

## **ADVERSARIAL REINFORCEMENT LEARNING: A DUALITY-BASED APPROACH TO SOLVING OPTIMAL CONTROL PROBLEMS**

Nan Chen, Mengzhou Liu, Xiaoyan Wang, and Nanyi Zhang

Dept. of Systems Engineering and Engineering Management, The Chinese University of Hong Kong,  
Hong Kong, CHINA

### **ABSTRACT**

We propose an adversarial deep reinforcement learning (ADRL) algorithm for high-dimensional stochastic control problem. Inspired by the information relaxation duality, ADRL reformulates the control problem as a min-max optimization between policies and adversarial penalties, enforcing non-anticipativity while preserving optimality. Numerical experiments demonstrate ADRL's superior performance to yield tight dual gaps. Our results highlight ADRL's potential as a robust computational framework for high-dimensional stochastic control in simulation-based optimization contexts.

### **1 INTRODUCTION**

Stochastic control establishes a powerful framework for modeling and solving decision-making problems in random environments, where uncertainty unfolds and decisions are made sequentially over time. The principle of dynamic programming provides a mathematically elegant characterization of the structure of optimal policies in control problems. However, the practical implementation of this principle is severely constrained by the curse of dimensionality, making it infeasible to solve many high-dimensional applications directly.

Recently an innovative approach emerges in the literature (e.g., Han and E 2016 and Han et al. 2018), leveraging the high expressivity of deep neural networks combined with Monte Carlo simulation to address high-dimensional control problems. This deep Monte Carlo optimization approach has garnered significant attention for its ability to overcome the limitations of traditional methods. For further theoretical developments, diverse applications, and numerical experiments, we refer to works such as (Becker et al. 2019), (Buehler et al. 2019), (Huré et al. 2021). However, some studies (e.g., Reppen and Soner 2023) suggest that excessively large neural networks may overfit to future randomness rather than accurately estimating it. As a result, the feedback policies trained by this deep Monte Carlo approach tend to outperform the original control problem in-sample by implicitly bypassing the essential adaptiveness restriction. Furthermore, this overfitting issue also causes feedback actions constructed using overly large hypothesis spaces to fail to generalize well and perform poorly out-of-sample.

The above considerations give rise to an intriguing research problem: how can we assess the optimality of approximate control policies, particularly when these policies are derived using complex approximation schemes such as deep neural networks? In this paper, we propose an adversarial deep reinforcement learning (ADRL) framework for constructing deep neural network-based approximations to the true value functions of multi-dimensional stochastic control problems. This novel approach is rooted in the concept of information relaxation and its associated dual formulation within the stochastic dynamic programming literature (see Rogers 2007, Brown et al. 2010, Brown and Smith 2022, and Chen et al. 2024). Drawing inspiration from the principle of strong duality, we find that the dual problem can be reformulated as a zero-sum game between an adversary and an agent. In this game, the adversary seeks to minimize the agent's expected (penalized) reward by strategically selecting an appropriate dual penalty, while the agent

responds by employing non-adaptive policies to maximize its rewards. From this perspective, the agent needs to solve deterministic optimization problems in the dual formulation.

We parameterize the dual penalties using deep neural networks and employ stochastic optimization techniques to develop learning algorithms for identifying the optimal hyperparameters of the dual networks. Both Robbins-Monro (RM) and Kiefer-Wolfowitz (KW) type methods are utilized in this framework. Notably, under certain regularity conditions, we derive a simple, unbiased gradient estimator for the RM method by leveraging the celebrated envelope theorem, which characterizes the differentiability properties of the value function in parameterized optimization problems.

Once the learning process for the dual part is complete, the ADRL algorithm can also generate adaptive control policies. This is achieved by utilizing the learned dual functions as approximations for continuation value functions within the one-step Bellman equation to determine optimal actions. It is important to emphasize that these greedy policies are suboptimal, serving as a lower bound for the true value of the original control problem. By combining these lower bounds with the upper bounds derived from the dual values, the ADRL algorithm is able to construct effective confidence intervals for the true values of the control problem. Numerical experiments demonstrate that ADRL produces tight confidence intervals even for high-dimensional control problems. These findings underscore the efficiency of the ADRL framework in learning high-quality policies with performance guarantees.

The information relaxation-based duality framework has spurred a rapidly growing body of literature. It has been widely used to demonstrate the near-optimality of certain heuristics by evaluating the tightness of the corresponding dual gap across various applications (see Brown and Smith 2022, Chen et al. 2024, and the references therein). However, much of this literature remains silent on how to proceed when the dual gap of a policy under evaluation is found to be loose. Complementing this literature, the main contribution of this paper is the development of a systematic approach for constructing tight dual gaps using deep neural networks, with the ultimate goal of identifying optimal control policies.

Recent advances in adversarial deep learning have primarily focused on efficiently crafting adversarial examples—inputs subtly perturbed to induce model misclassification—to train neural networks and enhance their robustness, ensuring high-confidence performance; see (Goodfellow et al. 2015; Huang et al. 2017). Another prominent technique, Generative Adversarial Networks (GANs) (Goodfellow et al. 2020), employs a min-max formulation to train generator networks through a zero-sum game between generators and discriminators. While our work shares structural similarities with GANs in its use of a min-max optimization framework, it is rooted in a different theoretical foundation and designed for distinct applications. Specifically, our approach leverages information relaxation theory to construct dual values for stochastic control problems. As mentioned earlier, this leads to a systematic approach for constructing tight dual gaps for policy evaluation for solving stochastic control problems.

The remainder of the paper is organized as follows. In Section 2, we review the basics of the SDP duality. Section 3 establishes the ADRL algorithm with discussions on how to implement the Robbins-Monro and Kiefer-Wolfowitz methods. Section 4 is devoted to one numerical experiment in optimal trade execution. Section 5 concludes the paper.

## 2 CONTROL PROBLEM AND INFORMATION RELAXATION

### 2.1 Mathematical Formulation

We consider a stochastic optimal control problem in which an agent makes sequential decisions within a random environment over a finite time horizon, denoted as  $\mathbb{T} = \{0, 1, \dots, T\}$ . At each time  $t \in \mathbb{T}$ , the state of the environment is represented by  $\mathbf{s}_t \in \mathbb{S} \subseteq \mathbb{R}^m$ , and the agent selects an action  $\mathbf{a}_t \in \mathbb{A}_t \subseteq \mathbb{R}^n$ . The agent receives a reward  $r_t(\mathbf{s}_t, \mathbf{a}_t)$ , and the environment transitions to the next state  $\mathbf{s}_{t+1}$  according to the dynamics:

$$\mathbf{s}_{t+1} = f_t(\mathbf{s}_t, \mathbf{a}_t, \xi_{t+1}), \quad (1)$$

where  $\xi_{t+1}$  is a random variable taking values in  $\Xi \subseteq \mathbb{R}^d$ . The action set  $\mathbb{A}_t$  is characterized by a set of equality and inequality constraints:

$$\mathbb{A}_t = \{a \in \mathbb{R}^n : g_i(s_t, a) = 0, 1 \leq i \leq E; g_j(s_t, a) \geq 0, 1 \leq j \leq I\}, \quad (2)$$

with continuously differentiable constraint functions  $\{g_i\}_{i=1}^E$  and  $\{g_j\}_{j=1}^I$ .

