

# Distributionally Robust Multi-Agent Reinforcement Learning for Dynamic Chute Mapping

Guangyi Liu<sup>1</sup>, Suzan Iloglu<sup>1</sup>, Michael Caldara<sup>1</sup>, Joseph W. Durham<sup>1</sup>, and Michael M. Zavlanos<sup>1,2</sup>

## Abstract

In Amazon robotic warehouses, the destination-to-chute mapping problem is crucial for efficient package sorting. Often, however, this problem is complicated by uncertain and dynamic package induction rates, which can lead to increased package recirculation. To tackle this challenge, we introduce a Distributionally Robust Multi-Agent Reinforcement Learning (DRMARL) framework that learns a destination-to-chute mapping policy that is resilient to adversarial variations in induction rates. Specifically, DRMARL relies on group distributionally robust optimization (DRO) to learn a policy that performs well not only on average but also on each individual subpopulation of induction rates within the group that capture, for example, different seasonality or operation modes of the system. This approach is then combined with a novel contextual bandit-based predictor of the worst-case induction distribution for each state-action pair, significantly reducing the cost of exploration and thereby increasing the learning efficiency and scalability of our framework. Extensive simulations demonstrate that DRMARL achieves robust chute mapping in the presence of varying induction distributions, reducing package recirculation by an average of 80% in the simulation scenario.

## 1 Introduction

In Amazon robotic sortation warehouses, mobile robots are deployed to transport and sort packages efficiently to different destinations [1, 2, 3, 4, 5]. The sorting process begins at induction stations, where packages are loaded onto mobile robots and subsequently transported to designated eject chutes based on their destinations (Figure 1). A critical factor determining the package throughput capacity of these facilities is the effective allocation of eject chutes to different destinations. Therefore, the destination-to-chute mapping policy plays a crucial role in optimizing the overall throughput performance of the robotic sortation warehouse.

Our previous work [6] addresses the destination assignment problem (DAP) [7] in robotic sorting systems by developing a dynamic chute mapping policy. This policy determines the optimal allocation of eject chutes to destinations with the objective of minimizing the number of unsorted packages. We proposed a model-free reinforcement learning approach that dynamically adjusts the number of chutes assigned to each destination throughout the day. Our solution formulates the chute mapping problem within a Multi-Agent Reinforcement Learning (MARL) framework [8, 9, 10, 11], where each destination is represented as an agent that controls its chute allocation at each time step.

While the MARL policy proposed in our previous work [6] demonstrates superior performance compared to traditional reactive chute mapping approaches often implemented in Amazon robotic sortation warehouses, its effectiveness assumes the induction distribution during deployment matches the training distribution and

---

<sup>1</sup>G. Liu, S. Iloglu, M. Caldara, J. W. Durham, and M. M. Zavlanos are with Amazon Robotics {gyliu, siloglu, caldaram, josepdur, miczavla}@amazon.com.

<sup>2</sup>M. M. Zavlanos is with Department of Mechanical Engineering and Materials Science, Duke University. {michael.zavlanos}@duke.edu.



Figure 1: Schematic layout of an Amazon robotic sortation warehouse featuring eject chutes.

the daily induction rate stays close to its average value. In practice, however, induction patterns exhibit significant temporal variations, potentially compromising the MARL policy’s performance when confronted with unexpected distribution changes.

To enhance robustness against such variations, in this paper, we propose a Distributionally Robust Multi-Agent Reinforcement Learning (DRMARL) framework that develops chute mapping policies capable of maintaining near-optimal performance across diverse induction distributions. Specifically, we formulate this problem as a group DRO problem, where each group represents a distinct induction distribution pattern extracted from a subset of a historical dataset. Our DRMARL framework optimizes policies for the worst-case induction patterns across these distribution groups. To address the computational cost of exhaustively evaluating all distribution groups during training, we introduce a contextual bandit (CB)-based worst-case reward predictor for each state-action pair. Through extensive evaluation, we demonstrate that our DRMARL framework yields robust chute mapping policies that not only outperform baseline MARL policies on out-of-distribution (OOD) induction data but also maintain consistent performance across varying induction distributions.

Our *contributions* are twofold: First, we introduce group distributionally robust optimization in multi-agent reinforcement learning, developing a novel framework to learn policies that are robust to distribution shifts. Second, we propose an innovative contextual bandit-based method for efficient worst-case reward prediction, significantly reducing the computational complexity of DRMARL training by eliminating the need for exhaustive group exploration to learn the worst-case reward. To the best of our knowledge, our framework is the first to integrate contextual bandits with group DRO and MARL, addressing a well-known challenge of distributionally robust reinforcement learning related to its computational cost. Our proposed framework has broad applicability to various large-scale industrial applications, beyond sortation systems, including resource allocation, collaborative robotics, and warehouse automation, where robustness to distribution shifts is crucial.

## 1.1 Literature Review

*Destination Assignment Problems:* Mathematical programming has been used to optimize warehouse systems, including destination assignment problems (DAP) [7] for sorting systems. The destination mapping approach in [12] optimizes package flow by minimizing travel distances between inbound and outbound stations in conveyor-based sorting systems, leading to improved throughput. [13] minimizes the worst-case flow imbalance across work stations on the sortation floor, developing a stochastic approach with chance and robust constraints. For robotic sorting systems, [14] proposes an integer programming method to solve DAPs that minimize sortation effort and satisfy package deadlines. A robust formulation addressing demand

uncertainty is presented in [15]. While these approaches effectively optimize destination assignment in sorting systems, they do not account for distributional uncertainty in demand and system dynamics. In contrast, our proposed DRMARL framework explicitly models such uncertainties, ensuring robust performance under varying operational conditions.

*MARL for Resource Allocation:* MARL has previously been applied to address resource allocation problems [16, 17, 18]. For example, a MARL framework for ocean transportation networks was proposed in [19]. This framework develops a multi-agent  $Q$ -learning algorithm where the local  $Q$ -networks depend on the joint states (including the limited shared resources) and the joint actions. However, since the joint state-action space grows exponentially with the number of agents, the local  $Q$ -networks are hard to learn and this approach does not scale well in practice. This limitation was addressed in our previous work [6], where the local  $Q$ -networks are only loosely coupled, enhancing the scalability while still being interconnected enough to capture the impact of robot congestion on the sortation floor. Compared to [19], the method proposed in [6] models resources explicitly as actions and considers budget constraints when taking joint actions. However, these MARL-based approaches do not incorporate distributional robustness, making them sensitive to demand fluctuations and uncertainty, which our DRMARL framework explicitly addresses to ensure reliable performance in dynamic sorting environments.

*Robust and Distributionally Robust RL:* Robust Reinforcement Learning (Robust RL) [20, 21, 22, 23, 24, 25] develops policies that maintain performance under worst-case conditions through adversarial perturbations. Distributionally Robust Reinforcement Learning (DRRL) [26, 27, 28, 29, 30, 31, 32] extends this by optimizing across environment distributions rather than single worst-case scenarios. While traditional DRRL primarily addresses ambiguity in MDP transition probabilities, this approach inadequately captures induction distribution changes in Amazon robotic sortation warehouses. Our problem requires focus on distributionally robust optimization of reward function distributions, building on [33, 34]. Recent advances in (Distributionally) Robust Multi-Agent RL [35, 36, 37, 38] have introduced frameworks like RMGs, ERNIE, and DRNVI to address environmental uncertainties, adversarial dynamics, and model uncertainties. While existing methods primarily focus on robustness in transition dynamics, adversarial interactions, and general environmental uncertainties, they do not explicitly address distributional shifts in package induction, which is a critical challenge in sortation warehouses. Our approach extends DRMARL to explicitly model and optimize against uncertainties in induction distributions, ensuring robust and consistent performance under varying operational conditions.

*Group DRO:* Group Distributionally Robust Optimization aims to enhance model robustness across diverse subpopulations by optimizing for the worst-performing groups rather than average performance [39]. This approach ensures fairness and resilience to distribution shifts, particularly for underrepresented groups. While initial work focused on single-agent supervised learning [40, 41], recent advances have extended these principles to more complex settings. Notably, [42] provided a soft-weighting method on distribution groups with convergence guarantees, while [43] and [44] demonstrated the applicability of group DRO in multi-agent systems and reinforcement learning, respectively. Our work bridges a critical gap by introducing group DRO principles to DRMARL. We begin by formulating the distributionally robust Bellman operator and addressing the computational challenges of exploring all distribution groups during the training. To tackle these challenges, we provide a DR Bellman operator specifically designed for MARL and introduce a contextual bandit (CB)-based worst-case distribution group predictor. This predictor enables efficient training by adaptively identifying the worst-case distribution groups.

The remainder of the paper is organized as follows. In Section 2, we formulate the dynamic chute mapping problem within a multi-agent reinforcement learning framework. In Section 3 we extend this formulation by incorporating group distributionally robust optimization into MARL, while in Section 4 we present our novel contextual bandit-based worst-case reward predictor to enhance training efficiency. Finally, in Section 5, we demonstrate the effectiveness of our proposed framework through extensive simulations. All

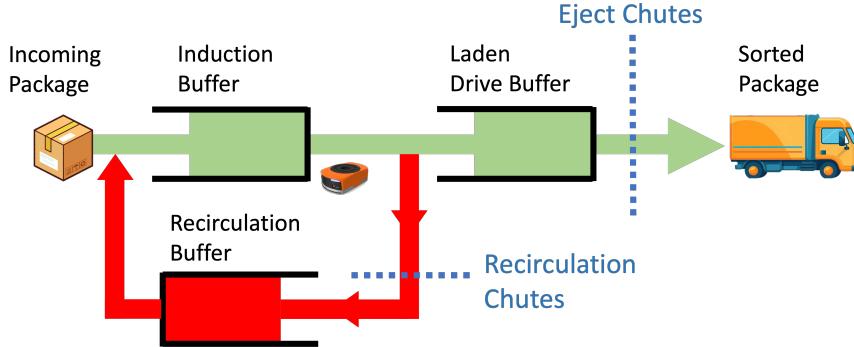


Figure 2: Flow of packages in the Amazon robotic sortation warehouse.

theoretical proofs are provided in Appendix A.

