 02, 2018, and August 30, 2019. Payments are standardized daily as a percentage of the collateral pledged to the LVTS system in each given day.



For training, we utilize the REINFORCE algorithm [10] with a RL setup and implement it in Python using TensorFlow [11]. Each agent's policy function π is approximated using a feed-forward neural network. For each episode, we simultaneously generate multiple trajectories using agent's current policy, where each trajectory is a complete path of state, action and reward. Subsequently, using the experience gathered through trajectories, we update the policy function at the end of each episode. This way, we train the agents over multiple episodes and monitor each agent's learning progress with the average cost over the training episodes. To compute the confidence intervals of the payment decisions at each intraday period, we simultaneously perform multiple independent training exercises.

4.2 Results of the training with two intraday periods, $T = 2$

In this scenario, to help in the illustration of the intuition, we use dummy payments demands that have the same support as the discretized action space: $P^A = [0,\ 0.15],\ \ P^B = [0.15,\ 0.05]$. Each entry in the vector represents the demand in each intraday period. In this case the cost function simplifies to $\mathcal{R} = r_c\ell_0 + P_1(1 - x_1)r_d + r_b\ell_{cb}$ for both the agents because in the second period all payment demands have to be satisfied, possibly by borrowing at cost r_b.

Notice that agent B has no incoming payment from A in the first period. Therefore, to avoid delay and borrowing costs (recall $r_c < r_d < r_b$), B has to allocate initial liquidity equal to its total payment demands. If agent B performs the optimal action, then agent A will have an incoming payment in the first period, which is equal to its payment in the second period. Consequently, agent A can use the received payment to meet its demand. Therefore, the optimal choice of A is to allocate nothing. In sum, the optimal liquidity choices for agent A and agent B are 0 and 0.2. Accordingly, the optimal costs are $\mathcal{R}^A = 0$ and $\mathcal{R}^B = 0.2 \times 0.1 = 0.02$.

Figure 3 shows the average training costs (left) and actions (right) for both agents A and B obtained by performing 50 independent training exercises using 50 episodes in each exercises. It can be observed that after initial exploration, both agents quickly learn to minimize the cost by selecting actions closer to their optimal choices. Agent B, which has higher payments demand and has no incoming payments, converged to a higher cost than agent A.

4.3 Results of the training with twelve intraday periods, $T = 12$

Similar to the two-period case, both agents are trained simultaneously while playing against each other, using the REINFORCE algorithm for 100 episodes. In this case; however, both agents are faced with the LVTS payments demand showed earlier in Figure 2. The average cost during training

and testing computed using 50 independent exercises is shown in Figure 4 (left). The cost for both agents plateaus to their minimum observed around the 60^{th} training episode. The confidence intervals over the course of training (at each episode calculated using 50 training exercises) show that both agents do not converge to a single liquidity choice. This is seemingly due to the nonstationarity in the environment. For instance, the total cost associated with the selected action from agent A could sometimes be influenced by the choice of agent B and vice-versa. This could cause uncertainty of payment arrivals in all twelve-periods and hence influence the agent's learning ability.

Figure 4 (right) shows this more clearly. The figure shows the average and range of agents' initial liquidity decisions at various points of the training. At the start of training, both agents chose initial liquidity around $1/2$. As training progresses, both agents reduce their liquidity choice with A showing the largest reduction. Agent B, which has higher payments demands, converged to a higher liquidity choice than agent A. Also, neither agent converged to a single solution by the end of the training, likely due to the uncertainty of payment arrivals form the other participant in each intraday period.

Note that for this more general case with twelve intraday periods the learning is slower compared to the two-period case in terms of training costs. Also, the confidence interval in twelve-period case are wider compared to the two-period case. These demonstrate the complexity involved in generalization of learning with higher intraday periods Also, remember that for twelve intraday periods case, we do not have the analytical solution of the optimal initial liquidity choice and therefore we cannot know what is the minimum equilibrium cost.