The agent's objective is to maximize the expected total reward over the time horizon  $\mathbb{T}$ . Let  $\mathcal{F}_t$  denote the smallest  $\sigma$ -algebra generated by the random variables  $\xi_1, \dots, \xi_t$ . A policy  $\pi = (\pi_0, \dots, \pi_{T-1})$  is said to be *non-anticipative* if  $\pi_t$  is  $\mathcal{F}_t$ -measurable, meaning it depends only on the information available up to time  $t$ . Let  $\Pi$  be the collection of all non-anticipative policies. The agent solves the following optimization problem:

$$V^*(\mathbf{s}) = \max_{\pi \in \Pi} V^\pi(\mathbf{s}) = \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \pi_t) + R(\mathbf{s}_T) \middle| \mathbf{s}_0 = \mathbf{s} \right], \quad (3)$$

where  $R(\cdot)$  denotes the terminal reward function. For simplicity, we assume that the random variables  $\{\xi_t\}_{t=1}^T$  are independent. Under this probabilistic structure, the optimal policy  $\pi^*$  is in *feedback* form; that is,  $\pi_t^*$  is a function of the current state. In addition, let  $V_t^*(s)$  represent the optimal expected reward achievable from time  $t$  onward, given the current state  $s$ . Assume in this paper that

**Assumption 2.1.** All the functions  $r_t(\mathbf{s}, \mathbf{a})$ ,  $R(\mathbf{s})$ , and  $f_t(\mathbf{s}, \mathbf{a}, \xi)$  are all bounded and Lipschitz in the sense that there exists a sufficiently large constant  $B$  such that, for any  $\mathbf{s}, \mathbf{s}' \in \mathbb{R}^m$  and  $\mathbf{a}, \mathbf{a}' \in \mathbb{R}^n$ ,

$$|r_t(\mathbf{s}, \mathbf{a}) - r_t(\mathbf{s}', \mathbf{a}')| \leq B(\|\mathbf{s} - \mathbf{s}'\| + \|\mathbf{a} - \mathbf{a}'\|), \quad |R(\mathbf{s}) - R(\mathbf{s}')| \leq B\|\mathbf{s} - \mathbf{s}'\|,$$

$$\text{and } \sup_{\xi \in \Xi} \|f_t(\mathbf{s}, \mathbf{a}, \xi) - f_t(\mathbf{s}', \mathbf{a}', \xi)\| \leq B(\|\mathbf{s} - \mathbf{s}'\| + \|\mathbf{a} - \mathbf{a}'\|).$$

## 2.2 Information Relaxation and duality

This paper applies the concept of *information relaxation (IR) based duality* from the literature of stochastic dynamic programming to develop an adversarial reinforcement learning (RL) approach for stochastic optimal control problems. The IR framework relaxes the non-anticipativity constraints, allowing policies to utilize future information, while imposing penalties to account for violations of these constraints. For foundational work on this approach, see (Rogers 2007), (Brown et al. 2010), (Brown and Smith 2022), and (Chen et al. 2024).

Let  $\mathbb{A}^T := \mathbb{A}_0 \times \dots \times \mathbb{A}_{T-1}$  and  $\Xi^T := \Xi_1 \times \dots \times \Xi_T$ . We define a function  $z(\mathbf{a}, \xi) : \mathbb{A}^T \times \Xi^T \rightarrow \mathbb{R}$  as a *penalty*, where  $\mathbf{a} = (\mathbf{a}_0, \dots, \mathbf{a}_{T-1})$  is a sequence of actions, and  $\xi = (\xi_1, \dots, \xi_T)$  is a sequence of random perturbations. A penalty function is considered *feasible* if it satisfies  $\mathbb{E}[z(\pi, \xi)] \leq 0$  for all non-anticipative policies  $\pi \in \Pi$ . Given a feasible penalty function  $z(\cdot, \cdot)$ , the corresponding dual value is defined as:

$$D^z(\mathbf{s}) := \mathbb{E} \left[ \max_{\mathbf{a} = \{\mathbf{a}_t\}_{t=0}^{T-1} \in \mathbb{A}^T} \left( \sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t) + R(\mathbf{s}_T) - z(\mathbf{a}, \xi) \right) \middle| \mathbf{s}_0 = \mathbf{s} \right]. \quad (4)$$

In this formulation, the optimization is placed inside the expectation, thus relaxing the non-anticipativity constraints. The penalty function  $z$  serves to counterbalance the advantage gained from this relaxation.

The IR literature points out that penalties can be explicitly constructed using a sequence of functions  $W = (W_0, \dots, W_T)$ , where  $W_t(\cdot) : \mathbb{S} \rightarrow \mathbb{R}$ . For  $(\mathbf{a}, \xi) \in \mathbb{A}^T \times \Xi^T$  and  $t \in \mathbb{T}$ , define

$$z_t^W(\mathbf{a}, \xi) := W_{t+1}(\mathbf{s}_{t+1}) - \mathbb{E} [W_{t+1}(f_t(\mathbf{s}_t, \mathbf{a}_t, \eta_{t+1})) \mid \mathbf{s}_t], \quad (5)$$

where  $\{\mathbf{s}_t : t \in \mathbb{T}\}$  is the state trajectory by substituting  $\mathbf{a}$  and  $\xi$  into the system dynamics (1), and  $\eta_t$  shares the same distribution as  $\xi_t$ . By summing over all time steps, we can obtain a feasible penalty as follows:

$$z^W(\mathbf{a}, \xi) = \sum_{t=0}^{T-1} z_t^W(\mathbf{a}, \xi). \quad (6)$$

Hereafter we refer to  $W$  as the *generating function* of IR penalties.