## 2 Problem Formulation

In Amazon robotic sortation warehouses, package flow is modeled using three buffers: induct, laden drive, and recirculation, as illustrated in Figure 2. The laden drive buffer represents packages currently being transported by robots to eject chutes. Due to the limited chute capacity, insufficient chute allocation for a destination can lead to sortation bottlenecks. When this occurs, excess packages from the laden drive buffer are redirected by robots to recirculation chutes, entering the recirculation buffer for reprocessing through the sortation system. Eject chutes can be reallocated under two conditions: when the destination vehicle reaches capacity or when the transportation schedule deadline is met. The dynamic chute mapping policy aims to optimize chute allocation across destinations, minimizing recirculation buffer volume while maximizing the system throughput.

### 2.1 MARL Structure

We formulate the dynamic chute mapping problem as a sequential decision making problem; specifically, a MARL problem that determines optimal chute allocations to minimize package recirculation at each time step. To do so, we define a Markov game over  $N$  agents (unique destinations) by the tuple  $(N, \mathcal{S}, \{\mathcal{O}^i\}_{i=1}^N, \{\mathcal{A}^i\}_{i=1}^N, P, \{r^i\}_{i=1}^N, \gamma, \rho_0)$ , where:

- (a) **Agents:**  $N$  agents, each corresponding to a unique destination.
- (b) **State Space:**  $\mathcal{S}$  denotes the joint state space.
- (c) **Observation Space:** For each agent  $i$ ,  $\mathcal{O}^i \subset \mathcal{S}$  represents its local observation at each time step, consisting of:
  - Number of packages recirculated until time  $t$  for agent  $i$
  - Total number of available chutes that can be assigned (uniform across all agents)
  - Number of chutes currently assigned to agent  $i$
- (d) **Action Space:** For each agent  $i$ ,  $\mathcal{A}^i$  represents its action space, determining the number of new chutes required. Actions take values in  $[0, 10]$ , where an action value  $a$  indicates the assignment of  $a$  new chutes to agent  $i$  at that time step. The joint action space is defined as  $\mathcal{A} = \prod_{i=1}^N \mathcal{A}^i$ .

- (e) **Transition Probability:**  $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{X} \rightarrow [0, 1]$  specifies the probability of packages being either sorted by chutes or sent to the recirculation buffer. In the large-scale Amazon robotic sortation warehouse setting, the transition probability is a function of the induction distribution  $\mathbb{P}$  with random variable  $X \in \mathcal{X}$ , which is addressed in detail in Appendix C.1.
- (f) **Reward Function:** For each agent  $i$ ,  $r^i : \mathcal{S} \times \mathcal{A} \times \mathcal{X} \rightarrow \mathbb{R}$  defines the reward function, penalizing both the number of allocated chutes and the number of packages in the recirculation buffer. The recirculation is a function of the induction distribution, which is defined in Section 2.2.

The model is completed with discount factor  $\gamma \in (0, 1)$  and initial state distribution  $\rho_0$ . To address the scalability of the state-action space and computational feasibility of the expectation in (3), we employ the Value Decomposition Network (VDN) [6] with budget constraints for computing joint actions. Implementation details are provided in Section 2.3 and Section 2.4.

The system operates in discrete time steps, where at each step  $t$ , individual agents corresponding to destinations make chute allocation decisions. Each agent  $i$  employs a local policy  $\pi^i : \mathcal{O}^i \times \mathcal{A}^i \rightarrow [0, 1]$ , which maps local observations  $o^i$  to probabilities over possible chute allocation actions. Each agent  $i$  learns its optimal local policy  $\pi^{i,*}$  by maximizing the expected discounted return  $\mathbb{E}[R_t^i] = \mathbb{E}\left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'}^i\right]$ , where  $r_{t'}^i$  denotes the instantaneous reward at time  $t'$ . The expectation accounts for both the stochastic nature of the policy and the environment dynamics. This formulation naturally aligns individual agent objectives with the global goal of minimizing recirculation while maintaining efficient sortation throughput. The instantaneous reward function for agent  $i$  at time step  $t$  is defined as:

$$r_t^i = -\text{recirc}_t^i - 2a_t^i, \quad (1)$$

where  $\text{recirc}_t^i \geq 0$  represents the number of packages in recirculation for destination  $i$ . Due to the coupled nature of agent decisions, we utilize the joint action-value function to determine optimal local policies  $\pi^{i,*}$ :

$$Q^\pi(s, a) = \mathbb{E}\left[\sum_{i=1}^N R_t^i | s_t = s, a_t = a\right], \quad (2)$$

which evaluates the expected return when taking joint action  $a = (a^1, \dots, a^N)$  in state  $s$  and following joint policy  $\pi$  thereafter.

To mitigate the exponentially growing policy space, we assume agents execute actions independently such that  $\pi = \prod_{i=1}^N \pi^i$ . The optimal policy  $\pi^*$  is learned using the Deep Q-Network (DQN) [45], where a neural network  $Q(s, a; \theta)$  with parameters  $\theta$  approximates the optimal action-value function  $Q^{\pi^*}(s, a)$ . The learning process minimizes the loss:

$$L(\theta; X) = \mathbb{E}_{s,a,r,s'} \left[ (Q(s, a; \theta) - y)^2 \right], \quad (3)$$

where  $y = \mathbb{E}_{X \sim \mathbb{P}}[r(s, a; X)] + \gamma \max_{a'} \bar{Q}(s', a'; \bar{\theta})$  approximates the optimal target values  $\mathbb{E}_{X \sim \mathbb{P}}[r(s, a; X)] + \gamma \max_{a'} Q^{\pi^*}(s', a')$ . Here,  $r(s, a; X)$  represents the instantaneous reward under the current state-action pair and induction pattern  $X$ .<sup>1</sup> Stability of the learning process is enhanced through two mechanisms: a target network  $\bar{Q}$  with periodic parameter updates using the most recent values of  $\theta$ , and the use of an experience replay buffer  $\mathcal{D}$  storing transition tuples  $(s, a, r, s')$ . The resulting optimal policy takes the form:

$$\pi^*(s, a) = \begin{cases} \frac{1}{|\mathcal{A}(s)|} & \text{if } a \in \mathcal{A}(s) \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

where  $\mathcal{A}(s) = \arg \max_a Q(s, a; \theta^*)$  and  $\theta^* = \arg \min L(\theta; X)$ .

---

<sup>1</sup>See Section 2.2 for detailed definition of induction pattern and induction generating distribution.

## 2.2 Induction Distribution

For a given sortation warehouse with  $D$  destinations and  $T$  time intervals (e.g., hours or minutes) during one day, we consider the random variable  $X = \{X_1, \dots, X_{DT}\} \in \mathbb{R}^{DT}$ , where  $X_i \geq 0$  represents the number of packages arriving at each destination-time pair  $(d, t)$  during the day. We model each day's induction as a random variable  $X$ , generated by an unknown probability distribution  $\mathbb{P}$ , which we refer to as the induction generating distribution. We assume the total daily package induction volume remains constant at  $V$  across all days, as the MARL chute mapping policy is primarily influenced by the distribution pattern rather than total volume variations.

**Definition 2.1.** (Induction Generating Distribution) Let us consider a multinomial distribution  $\mathcal{M}(n, p_1, \dots, p_k)$ , which obtains its support on

$$\left\{ (z_1, \dots, z_k) \mid \sum_{i=1}^k z_i = V, z_i \geq 0, \forall i = 1, \dots, n \right\}, \quad (5)$$

and the probability mass function is given by

$$\mathbb{P}(X_1 = z_1, \dots, X_k = z_k) = \frac{n!}{z_1! z_2! \dots z_k!} p_1^{z_1} p_2^{z_2} \dots p_k^{z_k}. \quad (6)$$

For a daily induction random variable  $X$  and induction patterns from temporally proximate dates (e.g., within the same week), we assume they follow the same induction generating distribution  $\mathcal{M}(n, p_1, \dots, p_{DT})$ . The empirical induction generating distribution is estimated using Sample Average Approximation (SAA) [46] from these temporally related dates.

In practice, we utilize induction data collected from Years 1-4, clustering it into 21 groups based on week numbers. For each group  $g$ , we construct an empirical induction generating distribution  $\mathbb{P}_g$  via SAA, modeled as a multinomial distribution using all induction data within that group. The group ambiguity set  $\mathfrak{M}$  in (9) is then formed using the collection of distributions  $\{\mathbb{P}_1, \mathbb{P}_2, \dots, \mathbb{P}_{21}\}$ .

The probability  $p_j$  of an incoming package being assigned to the  $j$ 'th destination-time pair is derived from the approximated empirical multinomial distribution. The distribution of  $V$  packages across destinations is then determined through sampling  $V$  times according to these probabilities.

## 2.3 Dimension Reduction of the State-Action Space

To manage the dimensionality of the state-action space, we decompose the joint  $Q$ -network into a sum of local  $Q$ -networks. Each local network captures the expected return of individual agents' chute mapping actions, while the joint network represents the expected return of the complete chute assignment across all agents. Specifically, we express the joint  $Q$ -network as:

$$Q(s, a, \theta) = \sum_{i=1}^N Q'(i, s^i, a^i; \theta), \quad (7)$$

where the input space scales linearly with the number of agents. While this decomposition is similar to [6], our approach learns a single shared  $Q'$  network for all agents instead of separate networks for each agent, resulting in improved computational efficiency.

## 2.4 Feasibility of Joint Actions

In unconstrained settings, agents would simply select actions that maximize their individual  $Q$ -networks, with the joint action being the collection of these individual choices. However, the chute mapping problem

introduces resource constraints, as agents must share a limited number of available chutes. This necessitates coordination to allocate resources optimally among agents based on their state-action values.

Given a budget constraint  $M$  on the joint actions such that  $\sum_{i=1}^N a_i \leq M$ , we formulate the following integer program to determine the optimal joint action that maximizes the joint  $Q$ -network for any state  $s$ :

$$\begin{aligned} & \underset{a^1, \dots, a^N}{\text{maximize}} \quad \sum_{i=1}^N Q'(i, s^i, a^i; \theta) \\ & \text{s.t. } \sum_{i=1}^N a^i \leq M, \quad a^i \in \mathbb{N} \end{aligned} \tag{8}$$

This integer program, which can be efficiently solved using commercial solvers such as Google-OR Tools [47] or Xpress [48], serves two purposes: it generates feasible data for the replay buffer to compute the expectation in (3) during training, and it determines the optimal actions once learning has converged. Notably, this optimization step is separate from the  $Q$ -learning process.