A key parameter in the environment is the delay cost that banks face which is unobserved by policymakers. Therefore, we examine the sensitivity of our results to the choice of delay cost, r_d. We performed the robustness exercise varying the cost of delay ranging from 0.01 to 0.3. Recall that in previous exercises $r_c = .1$, $r_d = .2$ and $r_b = .4$. Figure 5 show the box plots of the total cost of each agent at the end of the 50 training episodes for the different level of delay cost. When we vary the delay cost above the initial liquidity cost, i.e. for values above 0.1, agents increase their initial liquidity allocation incurring a larger total training cost, in order to minimize the delay cost. This sensitivity is, however, relatively small. In contrast, the total cost of agent A is very sensitive when delay cost is below 0.1. Agent A is the one with lower payment demands, therefore when delay cost is low enough, it learns to wait for the incoming payments from B, reducing its initial liquidity allocation and its total cost.

The case where $r_d \leq r_c$ is interesting because the best response function of the agents is no longer the one presented in the Section 4, even in cases with only $T = 2$. With this configuration of parameters, the agents might have the incentive to allocate less initial liquidity because if the value of incoming payments is large enough, the reduction of initial liquidity cost could compensate for the delay costs.

5 Conclusion

This paper employs RL to estimate the policy rules of two banks participating in a high-value payments system. Our results show that —in a simplified learning problem for which we know the optimal solution– policy rules trained with the RL algorithm converge to the optimal solution. Using the intuition and confidence in the training algorithm, we generalize to a problem where agents have to simultaneously learn a policy to choose their initial liquidity. Both agents learn to choose the initial liquidity that minimizes their cost of processing all payments. These results show the applicability of RL to estimate best-response functions in real-world strategic games.

The significance of these results are underlined by the fact that the initial liquidity decision of banks is complex even when agents have full knowledge of the environment, the data generating process and their opponent's strategies. Our agents had no knowledge of the environment, the data, or the game itself. From a normative point of view, if agents trained with RL methods can find alternative solutions, regulators could use RL policies to help in their mandates of ensuring safety and efficiency of these systems. Also, understanding these techniques will be relevant for policymakers as banks themselves are likely to attempt to use these algorithms for their liquidity management.

Before these promises can be realized, we need a more complete understanding of the liquidity management rules of financial institutions. This paper is the first in that agenda. Future work involves pursuing the following extensions. First, it is important to explore how would the training perform if agents were learning the initial liquidity and intraday payments problems at the same time. A

major extension will be to introduce some realistic features of the payment system like an intraday market for liquidity and non-divisible or urgent payments. These could be significant departures to the learning setup that might require new algorithms and neural networks. Another ambitious extension is to tackle the multi-agent problem which will require generalizing over more complex sets of payments demands to potentially derive richer policies that account to whom a payment is being sent and from who a payment has been received.

Note, for 12-period case, we get similar performance when tested on the unseen payments demand. Also, we performed a large number of robustness checks on the environment inputs and model parameters. However, due to space constrains, we've not included those results in the paper.

Figure 3: Two intraday periods: Average cost (left) and average initial liquidity decision (right) over training episodes. The solid lines are the averages at each point in the training computed using 50 independent training exercises and shaded areas are the 99% confidence interval bands.

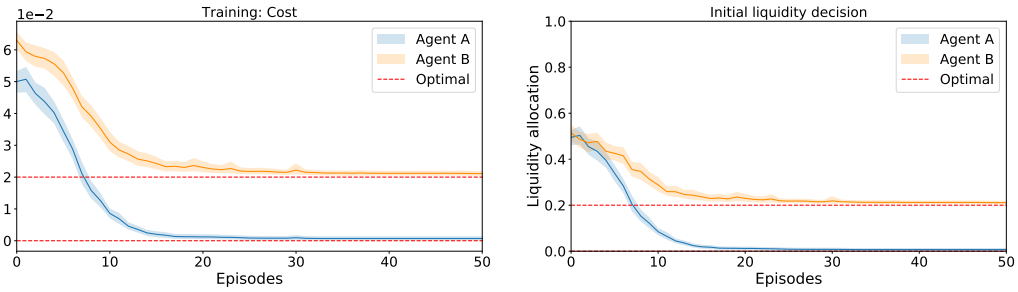


Figure 4: Twelve intraday periods: Average cost (left) and average initial liquidity decision (right) over training episodes. The solid lines are the averages at each point in the training computed using 50 independent training exercises and shaded areas are the 99% confidence interval bands.