The strong duality is achievable for finite-horizon, discrete-time control problems. Theorem 1 of (Rogers 2007) and Theorem 2.3 of (Brown et al. 2010) show that

$$\min_W \mathbb{E} \left[ \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^T \in \mathbb{A}^T} \left( \sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t) + R(\mathbf{s}_T) - z^W(\mathbf{a}, \xi) \right) \middle| \mathbf{s}_0 = \mathbf{s} \right] = V^*(\mathbf{s}). \quad (7)$$

In (7), the minimum on the left-hand side is attained when  $W = (V_0^*, \dots, V_T^*)$ , and the optimal policy  $\pi^*$  for the primal problem is also optimal for the dual problem. The strong duality relationship (7) presents a systematic approach to assess the optimality of a policy  $\pi$ . First, we may evaluate the policy by calculating

$$\underline{V}_t(\mathbf{s}) = \mathbb{E} \left[ \sum_{s=t}^{T-1} r_s(\mathbf{s}_s, \pi_s) + R_T(\mathbf{s}_T) \middle| \mathbf{s}_t = \mathbf{s} \right] \quad (8)$$

for all  $0 \leq t \leq T$ . Surely  $\underline{V}_t(\mathbf{s}) \leq V_t(\mathbf{s})$  for any  $\mathbf{s}$  because of the sub-optimality of  $\pi$ . Then, we replace the generic  $W$  in (6) by  $\underline{V}_t$  to construct a penalty  $z^{\underline{V}}$  and compute the associated dual value

$$\bar{V}_0(\mathbf{s}) = \mathbb{E} \left[ \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^T \in \mathbb{A}^T} \left( \sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t) + R(\mathbf{s}_T) - z^{\underline{V}}(\mathbf{a}, \xi) \right) \middle| \mathbf{s}_0 = \mathbf{s} \right] \quad (9)$$

From the strong duality (7), we know  $V_0(\mathbf{s}) \leq \bar{V}_0(\mathbf{s})$ , which implies  $0 \leq \bar{V}_0(\mathbf{s}) - V_0(\mathbf{s}) \leq \bar{V}_0(\mathbf{s}) - \underline{V}_0(\mathbf{s})$ . When the dual gap  $\bar{V}_0 - \underline{V}_0$  is sufficiently tight, we can conclude that the performance of policy  $\pi$  must be very close to the optimality. This line of work has sparked a rapidly expanding literature that leverages information-relaxation duality to certify the near-optimality of a wide range of heuristic policies. More recently, (Chen et al. 2024) develop a supersolution-based approach to improve the optimality for a given policy if we find that its duality gap is loose.

### 3 MAIN RESULTS

#### 3.1 Adversarial Deep Reinforcement Learning (ADRL)

The min-max formulation of (7) provides a game-theoretic perspective on strong duality. Consider an adversary whose goal is to minimize the agent's expected (penalized) reward by selecting an appropriate  $W$  (thereby penalty  $z^W$ ). Meanwhile, the agent seeks to maximize its rewards by employing anticipative policies. From this perspective, the interaction between the agent and the adversary can be framed as a zero-sum game. Based on this observation, we develop an adversarial RL algorithm in this section to learn the solution to (3).

Note that the generating function  $W$  of optimal penalties for (7) is typically a highly complex function of state  $\mathbf{s}$ , especially in many high-dimensional stochastic control problems. To address this challenge, we leverage the remarkable expressivity of deep neural networks (DNNs) to approximate the adversary's optimal choice for  $W$  in this work. In particular, let  $\phi_t \in \Phi_t \subset \mathbb{R}^s$  represent the hyperparameters of the network at time  $t$ , and we parameterize the DNNs as  $\{\rho_t(\cdot, \phi_t) : t \in \mathbb{T}\}$ . One may refer to Appendix A.1 for a brief description on the structure of DNN considered in this paper. Invoking (7), we train optimal  $\phi_t$  by solving the following minimization problem

$$\phi^* = \arg \min_{\phi=\{\phi_t : t \in \mathbb{T}\}} \mathbb{E} \left[ \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^T \in \mathbb{A}^T} \left( \sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t) + R(\mathbf{s}_T) - z^\phi(\mathbf{a}, \xi) \right) \middle| \mathbf{s}_0 = \mathbf{s} \right], \quad (10)$$

where  $z^\phi$  is a penalty based on the generating function  $\{\rho_t(\cdot, \phi_t) : t \in \mathbb{T}\}$ .

Structurally, the problem (10) is essentially a stochastic optimization problem. For any hyperparameter  $\phi$  in the construction of penalty function, action sequence  $\mathbf{a}$ , and noise sequence  $\xi$ , denote

$$Y(\phi, \mathbf{a}, \xi) := \sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t) + R(\mathbf{s}_T) - z^\phi(\mathbf{a}(\xi), \xi). \quad (11)$$

Here,  $\mathbf{s} = \{\mathbf{s}_t : t \in \mathbb{T}\}$  is the state sequence determined by the action and randomness pair  $(\mathbf{a}, \xi)$ . Using this notation, the optimization problem (10) is to maximize the expected value of  $\mathbb{E}[\max_{\mathbf{a}} Y(\phi, \mathbf{a}, \xi)]$ .

Inspired by this observation, we develop the following simulation based approach to solve it. We generate  $n$  sample paths of  $\xi$  and collect them into a training dataset  $\mathcal{L}^n := \{\xi^{(1)}, \dots, \xi^{(n)}\}$ , where each  $\xi^{(i)} = (\xi_1^{(i)}, \dots, \xi_T^{(i)})$ . Divide  $\mathcal{L}^n$  into  $m$  disjoint batches:  $\mathcal{L}^n = \mathcal{K}_1 \cup \dots \cup \mathcal{K}_m$ . For instance, we break sample paths evenly into  $m$  batches:  $\mathcal{K}_1 = \{\xi^{(1)}, \dots, \xi^{(l)}\}$ ,  $\mathcal{K}_2 = \{\xi^{(l+1)}, \dots, \xi^{(2l)}\}$ ,  $\dots$ ,  $\mathcal{K}_m = \{\xi^{((m-1)l+1)}, \dots, \xi^{(n)}\}$ , where  $l = n/m$ . We use one batch in each iteration of training. Suppose that the first  $b-1$  batches  $\mathcal{K}_1, \dots, \mathcal{K}_{b-1}$  have been used in the previous training iterations to yield the current estimate of the hyperparameter  $\{\phi_t^{b-1} : t \in \mathbb{T}\}$ . The  $b$ th iteration of ADRL consists of two stages:

**-Action Stage:** Construct the penalty  $z^{b-1}$  using  $\{\rho_t(\cdot, \phi_t^{b-1})\}$ . The agent solves the inner optimization problem

$$\mathbf{a}^b(\xi) = (\mathbf{a}_0^b(\xi), \dots, \mathbf{a}_{T-1}^b(\xi)) = \arg \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^{T-1} \in \mathbb{A}^T} Y(\phi^{b-1}, \mathbf{a}, \xi) \quad (12)$$

along each  $\xi \in \mathcal{K}_b$ . Therefore, the optimization problem in (12) is deterministic. There is a vast research literature about deterministic optimization that we can draw on to solve these problems; see, e.g., (Nocedal and Wright 2006). Let us assume for the inner optimization problem, we have

**Assumption 3.1.** (a) The constraint set  $\mathbb{A}_t$  is compact and convex for all  $t \in \mathbb{T}$ . In addition, it satisfies the Slater's condition.

(b) For any given  $\phi \in \Phi$ , the program (12) admits a unique solution for almost every  $\xi$ .

