Reinforcement Learning with Intrinsically Motivated Feedback Graph for Lost-sales Inventory Control

Zifan LIU[1] **Xinran LI**[1] **Shibo CHEN**[1]
Gen LI[2] **Jiashuo JIANG**[1,*] **Jun ZHANG**[1]
[1]The Hong Kong University of Science and Technology, [2]The Chinese University of Hong Kong

Abstract

Reinforcement learning (RL) has proven to be well-performed and versatile in inventory control (IC). However, further improvement of RL algorithms in the IC domain is impeded by two limitations of online experience. First, online experience is expensive to acquire in real-world applications. With the low sample efficiency nature of RL algorithms, it would take extensive time to collect enough data and train the RL policy to convergence. Second, online experience may not reflect the true demand due to the lost-sales phenomenon typical in IC, which makes the learning process more challenging. To address the above challenges, we propose a training framework that combines reinforcement learning with feedback graph (RLFG) and intrinsically motivated exploration (IME) to boost sample efficiency. In particular, we first leverage the MDP structure inherent in lost-sales IC problems and design the feedback graph (FG) tailored to lost-sales IC problems to generate abundant side experiences aiding in RL updates. Then we conduct a rigorous theoretical analysis of how the designed FG reduces the sample complexity of RL methods. Guided by these insights, we design an intrinsic reward to direct the RL agent to explore to the state-action space with more side experiences, further exploiting FG's capability. Experimental results on single-item, multi-item, and multi-echelon environments demonstrate that our method greatly improves the sample efficiency of applying RL in IC. Our code is available at `https://github.com/Ziffer-byakuya/RLIMFG4IC`

1 INTRODUCTION

Inventory control (IC) is a practical problem rooted in the real-world application of supply chain management (SCM). It mainly aims at minimizing operational costs while satisfying the demand of customers. The inherent complexity of modeling supply chain dynamics and demand uncertainty makes finding the optimal solutions for IC computationally intractable within reasonable time frames. To address this challenge, researchers (Zipkin, 2008a; Bai et al., 2023; Xin, 2021; Chen et al., 2024) have designed heuristic methods based on specific model assumptions over the past few decades. However, these methods face limitations in practice. Firstly, they struggle with the curse of dimensionality (Goldberg et al., 2021). As the lead time from order placement to reception increases, the problem size grows exponentially. Secondly, their reliance on specific assumptions may restrain their adaptability; they cannot easily generalize across diverse environmental settings. For example, most well-performed heuristic methods for single-item IC cannot be extended to multi-item or multi-echelon IC. These limitations call for a more adaptable approach to IC problems, for which data-driven control methods stand out as a promising alternative.

Reinforcement learning (RL) has shown promise in handling complex sequential decision-making problems. In particular, it offers several advantages for addressing the challenges of IC problems. Firstly, RL allows for the discovery of optimal policies without relying on strong problem-specific assumptions, enabling more generalizable solutions (Nian et al., 2020). Secondly, when combined with the deep neural network, Deep RL (DRL) can handle the high-dimensional state and action spaces typical of IC, thereby alleviating the curse of dimensionality (Dehaybe et al., 2024). Early works have demonstrated the feasibility of RL in IC problems. For instance, Oroojlooyjadid et al. (2022) showcases the ability of Deep Q-network (DQN) to discover near-optimal solutions for the widely recognized beer distribution game (a type of game for SCM simulation that

illustrates the complexities and challenges of IC). Gijsbrechts et al. (2022) achieves performance comparable to that of heuristic methods using the A3C algorithm in IC problems. Stranieri and Stella (2023) benchmarks various DRL methods such as A3C, PPO, and vanilla policy gradient (VPG) in IC problems. More recently, researchers have explored DRL for various IC scenarios, including non-stationary uncertain demand (Dehaybe et al., 2024; Park et al., 2023), multi-product (Sultana et al., 2020; Selukar et al., 2022), variable kinds of products (Meisheri et al., 2020, 2022), multi-echelon supply chains (Wu et al., 2023; Stranieri et al., 2024), one-warehouse multi-retailer (Kaynov et al., 2024), and the stochastic capacitated lot sizing problem (Van Hezewijk et al., 2023).