While the above MARL-based chute mapping policy demonstrates strong performance under standard operating conditions, it exhibits significant performance degradation when faced with distribution shifts in package induction patterns (see Figure 6 in Section 5). This vulnerability to out-of-distribution scenarios motivates our robust formulation.

In this paper, our *objective is to introduce robustness to distribution shifts* in the learned chute mapping policies. We achieve this by incorporating group DRO into the MARL framework, giving rise to the proposed DRMARL approach. Our proposed framework ensures reliable and robust performance on Amazon robotic sortation warehouses, even under unforeseen future induction scenarios.

### 3 Distributionally Robust Multi-Agent Reinforcement Learning with Group DRO

In this section, we enhance the MARL chute mapping framework described in Section 2 by incorporating group Distributionally Robust Optimization (DRO) to handle uncertainty and variability in package induction patterns. This approach enables us to develop robust policies that perform well across diverse induction scenarios, including previously unseen induction patterns.

Compared to traditional stochastic optimization that assumes fixed probability distributions, DRO [49, 50, 51, 52, 53, 54, 55] takes a more general approach that defines an ambiguity set containing multiple plausible distributions derived from the available data. By optimizing for the worst-case scenario within this set, DRO learns policies that remain effective even when the testing distribution differs from the training conditions, as is the case in the chute mapping problem considered here.

#### 3.1 Group DRO

In DRO, the ambiguity set that captures uncertainty in the data-generating distribution can be defined in various ways. Group DRO [40, 41] offers an efficient way of defining the ambiguity set using a finite collection of known distributions. For robotic sortation warehouses, these distributions can be derived from historical induction patterns.

Following [39], we define the unknown distribution  $\tilde{\mathbb{P}}$  as a combination of  $m$  distributions  $\mathbb{P}_g$ , each indexed by a group  $g$  in the set  $\mathcal{G} = \{1, 2, \dots, m\}$ . The ambiguity set  $\mathfrak{M}$  is then defined as a convex combination of these groups:

$$\mathfrak{M} := \left\{ \tilde{\mathbb{P}} = \sum_{g=1}^m q_g \mathbb{P}_g \mid q \in \Delta_m \right\}, \tag{9}$$

where  $\Delta_m$  denotes the  $(m - 1)$ -dimensional probability simplex [56] (see Figure 3).

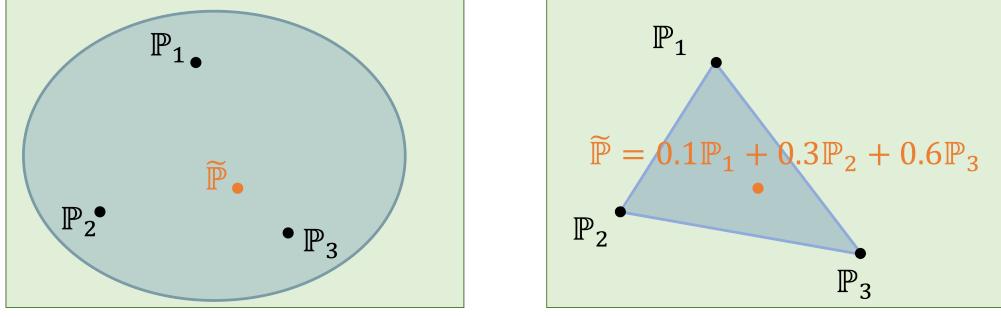


Figure 3: Ambiguity set  $\mathfrak{M}$  in regular DRO [51] (left) versus group DRO [39] (right).

In the dynamic chute mapping problem, we assume that past years of operational data from sortation warehouses provide sufficiently rich historical induction patterns that can be used to obtain representative distribution groups  $\mathcal{G}$ . With this assumption, it is reasonable to expect that any future induction pattern  $\tilde{\mathbb{P}}$  can be represented as a combination of the basis distributions  $\mathbb{P}_g$  with  $g \in \mathcal{G}$ . As shown in [39], evaluating the worst-case reward all  $m$  groups in  $\mathcal{G}$  is equivalent to evaluating the reward for the worst-case distribution within the ambiguity set  $\mathfrak{M}$  defined in (9).

**Lemma 3.1.** *Consider an ambiguity set  $\mathfrak{M}$  formed by  $\mathbb{P}_g$ s as defined in (9). For any state-action pair  $(s, a) \in \mathcal{S} \times \mathcal{A}$ , the worst-case expected reward satisfies:*

$$\inf_{g \in \mathcal{G}} \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)] = \inf_{\mathbb{P} \in \mathfrak{M}} \mathbb{E}_{X \sim \mathbb{P}} [r(s, a; X)], \quad (10)$$

where  $\mathcal{G}$  denotes the set of group indices.

Lemma 3.1 demonstrates a key advantage of group DRO: while general DRO problems are infinite-dimensional and computationally challenging, group DRO reduces the optimization to a finite-dimensional problem over  $m$  groups. This reduction makes the training of distributionally robust MARL (DRMARL) computationally tractable.

### 3.2 DRMARL with Group DRO

In the MARL framework described in Section 2, the policy parameters  $\theta$  are optimized as follows:

$$\theta^* := \arg \min_{\theta \in \Theta} \mathbb{E}_{X \sim \mathbb{P}} [L(\theta; X)], \quad (11)$$

and directly applying group DRO to MARL leads to:

$$\tilde{\theta} := \arg \min_{\theta \in \Theta} \left\{ \max_{g \in \mathcal{G}} \mathbb{E}_{X \sim \mathbb{P}_g} [L(\theta; X)] \right\}. \quad (12)$$

However, conventional group DRO approaches are not directly applicable to MARL problems, since minimizing the worst-case Bellman error does not necessarily lead to a policy that is optimal under worst-case rewards. Instead, our proposed DRMARL seeks a robust policy that selects actions that are optimal with respect to worst-case reward functions. To achieve this, we introduce the distributionally robust Bellman operator, defined in the following result.

**Lemma 3.2.** *For an ambiguity set  $\mathfrak{M}$  defined in (9) with group set  $\mathcal{G}$ , the distributionally robust Bellman operator is given by:*

$$\tilde{\mathcal{T}}_{\mathcal{G}}(\tilde{Q})(s, a) = \inf_{g \in \mathcal{G}} \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)] + \gamma \max_{a'} \tilde{Q}(s', a'),$$

---

**Algorithm 1** CB-based Worst-Case Reward Predictor

---

- 1: **Input:** Learning rate  $l_{\text{CB}}$ , initial parameters  $\psi_0$ , induction distribution groups  $\mathcal{G}$ , MARL policy with  $Q_{\text{MARL}}$ , exploration rate  $\varepsilon_{\text{CB}}$
  - 2: **Initialize:**  $\psi \leftarrow \psi_0$ , replay buffer  $\mathcal{D}_{\text{CB}} \leftarrow \emptyset$
  - 3: **for** episode = 1, ...,  $k_{\text{CB}}$  **do**
  - 4:   Initialize the environment with random group  $g' \in \mathcal{G}$  and observe initial state  $s_0$
  - 5:   **for** time step  $t = 0, \dots, T$  **do**
  - 6:     Select action  $a_t \leftarrow \arg \max_{a \in \mathcal{A}} Q_{\text{MARL}}(s_t, a)$
  - 7:     With probability  $\varepsilon_{\text{CB}}$ , select  $g' \sim \text{Uniform}(\mathcal{G})$ ; otherwise,  $g' \leftarrow \arg \min_{g \in \mathcal{G}} Q_{\text{CB}}(s_t, a_t, g; \psi)$
  - 8:     Execute action  $a_t$ , observe reward  $r_t$  and next state  $s_{t+1}$  using group  $g'$
  - 9:     Store transition  $(s_t, a_t, g', r_t, s_{t+1})$  in  $\mathcal{D}_{\text{CB}}$
  - 10:    Sample mini-batch from  $\mathcal{D}_{\text{CB}}$  and update  $\psi$ :
  - 11:      $\psi \leftarrow \psi - l_{\text{CB}} \nabla_{\psi} L_{\text{CB}}(\psi)$
  - 12:      $s_t \leftarrow s_{t+1}$ , reduce  $\varepsilon_{\text{CB}}$
  - 13:   **end for**
  - 14: **end for**
  - 15: **Output:** Optimized CB predictor parameters  $\psi^*$
- 

where  $\tilde{Q}$  is the distributionally robust  $Q$ -function.

Accordingly, the distributionally robust loss is given by

$$\tilde{L}(\theta; X) := \mathbb{E}_{s,a,r,a'} [(\tilde{Q}(s, a; \theta) - \inf_{g \in \mathcal{G}} \mathbb{E}_{X \sim \mathbb{P}_g}[r(s, a; X)] - \gamma \max_{a'} \tilde{Q}(s', a'; \bar{\theta}))^2], \quad (13)$$

and the distributionally robust parameters  $\tilde{\theta}_{\mathcal{G}}$  are optimized by solving

$$\tilde{\theta}_{\mathcal{G}} := \arg \min_{\theta \in \Theta} \tilde{L}(\theta; X). \quad (14)$$

## 4 Contextual Bandit-based Worst-Case Reward Predictor for DRMARL

Solving the MARL group DRO problem (14) is theoretically feasible since the worst-case reward can be evaluated for each  $(s, a)$  by exhaustively searching among all distribution groups  $\mathcal{G}$ . However, this approach is inefficient when the number of groups is large and forward simulation in the environment is expensive. This is particularly the case in the multi-agent dynamic chute mapping problem, where millions of packages are sorted across many destinations. Common group DRO techniques, such as soft reweighting [39], may not perform well in MARL because, unlike regression tasks, the data distribution depends on the agent's policy. As the policy evolves, the groups that are underrepresented or perform poorly may also change dynamically. This dynamic nature of the problem makes it challenging to apply static or even adaptive reweighting schemes, which assume a relatively stable data distribution.