**-Adversarial Stage:** Update  $\phi$  using an estimate  $\text{grad}_\phi^{b-1}$  of the following gradient:

$$\nabla_\phi \mathbb{E} \left[ \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^{T-1}} Y(\phi^{b-1}, \mathbf{a}, \xi) \mid \mathbf{s}_0 = \mathbf{s} \right]. \quad (13)$$

Update  $\phi$  by:  $\phi^b = \phi^{b-1} - \gamma_b \cdot \text{grad}_\phi^{b-1}$ , where  $\gamma_b$  is the learning rate. The gradient  $\text{grad}_\phi^{b-1}$  can be estimated via Robbins-Monro or Kiefer-Wolfowitz methods, detailed in the following subsections.

---

**Algorithm 1** Adversarial Deep Reinforcement Learning (ADRL)

---

**Require:** Training data  $\mathcal{L}^n$ , network architecture  $\{\rho_t(\cdot, \phi_t)\}$ , initial parameters  $\{\phi_t^0\}$ , learning rates  $\{\gamma_b\}$

- 1: **for**  $b = 1, 2, \dots, m$  **do**
- 2:     Construct penalty  $z^{b-1}$  using  $\{\rho_t(\cdot, \phi_t^{b-1})\}$
- 3:     Select minibatch  $\mathcal{K}_b$
- 4:     **for** each  $\xi \in \mathcal{K}_b$  **do**
- 5:         Solve (12) to find  $\mathbf{a}^b(\xi)$
- 6:     **end for**
- 7:     Compute gradient  $\text{grad}_\phi^{b-1}$
- 8:     Update  $\phi^b = \phi^{b-1} - \gamma_b \cdot \text{grad}_\phi^{b-1}$
- 9: **end for**

---

We encapsulate the ADRL algorithm in Algorithm 1.

The ADRL algorithm outputs generating functions  $\{\rho_t(\cdot, \phi_t^m), t \in \mathbb{T}\}$ . This set of functions can be used to define a greedy policy through the following one-step Bellman equation: for all state  $\mathbf{s}$  at  $t \in \mathbb{T}$ , we solve

$$\begin{aligned}\pi_t^m(\mathbf{s}) &= \arg \max_{\mathbf{a}_t \in \mathbb{A}_t} \left( r_t(\mathbf{s}, \mathbf{a}_t) + \mathbb{E} [\rho_{t+1}(\mathbf{s}_{t+1}, \phi_{t+1}^m) | \mathbf{s}_t = \mathbf{s}] \right) \\ &= \arg \max_{\mathbf{a}_t \in \mathbb{A}_t} \left( r_t(\mathbf{s}, \mathbf{a}_t) + \mathbb{E} [\rho_{t+1}(f_t(\mathbf{s}_t, \mathbf{a}_t, \xi_{t+1}), \phi_{t+1}^m) | \mathbf{s}_t = \mathbf{s}] \right).\end{aligned}\quad (14)$$

Note that such calculated  $\pi^m$  depends only on the current state  $\mathbf{s}$ . Hence, it is an non-anticipative policy. In most practical applications, computing the policy for all possible states in the state space is computationally expensive, and analytical expressions for the expectations are often unavailable. To address this, we calculate  $\pi^m$  for a representative subset of states and generalize it to the entire state space using least-squares fitting. In this process, a sample average based on a small-scale simulation of  $\xi$  can be used as an approximation for the expectation, facilitating computation; see (Bertsekas and Tsitsiklis 1997) for further details. When the training data is sufficient, we expect that the generating function  $\{\rho_t(\cdot, \phi_t^m), t \in \mathbb{T}\}$  learned from ADRL is close to the optimal one; that is,  $\rho_t(\cdot, \phi_t^m) \approx V_t^*(\cdot)$ . By the strong duality relation (7), we know that the greedy policy  $\pi^m$  defined in (14) is also close to the optimal one.

(Balseiro and Brown 2019) develop a general framework in which they derive both a policy and a performance bound from the same approximation value function. On the one hand, they treat this approximation as the continuation value and apply the greedy method to select actions with respect to it. On the other hand, they generate a dual feasible penalty, thereby the dual bounds, using the same approximate value. “Good” approximations are thus crucial for this framework to yield tight dual gaps. The ADRL algorithm presents a systematic way to construct such “good” value approximations from the dual side of the problem. The penalty  $z^m$  constructed from the generating function  $\{\rho_t(\cdot, \phi_t^m), t \in \mathbb{T}\}$  should be close to the optimal penalty. Meanwhile, the greedy policy  $\pi^m$  should also be nearly optimal, leading to a tight dual gap.

### 3.1.1 The Robbins-Monro Method

Under some regularity conditions, we can develop unbiased estimators for the gradient (13). Suppose that the solution to (12) exists at  $\mathbf{a}^b$  and interchanging expectation and differentiation in (15) is feasible. Then,

$$\nabla_\phi \mathbb{E} \left[ \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^{T-1}} Y(\phi^{b-1}, \mathbf{a}, \xi) \middle| \mathbf{s}_0 = \mathbf{s} \right] = \mathbb{E} \left[ \nabla_\phi \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^{T-1}} Y(\phi^{b-1}, \mathbf{a}(\xi), \xi) \middle| \mathbf{s}_0 = \mathbf{s} \right]. \quad (15)$$

Note that the differentiation operation inside the expectation on the right-hand side of the above equality is taken under fixed  $\xi$ . This pathwise derivative estimation is well-studied in simulation literature; see (Glasserman 2004), (Asmussen and Glynn 2007), and (Chau and Fu 2015) and the reference therein. Hence,

$$\nabla_\phi \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^{T-1}} Y(\phi^{b-1}, \mathbf{a}, \xi) = \nabla_\phi \left( \sum_{t=0}^{T-1} r_t(\mathbf{s}_t, \mathbf{a}_t^b) + R(\mathbf{s}_T) - z^{b-1}(\mathbf{a}^b(\xi), \xi) \right) = \nabla_\phi Y(\phi^{b-1}, \mathbf{a}^b(\xi), \xi) \quad (16)$$

provides an unbiased estimator for the gradient on the left-hand side of (15). Fixed  $\xi$ . Changes in  $\phi$  affect  $Y(\phi^{b-1}, \mathbf{a}(\xi), \xi)$  directly through  $z^{\phi^{b-1}}$  and indirectly through the optimal solution  $\mathbf{a}^{\phi^{b-1}}$  to the inner optimization problem (12). However, under certain conditions, the indirect effect is negligible due to the envelope theorem (Milgrom and Segal 2002). Using this intuition, we can show that

**Proposition 3.2.** Denote

$$\nabla_\phi z_t^\phi(\mathbf{a}_t^\phi(\xi), \xi) := \nabla_\phi \rho_{t+1}(\mathbf{s}_{t+1}(\mathbf{a}^\phi, \xi), \phi_{t+1}) - \mathbb{E} \left[ \nabla_\phi \rho_{t+1} \left( f_t \left( \mathbf{s}_t(\mathbf{a}^\phi, \xi), \mathbf{a}_t^\phi, \eta_{t+1} \right), \phi_{t+1} \right) \right] \quad (17)$$

for  $t \in \mathbb{T}$ , where  $\nabla_\phi \rho_t(\cdot, \phi_t)$  represents the gradient of  $\rho_t$  with respect to  $\phi$  and  $\{\mathbf{s}_t(\mathbf{a}, \xi) : t \in \mathbb{T}\}$  is obtained by applying action  $\mathbf{a}$  and the randomness  $\xi$  to the system.