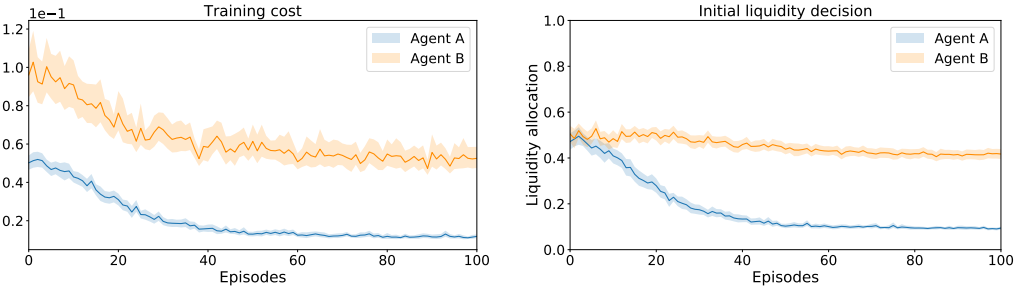
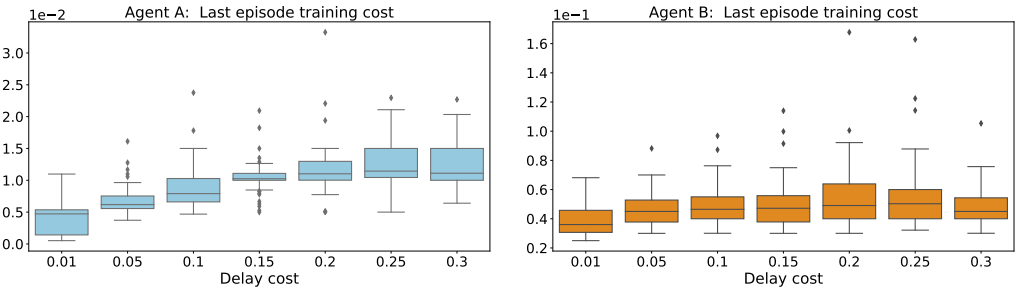


Figure 5: Twelve intraday periods: Boxplots showing the sensitivity of final costs after training across different selections of delay cost. Whiskers are defined as 1.5 · IQR (Interquartile Range).



References

[1] James McAndrews and Samira Rajan. The timing and funding of fedwire funds transfers. *Federal Reserve Bank of New York Economic Policy Review*, 6:17–32, 2000.

[2] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[3] Alvin E Roth and Ido Erev. Learning in extensive-form games: Experimental data and simple dynamic models in the intermediate term. *Games and Economic Behavior*, 8(1):164–212, 1995.

[4] Morten L Bech and Rod Garratt. The intraday liquidity management game. *Journal of Economic Theory*, 109(2):198–219, 2003.

[5] Luca Arciero, Claudia Biancotti, Leandro d'Aurizio, and Claudio Impenna. Exploring agent-based methods for the analysis of payment systems: A crisis model for StarLogo TNG. *Bank of Italy Temi di Discussione (Working Paper) No*, 686, 2008.

[6] Marco Galbiati and Kimmo Soramäki. An agent-based model of payment systems. *Journal of Economic Dynamics and Control*, 35(6):859–875, 2011.

[7] Naoki Funai. Convergence results on stochastic adaptive learning. *Economic Theory*, 68(4):907–934, 2019.

[8] Mitsuru Igami. Artificial intelligence as structural estimation: Deep blue, bonanza, and alphago. *The Econometrics Journal*, 2020.

[9] Stephan Zheng, Alexander Trott, Sunil Srinivasa, Nikhil Naik, Melvin Gruesbeck, David C Parkes, and Richard Socher. The AI Economist: Improving equality and productivity with ai-driven tax policies. *arXiv preprint arXiv:2004.13332*, 2020.

[10] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256, May 1992.

[11] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.