To enhance the efficiency of the training process, we propose a *novel contextual bandit-based worst-case reward distribution predictor* that trains a contextual bandit (CB) [57, 58] model to predict the worst-case distribution group  $g \in \mathcal{G}$  for each state-action pair  $(s, a)$ .

### 4.1 CB-based Worst-Case Reward Predictor

The CB treats the current state-action pair  $(s, a)$  as contextual information, and its arms are defined over the groups of distributions in the set  $\mathcal{G}$ . The CB aims to predict the group  $g$  that minimizes the reward

---

**Algorithm 2** DRMARL with CB-based Worst-Case Reward Predictor

---

- 1: **Input:** Learning rate  $l_r$ , initial parameters  $\theta_0$ , induction distribution groups  $\mathcal{G}$ , pre-trained CB predictor  $Q_{\text{CB}}$ , exploration rate  $\varepsilon$
  - 2: **Initialize:**  $\theta \leftarrow \theta_0$ , replay buffer  $\mathcal{D} \leftarrow \emptyset$
  - 3: **for** episode = 1, ...,  $k$  **do**
  - 4:   Initialize the environment with random group  $g' \in \mathcal{G}$  and observe initial state  $s_0$
  - 5:   **for** time step  $t = 0, \dots, T$  **do**
  - 6:     With probability  $\varepsilon$ , select  $a_t \sim \text{Uniform}(\mathcal{A})$ ; otherwise,  $a_t \leftarrow \arg \max_{a \in \mathcal{A}} \tilde{Q}(s_t, a; \theta)$
  - 7:     Predict worst-case distribution group using CB:  $g' \leftarrow \arg \min_{g \in \mathcal{G}} Q_{\text{CB}}(s_t, a_t, g)$
  - 8:     Execute  $a_t$ , observe reward  $r_t$  and next state  $s_{t+1}$  using group  $g'$
  - 9:     Store transition  $(s_t, a_t, g', r_t, s_{t+1})$  in  $\mathcal{D}$
  - 10:    Sample mini-batch from  $\mathcal{D}$  and update parameters:  $\theta \leftarrow \theta - l_r \nabla_{\theta} \tilde{L}(\theta)$
  - 11:     $s_t \leftarrow s_{t+1}$ , reduce  $\varepsilon$
  - 12:   **end for**
  - 13: **end for**
  - 14: **Output:** Optimized DRMARL policy parameters  $\tilde{\theta}_{\mathcal{G}}$
- 

$\mathbb{E}_{X \sim \mathbb{P}_g} r(s, a; X)$ , which represents the worst-case reward among all groups. The CB is constructed as follows:

**Context space:**  $\mathcal{S} \times \mathcal{A}$ , where  $(s, a) \in \mathcal{S} \times \mathcal{A}$  represents the state and the chute mapping actions at each step.

**Action space:**  $\mathcal{G} = \{1, 2, \dots, m\}$ , where  $g \in \mathcal{G}$  denotes a group associated with an induction distribution.

**Reward:** A reward function  $r : (\mathcal{S} \times \mathcal{A}) \times \mathcal{G} \rightarrow \mathbb{R}$ , where  $r(s, a; X)$  represents the observed reward for choosing distribution  $\mathbb{P}_g$  at the current state-action pair  $(s, a)$ .

The CB is represented by a  $Q$ -function that approximates the expected reward of choosing distribution  $\mathbb{P}_g$  given a context  $(s, a)$ . For this purpose, we use an independent DQN [45]:

$$Q_{\text{CB}}(s, a, g; \psi) = \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)], \quad (15)$$

where the reward function  $r(s, a; X)$  is observed after running a single-step forward simulation with  $(s, a)$  under the induction-generating distribution  $\mathbb{P}_g$ . The  $Q_{\text{CB}}$  function is learned by minimizing the following loss, with the detailed training process shown in Algorithm 1:

$$L_{\text{CB}}(\psi) := \mathbb{E}_{s, a} [(Q_{\text{CB}}(s, a, g; \psi) - \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)])^2]. \quad (16)$$

In Algorithm 1, the exploration of state-action pairs  $(s, a)$  is guided by the existing MARL policy with  $Q_{\text{MARL}}$ , which ensures sufficient exploration of the context space for the chute-mapping problem. For other applications, different exploration policies can be employed, such as random action selection, to ensure adequate coverage of the state-action space.

## 4.2 DRMARL with CB-based Worst-Case Reward Predictor

Once the  $Q_{\text{CB}}$  function has been trained, we can rewrite the distributionally robust loss (13) as:

$$\tilde{L}(\theta; X) = \mathbb{E}_{s, a, r, a'} \left[ (\tilde{Q}(s, a; \theta) - \mathbb{E}_{X \sim \mathbb{P}_{g'}} [r(s, a; X)] - \gamma \max_{a'} \tilde{Q}(s', a'; \bar{\theta}))^2 \right], \quad (17)$$

where  $g' = \arg \min_{g \in \mathcal{G}} Q_{\text{CB}}(s, a, g)$  is the index of the distribution group with the predicted worst-case reward.

Table 1: Key sortation metrics comparison across policies, averaged over  $m = 9$  groups.

| Policy                  | Recirculation Rate ( $\downarrow$ )   | Throughput ( $\uparrow$ ) | Recirculation Amount ( $\downarrow$ ) |
|-------------------------|---------------------------------------|---------------------------|---------------------------------------|
| MARL                    | $2.16\% \pm 2.35\%$                   | 11740.98                  | 259.02                                |
| DRMARL (random)         | $1.56\% \pm 1.45\%$                   | 11812.77                  | 187.23                                |
| DRMARL (with $Q_{CB}$ ) | <b><math>0.56\% \pm 0.18\%</math></b> | <b>11932.21</b>           | <b>67.79</b>                          |
| DRMARL (exhaustive)     | $0.55\% \pm 0.13\%$                   | 11933.68                  | 66.32                                 |
| MARL (group-specific)   | $0.53\% \pm 0.14\%$                   | 11936.60                  | 63.40                                 |

The training procedure of DRMARL is shown in Algorithm 2. The key difference compared to MARL training is that DRMARL aims to train a DQN that estimates the worst-case return among all groups, while MARL aims to estimate the observed return only for a specific induction distribution. Moreover, in contrast to traditional group DRO, the index of the worst-case distribution  $g'$  in DRMARL is not obtained via exhaustive search; instead, it is predicted by  $Q_{CB}$  given the state-action pair  $(s, a)$ . The independent  $Q$ -network  $Q_{CB}$  is learned beforehand using Algorithm 1 and remains unchanged during DRMARL training. While it may seem counter-intuitive that  $Q_{CB}$  predicts the worst-case group after  $\tilde{Q}$  selects an action, this design enables  $Q_{CB}$  to provide the worst-case expected return for each  $(s, a)$  pair, thereby enabling the learning of a robust policy.

## 5 Simulation

In this section, we demonstrate the effectiveness of the proposed DRMARL policy under adversarial induction changes in both a simplified simulation and a large-scale Amazon robotic sortation warehouse simulation environment.

### 5.1 Simplified Robotic Sortation Warehouse Simulation

In the simplified simulation environment, there are 10 eject chutes, one recirculation chute, and 20 total unique destinations. Packages arrive at the sortation warehouse according to the induction data  $X$  generated from an induction distribution  $\mathbb{P}$ . When packages exceed the chutes' capacities, they are sent to the recirculation chute. One training/testing episode consists of 5 hours, with each time step being 30 minutes, after which the environment is reset. An eject chute can be reallocated at each time step. The implementation details are provided in Appendix B.

We train the DRMARL policy over 300 episodes using training data generated from 9 distinct induction distribution groups. Similarly, the regular MARL policy is trained for 300 episodes on the same groups. Due to the stochastic nature of the induction-generating distributions, the induction data differs for each simulation trial.

The comparison of key metrics is shown in Table 1. The DRMARL policy with  $Q_{CB}$  achieves the best performance across all metrics. For reference, the last row shows the theoretical optimal performance of group-specific MARL policies, which are trained and tested on the same group. DRMARL performs only *marginally* below this optimal baseline, demonstrating a favorable trade-off between individual performance and distributional robustness. Comparison with both random group selection and exhaustive worst-case search demonstrates that  $Q_{CB}$  efficiently identifies worst-case groups while learning an equally effective policy, performing significantly better than random selection and matching the robustness of exhaustive search. Here, random group selection refers to replacing Line 7 of Algorithm 2 with:

$$g' \leftarrow \text{Uniform}(\mathcal{G})$$

instead of using:

$$g' \leftarrow \arg \min_{g \in \mathcal{G}} Q_{CB}(s_t, a_t, g)$$

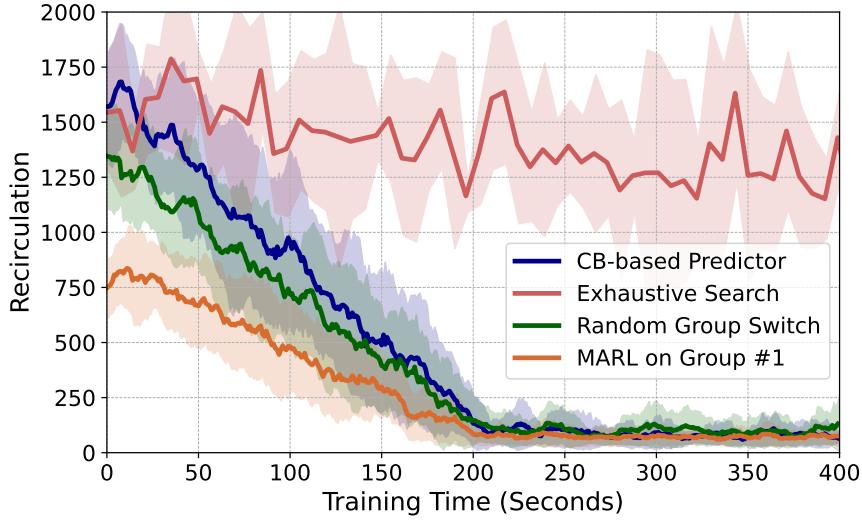


Figure 4: Training efficiency comparison in simplified warehouse.