Then, under Assumptions 2.1 and 3.1

$$\nabla_{\phi} \mathbb{E} \left[ \max_{\mathbf{a}=\{\mathbf{a}_t\}_{t=0}^{T-1}} Y(\phi, \mathbf{a}, \xi) \mid \mathbf{s}_0 = \mathbf{s} \right] = \mathbb{E} \left[ - \sum_{t=0}^{T-1} \nabla_{\phi} z_t^{\phi}(\mathbf{a}_t^{\phi}(\xi), \xi) \right]. \quad (18)$$

Due to the page limit, we omit the complete proof in this paper and refer readers to the website of the first author (<https://www1.se.cuhk.edu.hk/~nchenweb/Publications.html>) for a proof sketch. According to Proposition 3.2, we construct an unbiased gradient estimator:

$$\text{grad}_{\phi}^{b-1} := - \frac{1}{|\mathcal{K}_b|} \sum_{\xi \in \mathcal{K}_b} \sum_{t=0}^{T-1} \nabla_{\phi} z_t^{b-1}(\mathbf{a}_t^b(\xi), \xi). \quad (19)$$

This estimator enables the use of the Robbins-Monro method to solve the dual problem. Unlike the Kiefer-Wolfowitz method, this approach does not require solving (12) for multiple  $\phi$  values.

### 3.1.2 The Kiefer-Wolfowitz (KW) Method

Alternatively, we can use finite difference-based estimates for  $\nabla_{\phi} \mathbb{E}[Y(\phi, \xi)]$ . Unlike the Robbins-Monro method, the KW algorithm does not require  $Y(\phi, \xi)$  to be differentiable. Consider the simultaneous perturbation stochastic approximation (SPSA) algorithm (Spall 1992). In the  $b$ th iteration, we generate a random direction  $\Delta^b = (\Delta_{t,j}^b : t \in \mathbb{T}, j \in \mathbb{J})$  and perturb  $\phi^{b-1}$  by  $c_b \Delta^b$ . The finite difference approximation is:

$$\hat{\nabla} Y(\phi^{b-1}, \xi) := \frac{\max_{\mathbf{a}} Y(\phi^{b-1} + c_b \Delta^b, \mathbf{a}, \xi) - \max_{\mathbf{a}} Y(\phi^{b-1} - c_b \Delta^b, \mathbf{a}, \xi)}{2c_b} \cdot (\Delta^b)^{-1}, \quad (20)$$

where  $(\Delta^b)^{-1}$  is the element-wise inverse of  $\Delta^b$ . This estimator requires only two function evaluations, regardless of the dimension of  $\phi$ . It significantly reducing computational cost compared with the traditional KW estimator (Kiefer and Wolfowitz 1952). The random directions  $\Delta^b$  are i.i.d. with  $\mathbb{E}[\Delta^b] = 0$  and finite inverse moments. A common choice is the symmetric Bernoulli distribution ( $\pm 1$  with probability  $1/2$ ). The step sizes  $\{c_b\}$  and  $\{\gamma_b\}$  must satisfy some regularity conditions such as  $c_b \rightarrow 0$ ,  $\sum_b \gamma_b c_b < \infty$ , and  $\sum_b \gamma_b^2 c_b^{-2} < \infty$  to ensure the convergence; see (Spall 1992), (Spall 2005), (Chau and Fu 2015).

## 4 NUMERICAL EXPERIMENTS

Consider an optimal order execution problem in which a trader plans to transact a large block of equity over a fixed time horizon with minimum impact costs. It can be viewed as a variant of the models proposed in (Bertsimas and Lo 1998), (Almgren and Chriss 2001), and (Haugh and Wang 2014). Assume that there are  $n$  different assets traded in the market, and the trader aims to acquire  $\bar{\mathbf{R}}$  shares in each of the assets in  $T$  periods. The objective of the trader is to determine a trading schedule, i.e., how many shares to purchase in each period, denoted by  $\mathbf{a}_t := \{a_1, \dots, a_n\}$ ,  $t = 1, 2, \dots, T$ , to minimize the associated transaction cost. Let  $\mathbf{R}_t \in \mathbb{R}^n$  denote the number of remaining orders need to satisfy in each asset at time  $t$ . Then, a feasible training scheme should satisfy

$$\sum_{t=1}^T \mathbf{a}_t = \bar{\mathbf{R}}, \quad \mathbf{a}_t \geq 0, \quad \mathbf{a}_t \in \mathbb{R}^n, \quad (21)$$

$$\mathbf{R}_{t+1} = \mathbf{R}_t - \mathbf{a}_t, \quad \mathbf{R}_1 = \bar{\mathbf{R}}, \quad \mathbf{R}_{T+1} = 0, \quad (22)$$

where  $\mathbf{a}_t \geq 0$  indicate the no-shorting constraint.

To complete the statement of the problem, we specify the price dynamics to be a linear price-impact model, following (Han and E 2016). In particular, assume that price  $\mathbf{P}_t \in \mathbb{R}^n$  follows

$$\mathbf{P}_t = \mathbf{P}_{t-1} + \mathbf{A}\mathbf{a}_t + \mathbf{B}\mathbf{X}_t + \boldsymbol{\varepsilon}_t, \text{ for all } t = 1, \dots, T \quad (23)$$

where  $\mathbf{A} \in \mathbb{R}^{n \times n}$  is a positive definite matrix and  $\mathbf{B} \in \mathbb{R}^{n \times m}$ . Here  $\{\boldsymbol{\varepsilon}_t, t = 1, \dots, T\}$  is a sequence of white noise with mean zero and covariance matrix  $\Sigma_{\varepsilon}$ . In (23), the constant matrix  $\mathbf{A}$  is used to capture the intensity of the permanent impact: trading the amount of  $\mathbf{a}_t$  changes in assets' fundamental values by  $\mathbf{A}\mathbf{a}_t$ , and this change will last persistently in the future via the iterative relation of  $\mathbf{P}$ .