However, traditional RL methods suffer from low sample efficiency, presenting a major barrier to real-world implementation where experience collection can be costly and time-consuming (Boute et al., 2022; De Moor et al., 2022). Furthermore, this issue is escalated in lost-sales IC problems because of censored demands (Chen et al., 2024), where customers' real demands are unobservable due to insufficient inventory. For instance, if the order is placed daily, it takes over a year to collect four hundred experiences and part of them may be censored. The scarcity of valid experience presents a true challenge for training data-hungry RL policies. De Moor et al. (2022) tries to alleviate this problem by incorporating heuristic knowledge, such as the base-stock policy, into DQN with reward-shaping. Although this method can improve the sample efficiency, it still relies on specific model heuristics, making it hard to generalize to different scenarios. Therefore, overcoming the sample inefficiency of RL methods without assuming strong heuristics remains a key challenge for effectively applying these techniques to IC problems, especially under lost-sales consideration.

To address the above-mentioned limitations of DRL for IC problems, this paper proposes a novel training framework that combines reinforcement learning with feedback graphs (RLFG) and intrinsically motivated exploration (IME):

1) We tailor the feedback graph (FG) based on the general property of lost-sales IC problems rather than case-specific heuristics (e.g. known demand distribution). In particular, the connectivity of FG is adjusted dynamically based on the relationship between the demand and inventory to be adaptive to the lost-sales property. With FG, the sample efficiency in the training process is significantly improved by allowing the RL agent to acquire not only online experiences but also side experiences from FG.

2) We conduct theoretical analysis on how FG reduces

the sample complexity with Q-learning to reveal its mechanism of improving the update probabilities across all state-action pairs in IC problems.

3) Inspired by these theoretical insights, we design a novel intrinsic reward guiding the RL algorithm to explore towards the state-action space where more side experiences can be obtained thereby further boosting sample efficiency.

4) We evaluate our method across a wide range of IC settings, including the single-item, multi-item, and multi-echelon lost-sales inventory control environments. The empirical results demonstrate the superior sample efficiency and performance improvement of proposed methods, underlining the effectiveness of our designs and generalization ability to different scenarios.

2 RELATED WORK

In this section, we will briefly introduce heuristic methods and RL methods utilized in the lost-sales IC problem. Extended related work is provided in Appendix A.

2.1 Heuristic Methods in Lost-sales Inventory Control

Early research by Karlin (1958) demonstrated that the base-stock policy in backlogging scenarios is not optimal for lost-sales situations, as it fails to account for future inventory availability. Morton (1969), Zipkin (2008b) and Janakiraman and Roundy (2004) extend the former analysis by exploring upper and lower bounds of optimal policies, considering different and stochastic lead times. Another method, named constant-order policy, is proved to be better than base-stock policy by Reiman (2004) and then Goldberg et al. (2016) proves that the constant-order policy can be asymptotically optimal as the lead time goes to infinity. Xin (2021) improves this idea and propose capped base-stock policy. Nevertheless, the above-introduced methods need to search for the best parameter for specific settings to perform well. Besides them, another line of research introduces approximate dynamic programming (APD). These methods assume known demand distributions rather than relying on parameter searching. Morton (1971) proposes the Myopic method, which aims to minimize the expected cost when the placed order arrives.

2.2 Reinforcement Learning Methods in Lost-sales Inventory Control

Existing heuristic methods for IC need either a parameter search or an assumption about the demand

distribution, which are not general enough. This limitation has led to increased interest in applying RL to IC problems. While numerous RL methods (Oroojlooyjadid et al., 2022; Gijsbrechts et al., 2022; Stranieri and Stella, 2023) have demonstrated competitive performance compared to traditional heuristic approaches, a significant challenge persists: the low sample efficiency inherent in existing RL techniques. This inefficiency considerably constrains their practical application in real-world IC scenarios (Boute et al., 2022; De Moor et al., 2022), especially when obtaining experience is expensive or time-consuming. Besides, this sample inefficiency problem is esculated in lost-sales IC because of censored demands (Chen et al., 2024). De Moor et al. (2022) tries to alleviate this problem by incorporating heuristic knowledge, such as base-stock policy, into DQN with reward-shaping. Although this method can improve the sample efficiency, it still relies on specific heuristics, making it hard to generalize to a wider range of scenarios.