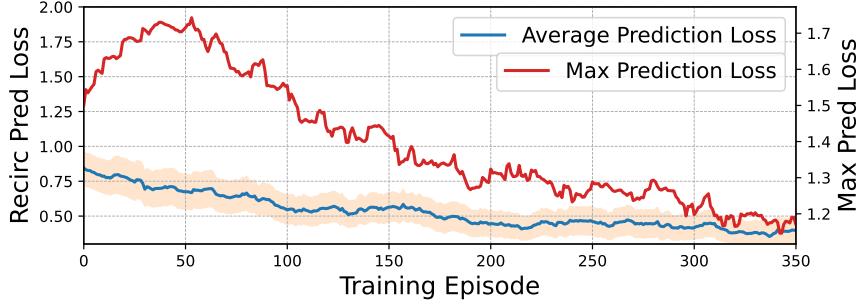


Figure 5: Training recirculation rate prediction loss (16) for CB in simplified robotic sortation warehouse.

This confirms that  $Q_{CB}$  effectively enables the DRMARL policy to explore worst-case reward functions during training. Furthermore, when compared to a DRMARL policy trained using exhaustive search over worst-case rewards for each state-action pair  $(s, a)$ , the  $Q_{CB}$ -based approach achieves equivalent policy performance and robustness while being computationally more efficient. The last row presents the theoretical optimal performance of group-specific MARL policies (trained and tested on the same group). DRMARL performs only marginally below this optimal baseline, demonstrating an effective balance between individual performance and distributional robustness.

Figure 5 illustrates the training progress of  $Q_{CB}$ , showing significant reductions in both average and maximum prediction errors of the recirculation rate throughout the training process. The training efficiency comparison across different approaches is presented in Figure 4. DRMARL with  $Q_{CB}$  requires less than 300 seconds to converge, while an exhaustive search for the worst-case group takes at least 2900 seconds to complete. DRMARL with exhaustive search requires the longest training time due to its comprehensive exploration across all distribution groups. In contrast, DRMARL with  $Q_{CB}$  converges significantly faster while matching the exhaustive search's recirculation performance in initial stages of the training, validating  $Q_{CB}$ 's ability to identify worst-case groups. DRMARL with random group selection shows faster convergence than the  $Q_{CB}$ -based approach, but this is because random exploration does not guarantee finding worst-case reward functions. The group-specific MARL policy exhibits the fastest convergence due to the relative simplicity of its training task.

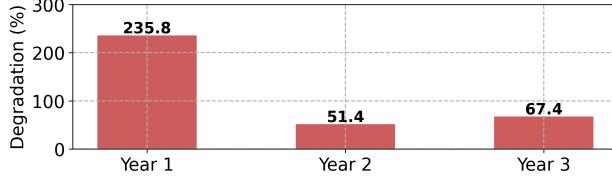


Figure 6: Relative degradation of MARL policy (trained on Year 4) on OOD (Years 1-3) induction data, compared to Year 4.

Table 2: Relative improvement ( $\uparrow$ ) of DRMARL over MARL baseline, averaged across  $m = 21$  groups.

| Policy                | Recirc Rate Reduction | Throughput Increase | Recirc Amount Reduction |
|-----------------------|-----------------------|---------------------|-------------------------|
| DRMARL                | 79.97%                | 5.62%               | 33.64%                  |
| MARL (group-specific) | 85.42%                | 9.80%               | 40.50%                  |

## 5.2 Large-Scale Amazon Robotic Sortation Warehouse Simulation

### 5.2.1 Implementation Details

In the large-scale simulation environment, there are 187 eject chutes, one recirculation chute, and 120 total unique destinations. Packages arrive at the sortation warehouse according to the corresponding induction data  $X$  generated from the induction distribution  $\mathbb{P}^2$ . When packages exceed chute capacities or miss departure transportation schedules, they are sent to the recirculation chute and added to the sequence of new packages at the next time step. One training/testing episode consists of 11 hours, with each time step being five minutes, after which the environment is reset. Every five minutes, we assign destinations to chutes that become available for reallocation.

We train the DRMARL policy over 200 episodes using training data generated from 21 distinct induction distribution groups spanning several years. Similarly, the regular MARL policy is trained for 200 episodes using induction data from Year 4. For testing, we evaluate both policies on newly generated induction data from 21 distinct distribution groups, conducting five experiments per generated induction. Due to the stochastic nature of the induction-generating distributions, the test induction data remains unseen during training for both policies.

### 5.2.2 Robustness of the Chute Mapping Policy

To motivate the need for the proposed DRMARL framework, we first study the performance of the regular MARL policy on out-of-distribution (OOD) induction data (e.g., induction data from Years 1-3), as shown in Figure 6. The results suggest that the regular MARL policy is not robust against OOD induction data.

Table 2 presents the average relative performance improvement of DRMARL across all 21 distribution groups, using MARL as the baseline. The proposed DRMARL method demonstrates robust performance on all induction groups, consistently outperforming the baseline MARL policy. For reference, the bottom row shows the theoretical optimal performance obtained by training and testing a group-specific MARL on each individual group. As expected, DRMARL performs *marginally* below these group-specific MARL policies, illustrating the trade-off between performance and distributional robustness. Detailed results are provided in Appendix D.

To evaluate DRMARL’s robustness beyond the ambiguity set  $\mathfrak{M}$ , we tested it on induction distributions  $\mathbb{P}'$  outside the group ambiguity set  $\mathfrak{M}$  ( $\mathbb{P}' \notin \mathfrak{M}$ ). As shown in Figure 7, DRMARL maintains consistent

<sup>2</sup>Due to common industry confidentiality practices, we cannot disclose the specific data source and report only relative performance improvements. The data represents realistic package flow patterns typical of Amazon robotic sortation facilities.

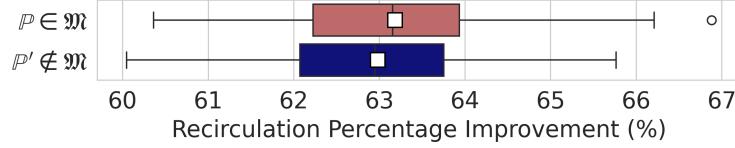


Figure 7: Recirculation rate improvement of DRMARL over two equally-performing MARL policies trained on distributions inside and outside  $\mathfrak{M}$ .



Figure 8: Training prediction loss (16) of  $Q_{CB}$  predictor, averaged over 11-hour simulations.

sortation performance and generalizes effectively to distributions even outside the ambiguity set  $\mathfrak{M}$ .

### 5.2.3 CB-based Worst-Case Reward Predictor

Following Section 4, we train an independent Q-network  $Q_{CB}$  to predict the worst-case recirculation (reward) for each state-action pair  $(s, a)$  across groups  $\mathcal{G}$ . The training of  $Q_{CB}$  uses the existing MARL policy, with the progress shown in Figure 8. The learned  $Q_{CB}$  achieves high accuracy, with prediction errors below 1% of the recirculation rate, enabling reliable identification of worst-case scenarios among groups  $g \in \mathcal{G}$ .

During each day’s 11-hour simulation, as illustrated in Figure 9,  $Q_{CB}$ ’s prediction accuracy improves substantially after the first hour. While initially suboptimal,  $Q_{CB}$ ’s performance remains sufficient for DRMARL training, as the impact of worst-case distributions on recirculation becomes more pronounced in later stages when fewer chutes are available.

### 5.2.4 Efficient Training with CB-based Worst-Case Reward Predictor

The CB-based worst-case reward predictor  $Q_{CB}$  substantially improves training efficiency by eliminating the need for exhaustive group evaluation at each time step, reducing the computational complexity of the worst-case group identification from  $\mathcal{O}(m)$  to  $\mathcal{O}(1)$ . As demonstrated in Figure 10, training with  $Q_{CB}$  achieves significantly faster convergence compared to exhaustive evaluation over  $\mathcal{G}$ , which requires approximately 924 hours on a cloud instance with 64 vCPUs (Intel Xeon Scalable 4th generation) and 128 GB RAM. The lightweight Q-network updates enabled CPU-only training, with most computation time spent on environment simulation. This efficiency advantage becomes even more pronounced when dealing with complex environments or larger group sets.

Figure 10 also compares the training efficiency of different approaches. The group-specific MARL, which trains on group #20, shows the fastest convergence due to its simplified learning objective. Among distributionally robust approaches, DRMARL with random group selection ( $g' \leftarrow \text{random}(\mathcal{G})$ ) converges initially faster than other variants but achieves suboptimal robustness since it may miss critical worst-case scenarios. DRMARL with  $Q_{CB}$  strikes a balance between training speed and performance, converging

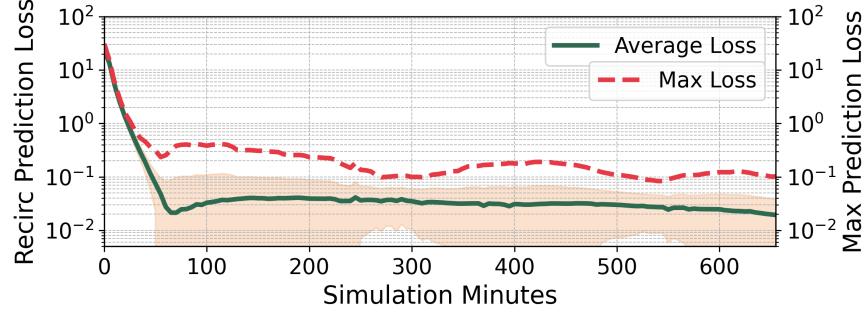


Figure 9:  $Q_{CB}$  prediction loss for recirculation percentage, averaged over 25 test simulations.