In addition, this model allows trader to incorporate predictive signals to extract insights about the stock's future movements, thereby improving the performance of trade execution. The auxiliary process  $\mathbf{X}_t \in \mathbb{R}^m$  in (23) serves this purpose. There are several approaches in the literature for selecting such signals. For example, (Bertsimas and Lo 1998) propose that  $\mathbf{X}$  could represent the return of a broader market index, such as the S&P 500, a factor commonly employed in traditional asset pricing models like the CAPM. Alternatively,  $\mathbf{X}$  could be the outputs of an alpha model derived from the trader's private stock-specific analysis, capturing information that has not yet been fully reflected in market prices. The exact nature of  $\mathbf{X}$  is irrelevant in this experiment and thus we abstract its interpretation and assume it follows a stationary AR(1) process:  $\mathbf{X}_t = \mathbf{C}\mathbf{X}_{t-1} + \boldsymbol{\eta}_t$ . The random noise term  $\boldsymbol{\eta}_t$  has zero mean and covariance matrix  $\Sigma_{\eta}$ , independent of  $\boldsymbol{\varepsilon}_t$ . The trader's objective is to minimize

$$\min_{\{\mathbf{a}_t, 1 \leq t \leq T\}} \mathbb{E} \left[ \sum_{t=1}^T \mathbf{P}_t^{\text{tr}} \mathbf{a}_t \right]. \quad (24)$$

The state variable of this model can be chosen as  $(\mathbf{P}_{t-1}, \mathbf{X}_t, \mathbf{R}_t)$ . The control policy is a function of these state variables.

In general, this problem has no closed form solution; see the discussion in (Bertsimas and Lo 1998) about why this problem becomes intractable due to the non-negative constraints on  $\{\mathbf{a}_t\}$ . In our implementation, we set  $n = 10$  and  $m = 3$ , and  $T = 20$ , resulting in a high-dimensional space of control policies:  $\mathbb{R}^{23} \rightarrow \mathbb{R}^{10}$ . These challenging settings highlight the effectiveness of our ADRL algorithm in generating high-quality policies with performance guarantees. For the neural network architecture, we use two hidden layers, each with ReLU as the activation function. We change the number of neurons ( $k = 10, 50, 100$ ) in the experiments to test the approximation accuracy of neural networks of different complexity. During training procedure, we apply feature transformation to the state  $\mathbf{s}_t = (\mathbf{P}_{t-1}, \mathbf{X}_t, \mathbf{R}_t)$ , as this approach has been shown to accelerate convergence and enhance generalization. Specifically, we construct quadratic features as follows:  $(\mathbf{P}_{t-1}, \mathbf{X}_t, \mathbf{R}_t, \mathbf{X}_t \mathbf{X}_t^{\text{tr}}, \mathbf{R}_t \mathbf{R}_t^{\text{tr}}, \mathbf{P}_{t-1} \mathbf{R}_t^{\text{tr}}, \mathbf{X}_t \mathbf{R}_t^{\text{tr}}) \in \mathbb{R}^{263}$ .

These transformed features are then fed into the neural network for training. To compute the expectation in penalty construction (5), we employ a Monte Carlo simulation with a sample size of 2,000. In the Action Stage of the algorithm, we use the Sequential Least Squares Quadratic Programming (SLSQP) method to solve the optimization problem (12). In the Adversarial Stage, we apply the Adam optimizer to dynamically adjust the learning rates,  $\gamma_b$ , utilizing historical gradient information.

The first set of experiments focuses on the case without the no-shorting constraint,  $\mathbf{a} \geq 0$ . This scenario admits an explicit solution via dynamic programming; see (Bertsimas and Lo 1998) for the analytical expression of the optimal execution strategy and the corresponding optimal cost. For reference, the dotted line in Figure 1 represents the value of the optimal cost. The horizontal axis in Figure 1 shows the number of iterations.

The blue line displays the dual values with the penalty constructed on the generating function  $\{\rho_t(\cdot, \phi_t^m), t \in \mathbb{T}\}$  after each iteration. The shaded area around the blue line indicates the 95% confidence interval for these dual estimates. In each iteration, a single sample trajectory of  $\xi$  is generated as a mini-batch.

Theoretically, based on the strong duality relation (7), the obtained dual values should provide lower bounds for the true optimal value. This is supported by the observation that the blue line remains below

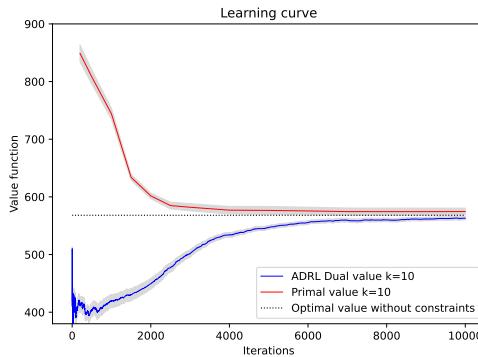


Figure 1: ADRL’s lower bound (blue) and policy-based upper bound (red) with 95% confidence intervals (shaded area). All networks trained for  $10^4$  iterations on an NVIDIA RTX 4090 GPU (sustained 10% utilization). The number of neurons  $k = 10$  in this figure. The total running times is 191 mins.

the dashed line in the figure. Furthermore, Figure 1 demonstrates that the dual values generated by our ADRL algorithm converge to the optimal value of the problem (24). As discussed in Section 3.1, we can construct non-anticipative policies using the greedy policy defined on the dual values. Specifically, we take the dual value functions after every 500 iterations and simulate (14) based on 2,000 samples of  $\xi$  to evaluate this greedy policy. The red line in Figure 1 shows the values associated with these policies. They are all above the dotted line because any policies are suboptimal. However, we can easily see that the greedy policy obtained after 4,000 iterations of ADRL is sufficiently optimal because the policy value is very close to the lower bound provided by dual values. To quantify the policy optimality, define

$$\text{gap} := \frac{\text{dual\_value} - \text{primal\_value}}{\text{dual\_value}}.$$

At 5,000 iterations, the gap reaches 2.82%. Since the true value lies between the dual and primal values (by strong duality), the relative error of the greedy policy at 5,000 iterations must be less than 2.82%. As a result, our ADRL algorithm can generate a good policy and meanwhile provide the corresponding dual values as performance guarantee, which presents a systematic way to construct the framework in (Balseiro and Brown 2019). The results for more complex network configurations ( $k=50$  and  $k=100$ ) exhibit trends similar to those shown in Figure 1. Therefore, we omit the corresponding figures for brevity.

In the second set of experiments, we impose no-shorting constraints on  $\mathbf{a}$  (i.e.,  $\mathbf{a} \geq 0$ ). As noted in (Bertsimas and Lo 1998), no closed-form solution exists for this case. We continue to examine a ten-asset portfolio setting ( $n = 10$ ,  $m = 3$ ,  $T = 20$ ), consistent with the first experiment. Figures 2a and 2b show that we can still construct effective confidence interval estimates for the problem. After 6,000 iterations, the dual gaps average 1.85% for  $k = 50$  and 2.86% for  $k = 100$ . Based on these tight gaps, we conclude with high confidence that the true value of the control problem in (24) with no-shorting constraints is likely to lie between 607.47 (dual value) and 621.08 (primal value). This value exceeds that of the case where short selling is allowed, which aligns with our expectation that smaller feasible sets lead to higher objective values for minimization problems.