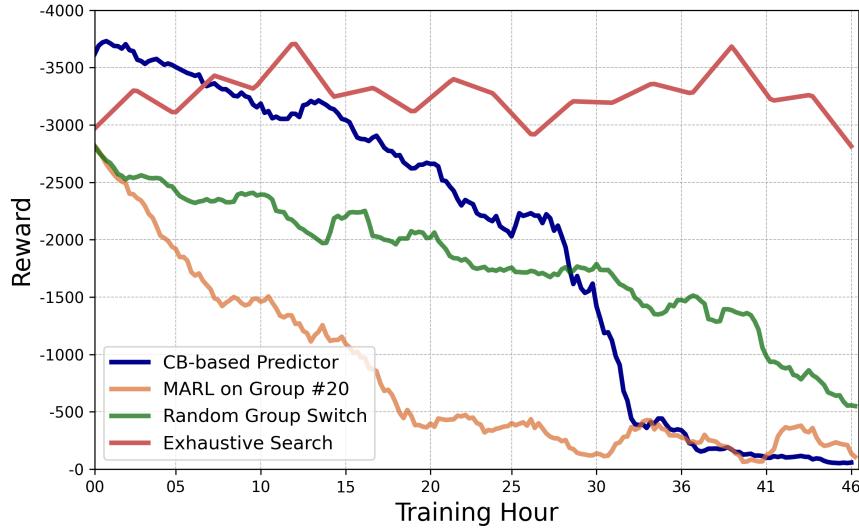


Figure 10: Training computational efficiency comparison in large-scale Amazon robotic sortation warehouse simulation environments.

significantly faster than exhaustive search while maintaining near-optimal worst-case performance guarantees. As expected, DRMARL with exhaustive search over all distribution groups requires the longest training time, though it serves as a valuable baseline for validating the efficiency of our  $Q_{CB}$ -based approach.

## 6 Conclusion

In this paper, we introduced DRMARL, a framework that integrates group-DRO into MARL to enhance policy robustness against adversarial distribution shifts in warehouse sortation systems. To address the computational cost of identifying the worst-case group, we developed a CB-based predictor that significantly reduces worst-case identification complexity from  $\mathcal{O}(m)$  to  $\mathcal{O}(1)$ . Experimental results from both simplified and large-scale warehouse environments demonstrate that DRMARL achieves near-optimal performance across all distribution groups while maintaining computational efficiency. The framework shows strong generalization even to distributions outside the training set, and its design principles can be extended to other MARL applications where distributional robustness is crucial.

## Acknowledgments

We would like to express our sincere gratitude to our colleagues at Amazon for their support and valuable contributions throughout this research. In particular, we extend special thanks to Rahul Chandan and Mouhacine Benosman for their insightful feedback and constructive discussions to this work.

## References

- [1] Peter R Wurman, Raffaello D’Andrea, and Mick Mountz. “Coordinating hundreds of cooperative, autonomous vehicles in warehouses”. In: *AI magazine* 29.1 (2008), pp. 9–9.
- [2] Kaveh Azadeh, René De Koster, and Debjit Roy. “Robotized and automated warehouse systems: Review and recent developments”. In: *Transportation Science* 53.4 (2019), pp. 917–945.
- [3] Amazon. *How Amazon robots navigate congestion*. <https://www.amazon.science/latest-news/how-amazon-robots-navigate-congestion>. 2022.
- [4] Amazon. *10 years of Amazon robotics: how robots help sort packages, move product, and improve safety*. <https://www.aboutamazon.com/news/operations/10-years-of-amazon-robotics-how-robots-help-sort-packages-move-product-and-improve-safety>. Accessed: 2023-01-23. 2022.
- [5] Amazon. *Tour an Amazon fulfillment center*. <https://www.aboutamazon.com/workplace/tours>. Accessed: 2023-01-23. 2023.
- [6] Yi Shen, Benjamin McClosky, Joseph W Durham, and Michael M Zavlanos. “Multi-Agent Reinforcement Learning for Resource Allocation in Large-Scale Robotic Warehouse Sortation Centers”. In: *2023 62nd IEEE Conference on Decision and Control (CDC)*. IEEE. 2023, pp. 7137–7143.
- [7] Nils Boysen and Malte Fliedner. “Cross dock scheduling: Classification, literature review and research agenda”. In: *Omega* 38.6 (2010), pp. 413–422.
- [8] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. “Multi-agent actor-critic for mixed cooperative-competitive environments”. In: *Advances in neural information processing systems* 30 (2017).
- [9] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. “Value-decomposition networks for cooperative multi-agent learning”. In: *arXiv preprint arXiv:1706.05296* (2017).
- [10] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. “The starcraft multi-agent challenge”. In: *arXiv preprint arXiv:1902.04043* (2019).
- [11] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. “Monotonic value function factorisation for deep multi-agent reinforcement learning”. In: *Journal of Machine Learning Research* 21.178 (2020), pp. 1–51.
- [12] Stefan Fedtke and Nils Boysen. “Layout planning of sortation conveyors in parcel distribution centers”. In: *Transportation Science* 51.1 (2017), pp. 3–18.
- [13] Luis J Novoa, Ahmad I Jarrah, and David P Morton. “Flow balancing with uncertain demand for automated package sorting centers”. In: *Transportation Science* 52.1 (2018), pp. 210–227.
- [14] Reem Khir, Alan Erera, and Alejandro Toriello. “Two-stage sort planning for express parcel delivery”. In: *IIE Transactions* 53.12 (2021), pp. 1353–1368.

- [15] Reem Khir, Alan Erera, and Alejandro Toriello. “Robust planning of sorting operations in express delivery systems”. In: *European Journal of Operational Research* (2022).
- [16] Hongrui Nie, Shaosheng Li, and Yong Liu. “Multi-agent deep reinforcement learning for resource allocation in the multi-objective HetNet”. In: *2021 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE. 2021, pp. 116–121.
- [17] Ruru Mei and Zhugang Wang. “Multi-Agent Deep Reinforcement Learning-Based Resource Allocation for Cognitive Radio Networks”. In: *IEEE Transactions on Vehicular Technology* (2024).
- [18] Wang Jun-Han, He He, Jaesang Cha, Incheol Jeong, and Ahn Chang-Jun. “Multi-Agent Reinforcement Learning for Efficient Resource Allocation in Internet of Vehicles”. In: *Electronics* 14.1 (2025), p. 192.
- [19] Xihan Li, Jia Zhang, Jiang Bian, Yunhai Tong, and Tie-Yan Liu. “A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network”. In: *arXiv preprint arXiv:1903.00714* (2019).
- [20] Jun Morimoto and Kenji Doya. “Robust reinforcement learning”. In: *Neural computation* 17.2 (2005), pp. 335–359.
- [21] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. “Robust Markov decision processes”. In: *Mathematics of Operations Research* 38.1 (2013), pp. 153–183.
- [22] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. “Robust adversarial reinforcement learning”. In: *International conference on machine learning*. PMLR. 2017, pp. 2817–2826.
- [23] Janosch Moos, Kay Hansel, Hany Abdulsamad, Svenja Stark, Debora Clever, and Jan Peters. “Robust reinforcement learning: A review of foundations and recent advances”. In: *Machine Learning and Knowledge Extraction* 4.1 (2022), pp. 276–315.
- [24] Vineet Goyal and Julien Grand-Clement. “Robust markov decision processes: Beyond rectangularity”. In: *Mathematics of Operations Research* 48.1 (2023), pp. 203–226.
- [25] Taku Yamagata and Raul Santos-Rodriguez. “Safe and Robust Reinforcement-Learning: Principles and Practice”. In: *arXiv preprint arXiv:2403.18539* (2024).
- [26] Huan Xu and Shie Mannor. “Distributionally robust Markov decision processes”. In: *Advances in Neural Information Processing Systems* 23 (2010).
- [27] Elena Smirnova, Elvis Dohmatob, and Jérémie Mary. “Distributionally robust reinforcement learning”. In: *arXiv preprint arXiv:1902.08708* (2019).
- [28] Linfang Hou, Liang Pang, Xin Hong, Yanyan Lan, Zhiming Ma, and Dawei Yin. “Robust reinforcement learning with Wasserstein constraint”. In: *arXiv preprint arXiv:2006.00945* (2020).
- [29] Shengbo Wang, Nian Si, Jose Blanchet, and Zhengyuan Zhou. “On the foundation of distributionally robust reinforcement learning”. In: *arXiv preprint arXiv:2311.09018* (2023).
- [30] Shyam Sundhar Ramesh, Pier Giuseppe Sessa, Yifan Hu, Andreas Krause, and Ilija Bogunovic. “Distributionally robust model-based reinforcement learning with large state spaces”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2024, pp. 100–108.
- [31] Zhengfei Zhang, Kishan Panaganti, Laixi Shi, Yanan Sui, Adam Wierman, and Yisong Yue. “Distributionally Robust Constrained Reinforcement Learning under Strong Duality”. In: *arXiv preprint arXiv:2406.15788* (2024).
- [32] Miao Lu, Han Zhong, Tong Zhang, and Jose Blanchet. “Distributionally Robust Reinforcement Learning with Interactive Data Collection: Fundamental Hardness and Near-Optimal Algorithm”. In: *arXiv preprint arXiv:2404.03578* (2024).

- [33] Allen Z Ren and Anirudha Majumdar. “Distributionally robust policy learning via adversarial environment generation”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1379–1386.
- [34] Zijian Liu, Qinxun Bai, Jose Blanchet, Perry Dong, Wei Xu, Zhengqing Zhou, and Zhengyuan Zhou. “Distributionally Robust  $Q$ -Learning”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 13623–13643.
- [35] Kaiqing Zhang, Tao Sun, Yunzhe Tao, Sahika Genc, Sunil Mallya, and Tamer Basar. “Robust multi-agent reinforcement learning with model uncertainty”. In: *Advances in neural information processing systems* 33 (2020), pp. 10571–10583.
- [36] Alexander Bukharin, Yan Li, Yue Yu, Qingru Zhang, Zehui Chen, Simiao Zuo, Chao Zhang, Songan Zhang, and Tuo Zhao. “Robust multi-agent reinforcement learning via adversarial regularization: Theoretical foundation and stable algorithms”. In: *Advances in Neural Information Processing Systems* 36 (2024).
- [37] Laixi Shi, Eric Mazumdar, Yuejie Chi, and Adam Wierman. “Sample-Efficient Robust Multi-Agent Reinforcement Learning in the Face of Environmental Uncertainty”. In: *arXiv preprint arXiv:2404.18909* (2024).
- [38] Laixi Shi, Jingchu Gai, Eric Mazumdar, Yuejie Chi, and Adam Wierman. “Breaking the Curse of Multiagency in Robust Multi-Agent Reinforcement Learning”. In: *arXiv preprint arXiv:2409.20067* (2024).
- [39] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. “Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization”. In: *arXiv preprint arXiv:1911.08731* (2019).
- [40] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. “Does distributionally robust supervised learning give robust classifiers?” In: *International Conference on Machine Learning*. PMLR. 2018, pp. 2029–2037.
- [41] Yonatan Oren, Shiori Sagawa, Tatsunori B Hashimoto, and Percy Liang. “Distributionally robust language modeling”. In: *arXiv preprint arXiv:1909.02060* (2019).
- [42] Tasuku Soma, Khashayar Gatmiry, and Stefanie Jegelka. “Optimal algorithms for group distributionally robust optimization and beyond”. In: *arXiv preprint arXiv:2212.13669* (2022).
- [43] Xunhao Wu and Jun Fu. “Distributed robust optimization for multi-agent systems with guaranteed finite-time convergence”. In: *arXiv preprint arXiv:2309.01201* (2023).
- [44] Mengdi Xu, Peide Huang, Yaru Niu, Visak Kumar, Jielin Qiu, Chao Fang, Kuan-Hui Lee, Xuewei Qi, Henry Lam, Bo Li, et al. “Group distributionally robust reinforcement learning with hierarchical latent variables”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2023, pp. 2677–2703.
- [45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. “Human-level control through deep reinforcement learning”. In: *nature* 518.7540 (2015), pp. 529–533.
- [46] Sujin Kim, Raghu Pasupathy, and Shane G Henderson. “A guide to sample average approximation”. In: *Handbook of simulation optimization* (2015), pp. 207–243.
- [47] Laurent Perron and Vincent Furnon. “OR-Tools”. Version v9.9. In: *Google* (2024). URL: <https://developers.google.com/optimization/>.
- [48] FICO. *Xpress Optimizer*. <https://www.fico.com/en/products/fico-xpress-optimization>. Accessed: 2023-01-23. 2023.