Finally, we apply a popular reinforcement learning (RL) algorithm from the literature—the deep empirical risk minimization (DERM) algorithm (Han and E 2016)—to the problem with the no-shorting constraint to demonstrate the advantages of working with ADRL. As summarized in A.2, DERM uses in-sample trajectories of the noise process to train policy networks. For this purpose, we simulate 256 samples of  $\{(\varepsilon_t, \eta_t) : 0 \leq t \leq T\}$  and construct policy networks with two hidden layers, each containing 256 neurons. In each iteration, the samples are used to compute the gradient of the empirical loss on the right-hand side of (26) with respect to the hyperparameter  $\theta$  for updating the policy networks.

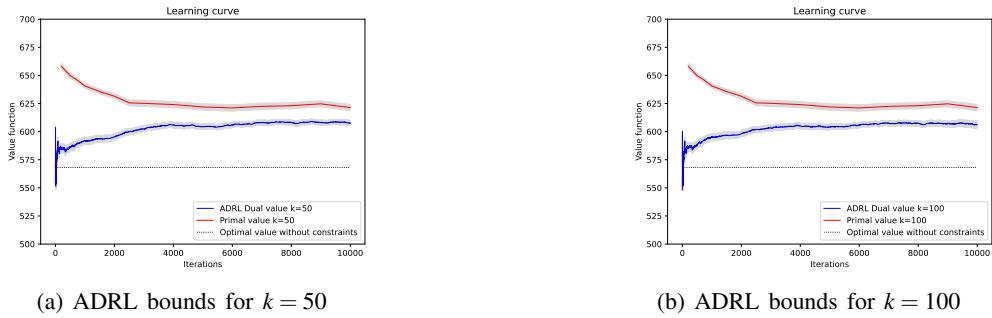


Figure 2: ADRL bounds ( $k = 50, 100$ ) showing learning curve and policy-based upper bound with 95% confidence intervals. The dotted line indicates the unconstrained theoretical optimal value.

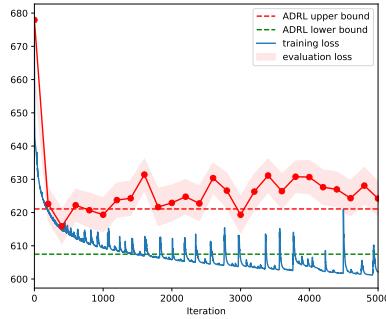


Figure 3: Training and evaluation loss of the ERM-based policy. The blue line shows in-sample training loss; The red dots with shaded bands show the out-of-sample evaluation of the learned policy with 95% confidence intervals. The two dashed horizontal lines indicate ADRL upper (red) and lower (green) bounds from Figure 2.

The blue line in Figure 3 shows the training loss, which continues to decrease throughout training. However, the evaluation loss (red line) initially improves but eventually worsens, indicating over training, see (Reppen and Soner 2023). During the initial stages of training, the loss decreases sharply, indicating that the algorithm quickly learns effective policies from the provided data. However, after 2,000 iterations, the algorithm begins to overfit, as evidenced by the training loss dropping below the lower bound provided by ADRL. In other words, the in-sample performance of the trained network surpasses that of the optimal non-anticipative policies. Note that the minimizer  $\theta^*$  for (26) depends on the entire trajectory  $s_t$ . As a result, DERM transitions from learning better non-anticipative policies to effectively predicting future states in the training data. Naturally, such an overfitted policy cannot generalize well to out-of-sample scenarios. As shown by the red dots in the figure, the out-of-sample value of the policies from the latter stage of learning performs worse (higher than the upper bound provided by ADRL).

A critical problem to resolve this trade off is when to stop the learning. ADRL provides us a systematic approach to address this issue. For instance, when we stop in-sample learning if the training loss lies in between the lower and upper bounds from ADRL, the strong duality relation implies that in-sample performance of the learned policy must be close to the optimality. Figure 3 illustrates that such stopping rule also leads to good out-of-sample performance. However, we need to acknowledge that, in terms of computational efficiency, the primal method significantly outperforms ADRL, completing 5,000 training iterations in approximately 7 minutes. This advantage is largely attributed to high degree of parallelism. In contrast, ADRL involves repeatedly solving inner optimization problems. How to combine the strength of both algorithms to enhance the scalability of ADRL will be left for future investigation.

## 5 CONCLUSION

In this study, we develop a novel approach to high-dimensional stochastic control problems by integrating the information relaxation technique with neural network approximations. Our ADRL method simultaneously generates upper and lower bounds for the control problem. Future work will further investigate ADRL's mechanism for mitigating overfitting through adversarial penalties and extend the framework to model-free settings.

## ACKNOWLEDGMENTS

This research project is partially supported by General Research Fund Scheme (GRF) of Hong Kong Research Grant Council (Project No. 14211023 and 14205824).

## A APPENDIX

### A.1 Neural Networks Architecture

In this paper, we employ feedforward neural networks with  $N \geq 2$  hidden layers (each containing  $k$  neurons) to approximate control policies and value functions. We specify the detailed architecture in this appendix. The network architecture consists of one input layer, one output layer, and  $N$  hidden layers. The state vector  $\mathbf{s}$  is encoded into the network through the input layer. More precisely, we can represent it as follows:

$$F(\mathbf{s}; \phi) = h_o \circ \sigma_N \circ h_N \circ \cdots \circ \sigma_1 \circ h_1(\mathbf{s}), \quad (25)$$

where  $h_1 : \mathbb{R}^m \rightarrow \mathbb{R}^k$ ,  $h_n : \mathbb{R}^k \rightarrow \mathbb{R}^k$  ( $2 \leq n \leq N$ ), and  $h_o : \mathbb{R}^k \rightarrow \mathbb{R}^d$  are all affine functions. In particular,  $h_1(\mathbf{x}) = \mathbf{H}^1 \mathbf{x} + \mathbf{b}^1$ ,  $h_n(\mathbf{x}) = \mathbf{H}^n \mathbf{x} + \mathbf{b}^n$ ,  $h_o(\mathbf{x}) = \mathbf{H}^o \mathbf{x} + \mathbf{b}^o$ , where  $\mathbf{H}^1 \in \mathbb{R}^{k \times m}$  (a  $k \times m$  matrix),  $\mathbf{H}^n \in \mathbb{R}^{k \times k}$  (a  $k \times k$  matrix),  $\mathbf{H}^o \in \mathbb{R}^{d \times k}$  (a  $d \times k$  matrix), and  $\mathbf{b}^1, \mathbf{b}^n \in \mathbb{R}^k$ , for  $n = 1, \dots, N$ ,  $\mathbf{b}^o \in \mathbb{R}^d$ .  $\sigma_n : \mathbb{R}^k \rightarrow \mathbb{R}^k$  is a component-wise activation function given by  $\sigma_n(x_1, \dots, x_k) = (\sigma(x_1), \dots, \sigma(x_k))$  for each  $n = 1, \dots, N$ . The hyperparameter  $\phi$  in the body text of the paper refers to the matrices  $\{\mathbf{H}^1, \dots, \mathbf{H}^N, \mathbf{H}^o\}$  and  $\{\mathbf{b}^1, \dots, \mathbf{b}^N, \mathbf{b}^o\}$  used in the construction of network (25).