- [49] Erick Delage and Yinyu Ye. “Distributionally robust optimization under moment uncertainty with application to data-driven problems”. In: *Operations research* 58.3 (2010), pp. 595–612.
- [50] Alexander Shapiro. “Distributionally robust stochastic programming”. In: *SIAM Journal on Optimization* 27.4 (2017), pp. 2258–2275.
- [51] Hamed Rahimian and Sanjay Mehrotra. “Distributionally robust optimization: A review”. In: *arXiv preprint arXiv:1908.05659* (2019).
- [52] Jianzhe Zhen, Daniel Kuhn, and Wolfram Wiesemann. “A unified theory of robust and distributionally robust optimization via the primal-worst-equals-dual-best principle”. In: *Operations Research* (2023).
- [53] Guangyi Liu, Arash Amini, Vivek Pandey, and Nader Motee. “Data-Driven Distributionally Robust Mitigation of Risk of Cascading Failures”. In: *2024 American Control Conference (ACC)*. IEEE. 2024, pp. 3264–3269.
- [54] Daniel Kuhn, Soroosh Shafiee, and Wolfram Wiesemann. *Distributionally Robust Optimization*. 2024. arXiv: 2411.02549 [math.OC]. URL: <https://arxiv.org/abs/2411.02549>.
- [55] Xenia Konti, Hans Riess, Manos Giannopoulos, Yi Shen, Michael J Pencina, Nicoleta J Economou-Zavlanos, and Michael M Zavlanos. “Distributionally Robust Clustered Federated Learning: A Case Study in Healthcare”. In: *arXiv preprint arXiv:2410.07039* (2024).
- [56] Branko Grünbaum, Victor Klee, Micha A Perles, and Geoffrey Colin Shephard. *Convex polytopes*. Vol. 16. Springer, 1967.
- [57] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. “A contextual-bandit approach to personalized news article recommendation”. In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 661–670.
- [58] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. “A tutorial on thompson sampling”. In: *Foundations and Trends® in Machine Learning* 11.1 (2018), pp. 1–96.

## Appendix

### A Proofs of Theoretical Results

#### A.1 Proof of Lemma 3.1

Recall the definition of  $\mathfrak{M}$  in (9) and for any probability distribution  $\mathbb{P} \in \mathfrak{M}$ , we have

$$\begin{aligned}\mathbb{E}_{X \sim \mathbb{P}} [r(s, a; X)] &= \int r(s, a; X) \mathbb{P}(X) dX \\ &= \int r(s, a; X) \left( \sum_{g=1}^m q_g \mathbb{P}_g(X) \right) dX \\ &= \sum_{g=1}^m q_g \int r(s, a; X) \mathbb{P}_g(X) dX = \sum_{g=1}^m q_g \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)].\end{aligned}\tag{18}$$

Then, taking the infimum on both sides yields:

$$\begin{aligned}\inf_{\mathbb{P} \in \mathfrak{M}} \mathbb{E}_{X \sim \mathbb{P}} [r(s, a; X)] &= \inf_{q \in \Delta_m} \sum_{g=1}^m q_g \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)] \\ &= \inf_{g \in \mathcal{G}} \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)],\end{aligned}\tag{19}$$

since optimum of a linear program over simplex  $\Delta_m$  is obtained at vertices. ■

#### A.2 Proof of Lemma 3.2

In order to find the group distributionally robust action-value function  $\tilde{Q}$ , we consider the worst-case immediate reward at each state-action pair  $(s, a)$  as:

$$\tilde{r}(s, a) = \min_{g \in \mathcal{G}} \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)] \leq \mathbb{E}_{X \sim \mathbb{P}} [r(s, a; X)],\tag{20}$$

for all unknown distribution  $\mathbb{P} \in \mathfrak{M}$ . Then, the worst-case return is given by:

$$\tilde{R}_t = \sum_{k=0}^{\infty} \gamma^k \tilde{r}(s, a) \leq \sum_{k=0}^{\infty} \gamma^k \mathbb{E}_{X \sim \mathbb{P}_k} [r_{t+k+1}(s, a; X)],\tag{21}$$

where  $\gamma \in [0, 1]$  is the discount factor and the inequality holds for all possible sequences  $\{\mathbb{P}_k\}_{k=0}^{\infty}$  with  $\mathbb{P}_k \in \mathfrak{M}$ . Then, the worst-case action-value function under policy  $\pi$  can be expressed as:

$$\begin{aligned}\tilde{Q}^{\pi}(s, a) &= \mathbb{E}_{\pi} \left[ \tilde{r}(s, a) + \gamma \tilde{Q}^{\pi}(s_{t+1}, a_{t+1}) \mid s_t = s, a_t = a \right] \\ &= \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[ \tilde{r}(s, a) + \gamma \mathbb{E}_{a' \sim \pi(\cdot|s')} \tilde{Q}^{\pi}(s', a') \right],\end{aligned}\tag{22}$$

and the optimal worst-case action-value function satisfies the Bellman optimality equation:

$$\tilde{Q}^*(s, a) = \tilde{r}(s, a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[ \max_{a'} \tilde{Q}^*(s', a') \right].\tag{23}$$

Following the result from [34], the distributionally robust Bellman Operator with ambiguity sets  $\mathfrak{P}$  and  $\mathfrak{R}$  is given by

$$\tilde{\mathcal{T}}_{\mathfrak{P}, \mathfrak{R}}(\tilde{Q})(s, a) = \inf_{\substack{p_{s,a} \in \mathfrak{P} \\ r_{s,a} \in \mathfrak{R}}} \left\{ \mathbb{E}_{r_{s,a}}[r(s, a)] + \gamma \mathbb{E}_{p_{s,a}} \left[ \max_{a'} \tilde{Q}(s', a') \right] \right\}, \quad (24)$$

where  $p_{s,a}$  and  $r_{s,a}$  denote the distributions of state transitions probabilities and reward functions respectively, with their corresponding ambiguity sets  $\mathfrak{P}$  and  $\mathfrak{R}$ . Since the distribution shift in the random variable  $X$  only affects the reward, i.e., the distribution of the reward function  $r_{s,a}$ , we have:

$$\begin{aligned} \tilde{\mathcal{T}}_{\mathfrak{R}}(\tilde{Q})(s, a) &= \inf_{r_{s,a} \in \mathfrak{R}} \left\{ \mathbb{E}_{r_{s,a}}[r(s, a)] + \gamma \left[ \max_{a'} \tilde{Q}(s', a') \right] \right\} \\ &= \tilde{r}(s, a) + \gamma \max_{a'} \tilde{Q}(s', a') \\ &= \inf_{g \in \mathcal{G}} \{ \mathbb{E}_{X \sim \mathbb{P}_g}[r(s, a; X)] \} + \gamma \max_{a'} \tilde{Q}(s', a') = \tilde{\mathcal{T}}_{\mathcal{G}}(\tilde{Q})(s, a). \end{aligned} \quad (25)$$

We further show the distributionally robust Bellman operator is a contraction map under the  $\ell_\infty$  norm. Consider two arbitrary robust action-value functions  $\tilde{Q}_1$  and  $\tilde{Q}_2$  such that

$$\begin{aligned} \tilde{\mathcal{T}}_{\mathcal{G}}(\tilde{Q}_1)(s, a) &= \tilde{r}(s, a) + \gamma \max_{a'} \tilde{Q}_1(s', a') \\ \tilde{\mathcal{T}}_{\mathcal{G}}(\tilde{Q}_2)(s, a) &= \tilde{r}(s, a) + \gamma \max_{a'} \tilde{Q}_2(s', a'). \end{aligned} \quad (26)$$

Finding the difference yields

$$|\tilde{\mathcal{T}}_{\mathcal{G}}(\tilde{Q}_1)(s, a) - \tilde{\mathcal{T}}_{\mathcal{G}}(\tilde{Q}_2)(s, a)| \leq \gamma \max_{s', a'} |\tilde{Q}_1(s', a') - \tilde{Q}_2(s', a')|, \quad (27)$$

and taking the maximum over all feasible state-action pair  $(s, a)$  implies

$$\|\tilde{\mathcal{T}}_{\mathcal{G}}(\tilde{Q}_1) - \tilde{\mathcal{T}}_{\mathcal{G}}(\tilde{Q}_2)\|_{\ell_\infty} \leq \gamma \|\tilde{Q}_1 - \tilde{Q}_2\|_{\ell_\infty}. \quad (28)$$

Since  $\gamma \in [0, 1]$ , the robust Bellman operator is contraction map and the  $Q$ -learning algorithm will converge to  $\tilde{Q}^*$ . ■

## B Implementation Details for Simplified Robotic Sortation Warehouse

We define a Markov game for  $N$  agents (representing unique destinations) by the tuple  $(N, \mathcal{S}, \{\mathcal{O}^i\}_{i=1}^N, \{\mathcal{A}^i\}_{i=1}^N, P, \{r^i\}_{i=1}^N, \gamma, \rho_0)$ , where:

- (a) **Agents:** The set of  $N$  agents, each corresponding to a unique destination.
- (b) **State Space:**  $\mathcal{S}$  denotes the joint state space.
- (c) **Observation Space:** For each agent  $i$ ,  $\mathcal{O}^i \subset \mathcal{S}$  represents its local observation at each time step, consisting of:
  - The total number of assignable chutes (uniform across all agents)
  - The number of chutes currently assigned to agent  $i$
- (d) **Action Space:** For each agent  $i$ ,  $\mathcal{A}^i \subset [0, 1]$  represents its action space, where each action determines if a new chute will be allocated. An action value of 1 indicates the assignment of a new chute to destination  $i$ . The joint action space is defined as  $\mathcal{A} = \prod_{i=1}^N \mathcal{A}^i$ .

(e) **Transition Probability:**  $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$  specifies the probability of transitioning between states, representing the likelihood of packages being either successfully sorted or diverted to the recirculation buffer.

(f) **Reward Function:** For each agent  $i$ ,  $r^i : \mathcal{S} \times \mathcal{A} \times \mathcal{X} \rightarrow \mathbb{R}$  defines the reward function, which penalizes the number of packages in the recirculation buffer resulting from the current chute allocation.

The model is completed with discount factor  $\gamma \in (0, 1)$  and initial state distribution  $\rho_0$ . In Section 5.1, we employ the Value Decomposition Network (VDN) [6] combined with budget constraints in computing joint actions. This approach addresses both the scalability challenges of the state-action space and the computational feasibility of the expectation in (3). Detailed implementations are provided in Section 2.3 and Section 2.4.

In the simplified robotic sortation environment, we fix the total induction volume at each time step to 1200 packages. The number of incoming packages for each destination  $i = 1, \dots, N$  follows an unknown normal distribution  $\mathcal{N}(\mu, \sigma)$ . For each destination  $i$ , the probability that an incoming package is assigned to destination  $i$  is given by:

$$\mathbb{P}\{\text{incoming package belongs to } i\} = \frac{\Phi(\frac{i-\mu}{\sigma}) - \Phi(\frac{i-1-\mu}{\sigma})}{\Phi(\frac{N-\mu}{\sigma}) - \Phi(\frac{-\mu}{\sigma})}, \quad (29)$$

where  $\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^z e^{-\frac{t^2}{2}} dt$  is the cumulative distribution function of the standard normal distribution. This formulation ensures  $\sum_{i=1}^N \mathbb{P}\{\text{incoming package belongs to } i\} = 1$ . The distribution of packages across destinations is then determined by sampling 1200 packages according to the probabilities defined in (29) at each time step.

In Section 5.1, we assume the destination transportation vehicle has infinite capacity, meaning packages enter the recirculation buffer only when incoming packages are destined for a location without an assigned eject chute. In this example, we construct the ambiguity set as:

$$\mathfrak{M} := \left\{ \tilde{\mathbb{P}} = \sum_{g=1}^m q_g \mathbb{P}_g \mid q \in \Delta_m \right\}, \quad (30)$$

where each  $\mathbb{P}_g$  represents a normal distribution  $\mathcal{N}(\mu_g, \sigma)$ . For our simulation, we construct the ambiguity set using  $m = 9$  groups with means  $\mu_g \in \{-4, -3, \dots, 0, \dots, 4\}$ , standard deviation  $\sigma = 2$ , and index set  $\mathcal{G} = \{1, 2, \dots, 9\}$ .

## C Implementation Details for Large-Scale Amazon Robotic Sortation Warehouse

### C.1 Distributionally Robust Bellman Operator in Large-Scale Robotic Sortation Warehouses

In large-scale robotic sortation warehouses, we observe that the transition probability is dependent on the induction random variable  $X$ , which violates the assumption in Lemma 3.2. Consequently, we must compute:

$$\inf_{\mathbb{P} \in \mathfrak{M}} \gamma \mathbb{E}_{p_{s,a}(X), X \sim \mathbb{P}_g} \left[ \max_{a'} \tilde{Q}(s', a') \right]$$

for the distributionally robust Bellman operator. This leads to:

$$\tilde{\mathcal{T}}_{\mathfrak{R}}(\tilde{Q})(s, a) = \inf_{g \in \mathcal{G}} \left\{ \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)] \right\} + \inf_{\mathbb{P} \in \mathfrak{M}} \left\{ \gamma \mathbb{E}_{p_{s,a}(X), X \sim \mathbb{P}_g} \left[ \max_{a'} \tilde{Q}(s', a') \right] \right\} \quad (31)$$



Figure 11: Comparison of worst-case returns: predictions using  $Q_{\text{CB}}$  for both immediate rewards and transition probabilities versus observed values from extensive state-action space exploration. The close alignment validates our approach of using  $Q_{\text{CB}}$  to approximate both components of the worst-case scenario (32).

where the computation becomes infinite-dimensional and practically intractable. To address this, during training, we approximate the distributionally robust Bellman operator with:

$$\tilde{\mathcal{U}}_{\mathfrak{R}}(\tilde{Q})(s, a) = \inf_{g \in \mathcal{G}} \left\{ \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)] + \gamma \mathbb{E}_{p_{s,a}(X), X \sim \mathbb{P}_g} \left[ \max_{a'} \tilde{Q}(s', a') \right] \right\}, \quad (32)$$

which provides an upper bound for the distributionally robust Bellman operator, as shown by:

$$\begin{aligned} \tilde{\mathcal{U}}_{\mathfrak{R}}(\tilde{Q})(s, a) &\geq \inf_{g \in \mathcal{G}} \left\{ \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)] \right\} + \inf_{g \in \mathcal{G}} \left\{ \gamma \mathbb{E}_{p_{s,a}(X), X \sim \mathbb{P}_g} \left[ \max_{a'} \tilde{Q}(s', a') \right] \right\} \\ &\geq \inf_{g \in \mathcal{G}} \left\{ \mathbb{E}_{X \sim \mathbb{P}_g} [r(s, a; X)] \right\} + \inf_{\mathbb{P} \in \mathfrak{M}} \left\{ \gamma \mathbb{E}_{p_{s,a}(X), X \sim \mathbb{P}_g} \left[ \max_{a'} \tilde{Q}(s', a') \right] \right\} \\ &= \tilde{\mathcal{T}}_{\mathfrak{R}}(\tilde{Q})(s, a). \end{aligned} \quad (33)$$

During training, we use (32) to construct the loss function (17) for DRMARL, where the optimization problem within (32) is solved using the solution from the CB-based worst-case reward predictor  $Q_{\text{CB}}$ . In practice, this approximation (32) proves highly effective for the worst-case return, with the relative approximation error of  $\tilde{\mathcal{U}}_{\mathfrak{R}}(\tilde{Q})(s, a)$  to  $\tilde{\mathcal{T}}_{\mathfrak{R}}(\tilde{Q})(s, a)$  being less than 0.57% (see Figure 11). This small error margin indicates that  $\tilde{\mathcal{U}}_{\mathfrak{R}}(\tilde{Q})(s, a)$  does not impede DRMARL’s ability to observe the worst-case return.

## D Additional Simulation Result for Large-Scale Amazon Robotic Sortation Environments

Table 3 presents detailed validation results comparing MARL and DRMARL chute mapping policies across all induction distribution groups from Year 1-4. The DRMARL policy demonstrates superior performance across most groups, achieving both higher package sortation and lower recirculation rates. The only exceptions are two groups in Year 4, where the MARL policy shows marginally better throughput but at the cost of higher recirculation rates. This is expected behavior since the MARL policy is specifically trained on Year 4 induction data, while DRMARL optimizes for robustness rather than throughput maximization. Overall, DRMARL achieves significant improvements, reducing recirculation by 80% on average while simultaneously increasing throughput by 5.62% on average.

Table 3: Key metrics improvements over MARL baseline trained on Year 4 data in large-scale Amazon robotic sortation warehouses.

| GROUP NUMBER | YEAR | RECIRCULATION RATE REDUCTION (%) | PACKAGE THROUGHPUT INCREASE (%) | PACKAGE RECIRCULATION AMOUNT REDUCTION (%) |
|--------------|------|----------------------------------|---------------------------------|--|
| 1            | 1    | 94.75                            | 31.66                           | 94.21                                      |
| 2            | 1    | 93.19                            | 23.47                           | 92.55                                      |
| 3            | 1    | 94.62                            | 35.69                           | 94.26                                      |
| 4            | 1    | 93.85                            | 25.94                           | 93.85                                      |
| 5            | 1    | 95.90                            | 19.50                           | 95.67                                      |
| 6            | 2    | 90.47                            | 5.79                            | 90.44                                      |
| 7            | 2    | 91.27                            | 5.24                            | 91.11                                      |
| 8            | 2    | 77.22                            | 4.13                            | 77.22                                      |
| 9            | 2    | 83.34                            | 2.49                            | 83.34                                      |
| 10           | 2    | 85.12                            | 5.07                            | 85.12                                      |
| 11           | 2    | 92.85                            | 6.97                            | 92.82                                      |
| 12           | 3    | 84.22                            | 4.54                            | 84.59                                      |
| 13           | 3    | 83.14                            | 15.89                           | 81.57                                      |
| 14           | 3    | 85.53                            | 4.17                            | 86.04                                      |
| 15           | 3    | 83.63                            | 5.51                            | 83.58                                      |
| 16           | 3    | 84.24                            | 5.52                            | 84.19                                      |
| 17           | 4    | 34.18                            | -0.99                           | 36.71                                      |
| 18           | 4    | 57.04                            | -5.85                           | 62.34                                      |
| 19           | 4    | 75.78                            | 4.47                            | 75.73                                      |
| 20           | 4    | 66.59                            | 9.85                            | 64.18                                      |
| 21           | 4    | 75.07                            | 5.64                            | 75.83                                      |