### A.2 Deep Empirical Risk Minimization Algorithm

(Han and E 2016) developed a deep learning-based approach to directly solves the primal control problem. Its key idea is to approximate the functional dependence of optimal control  $\pi^*$  on the state by multilayer feedforward neural networks  $\varpi_t(\cdot, \theta_t)$ , where  $\theta_t$  is the corresponding network hyperparameter.

To learn the optimal feedback controls, we first repeatedly simulate  $n$  i.i.d. replica of the trajectory of  $\xi$ :  $\mathcal{L}^n := \{\xi^{(1)}, \dots, \xi^{(n)}\}$ , with  $\xi^{(i)} = (\xi_1^{(i)}, \dots, \xi_T^{(i)})$  for  $1 \leq i \leq n$ . Then, we may find the optimal hyperparameters through minimizing the following empirical risk

$$v^*(\mathcal{L}^n) := \max_{\{\theta_t\}_{t=0}^{T-1}} \frac{1}{n} \sum_{i=1}^n \left[ \sum_{t=0}^{T-1} r(\mathbf{s}_t^{(i)}, \varpi_t(\mathbf{s}_t^{(i)}, \theta_t)) + R(\mathbf{s}_T^{(i)}) \right]. \quad (26)$$

The standard stochastic gradient descent method with backpropagation can be easily adapted for this purpose. After the in-sample training, we can apply the trained policy network  $\varpi(\cdot, \theta_t^*)$  in out of sample to control.

## REFERENCES

Almgren, R., and N. Chriss. 2001. "Optimal Execution of Portfolio Transactions". *Journal of Risk* 3:5–40.  
 Asmussen, S., and P. W. Glynn. 2007. *Stochastic Simulation: Algorithms and Analysis*, Volume 57. New York: Springer.  
 Balseiro, S. R., and D. B. Brown. 2019. "Approximations to Stochastic Dynamic Programs via Information Relaxation Duality". *Operations Research* 67(2):577–597.

Becker, S., P. Cheridito, and A. Jentzen. 2019. “Deep Optimal Stopping”. *Journal of Machine Learning Research* 20(1):2712–2736.

Bertsekas, D. P., and J. N. Tsitsiklis. 1997. *Neuro-Dynamic Programming*. Belmont, Massachusetts: Athena Scientific.

Bertsimas, D., and A. W. Lo. 1998. “Optimal Control of Execution Costs”. *Journal of Financial Markets* 1(1):1–50.

Brown, D. B., and J. E. Smith. 2022. “Information Relaxations and Duality in Stochastic Dynamic Programs: A Review and Tutorial”. *Foundations and Trends in Optimization* 5(3):246–339.

Brown, D. B., J. E. Smith, and P. Sun. 2010. “Information Relaxations and Duality in Stochastic Dynamic Programs”. *Operations Research* 58(4-part-1):785–801.

Buehler, H., L. Gonon, J. Teichmann, and B. Wood. 2019. “Deep Hedging”. *Quantitative Finance* 19(8):1271–1291.

Chau, M., and M. C. Fu. 2015. “An Overview of Stochastic Approximation”. In *Handbook of Simulation Optimization*, 149–178.

Chen, N., X. Ma, Y. Liu, and W. Yu. 2024. “Information Relaxation and a Duality-Driven Algorithm for Stochastic Dynamic Programs”. *Operations Research* 72(6):2302–2320.

Glasserman, P. 2004. *Monte Carlo Methods in Financial Engineering*, Volume 53. New York: Springer.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, et al. 2020. “Generative Adversarial Networks”. *Communications of the ACM* 63(11):139–144.

Goodfellow, I. J., J. Shlens, and C. Szegedy. 2015. “Explaining and Harnessing Adversarial Examples”. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Available at <https://arxiv.org/abs/1412.6572>.

Han, J., and W. E. 2016. “Deep Learning Approximation for Stochastic Control Problems”. *arXiv preprint arXiv:1611.07422*.

Han, J., A. Jentzen, and W. E. 2018. “Solving High-Dimensional Partial Differential Equations Using Deep Learning”. *Proceedings of the National Academy of Sciences* 115(34):8505–8510.

Haugh, M., and C. Wang. 2014. “Dynamic Portfolio Execution and Information Relaxations”. *SIAM Journal on Financial Mathematics* 5(1):316–359.

Huang, S., N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. 2017. “Adversarial Attacks on Neural Network Policies”. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track*. Available at <https://openreview.net/pdf?id=ryv1RyBKI>.

Huré, C., H. Pham, A. Bachouch, and N. Langrené. 2021. “Deep Neural Networks Algorithms for Stochastic Control Problems on Finite Horizon: Convergence Analysis”. *SIAM Journal on Numerical Analysis* 59(1):525–557.

Kiefer, J., and J. Wolfowitz. 1952. “Stochastic Estimation of the Maximum of a Regression Function”. *The Annals of Mathematical Statistics*:462–466.

Milgrom, P., and I. Segal. 2002. “Envelope Theorems for Arbitrary Choice Sets”. *Econometrica* 70(2):583–601.

Nocedal, J., and S. J. Wright. 2006. *Numerical Optimization*. New York: Springer.

Reppen, A. M., and H. M. Soner. 2023. “Deep Empirical Risk Minimization in Finance: Looking into the Future”. *Mathematical Finance* 33(1):116–145.

Rogers, L. C. G. 2007. “Pathwise Stochastic Optimal Control”. *SIAM Journal on Control and Optimization* 46(3):1116–1132.

Spall, J. C. 1992. “Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation”. *IEEE Transactions on Automatic Control* 37(3):332–341.

Spall, J. C. 2005. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons.

## AUTHOR BIOGRAPHIES

**NAN CHEN** is a Professor in the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong (CUHK). His research interests include financial engineering and FinTech, with a focus on reinforcement learning, quantitative modeling in finance and risk management, Monte Carlo simulation, and applied probability. His email address is [nchen@se.cuhk.edu.hk](mailto:nchen@se.cuhk.edu.hk) and his website is <https://www1.se.cuhk.edu.hk/~nchenweb/>.

**MENGZHOU LIU** is a PhD in the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong (CUHK). His email address is [mzliu@link.cuhk.edu.hk](mailto:mzliu@link.cuhk.edu.hk).

**XIAOYAN WANG** is a postdoctoral researcher in the Centre for Financial Engineering at the Chinese University of Hong Kong (CUHK). Her email address is [wangxiaoyan235@link.cuhk.edu.hk](mailto:wangxiaoyan235@link.cuhk.edu.hk).

**NANYI ZHANG** is a PhD student in the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong (CUHK). His email address is [nyzhang@se.cuhk.edu.hk](mailto:nyzhang@se.cuhk.edu.hk).