3 PRELIMINARY

3.1 Lost Sales Inventory Control Problem

Table 1: Nomenclature for environmental parameters.

Symbol	Representation
$t = 1, ..., T$	Time index
$a_t \in \mathbb{N}$	Action: order at time t
$a^{\max} \in \mathbb{N}$	Maximum order
$c \in \mathbb{R}$	Unit cost of procurement
$d_t \in \mathbb{N}$	Real demand at time t
$d_t^o \in \mathbb{N}$	Demand observed at time t
$d^{\max} \in \mathbb{N}$	Maximum demand
$h \in \mathbb{R}$	Unit cost of holding inventory
$L \in \mathbb{N}$	Order lead time
$p \in \mathbb{R}$	Unit cost penalty of lost sales
$\boldsymbol{s}_t \in \mathbb{N}^L$	State: $(y_t, a_{t+1-L}, ..., a_{t-1})$
$y_t \in \mathbb{N}$	Inventory after receiving the arrived order at time t.
$y^{\max} \in \mathbb{N}$	Maximum inventory after receiving the arrived order at time t.
$\gamma \in (0, 1]$	Discount factor

We formulate a standard, single-item, discrete-time, and lost-sales IC problem following Zipkin (2008a), Gijsbrechts et al. (2022) and Xin (2021) with environment variables defined in Table 1. In lost-sales settings, excess demand is unrecorded when inventory is insufficient, with customers leaving due to unsatisfied demands. The lost-sales IC problem considers three kinds of costs in each time step t, which are the cost of procurement $f_1(\boldsymbol{s}_t)$, the cost of holding inventory $f_2(\boldsymbol{s}_t)$, and the cost penalty of lost sales $f_3(\boldsymbol{s}_t)$. The cost terms are specified below,

$$
\begin{aligned}
&f_1(\boldsymbol{s}_t, a_t) = ca_t, \quad f_2(\boldsymbol{s}_t) = h[y_t - d_t]^+, \\
&f_3(\boldsymbol{s}_t) = p[d_t - y_t]^+, \text{where } [x]^+ = \max(x, 0).
\end{aligned} \tag{1}
$$

In this problem, the real demand is unobservable when it exceeds the current inventory. Thus we define the real demand d_t as a random variable and the observed demand d_t^o, which may be censored by y_t and d_t, given as

$$
d_t^o = \min(d_t, y_t) = \begin{cases} d_t, & \text{if } d_t \leq y_t \\ y_t, & \text{otherwise} \end{cases}. \tag{2}
$$

Here, the term "censored" refers to the case when $d_t > y_t$, i.e. we can only observe all inventory is sold but do not know the value of d_t. The objective is to minimize the total cost over the time horizon T under the uncertainty on the demand side, which is

$$
\begin{aligned}
\min_{\{a_t | t=0, ..., T\}} &\sum_{t=0}^{T} \gamma^t f(\boldsymbol{s}_t, a_t) \\
&= \sum_{t=0}^{T} \gamma^t [f_1(\boldsymbol{s}_t, a_t) + f_2(\boldsymbol{s}_t) + f_3(\boldsymbol{s}_t)],
\end{aligned} \tag{3}
$$

where $\{a_t | t = 0, ..., T\}$ denotes $\{a_0, a_1, ..., a_{T-1}, a_T\}$.

3.2 MDP Formulation

To better understand the lost-sales IC problem from the perspective of RL, we formulate it into an infinite-horizon MDP with discounted rewards (Li et al., 2020), represented by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. The MDP consists of the state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $P(\boldsymbol{s}'|\boldsymbol{s}, a)$, reward function R, and discount factor γ. The detailed composition of MDP is shown as follows:

State: The state includes the inventory at time t after receiving the orders and all upcoming orders due to lead time. We define it as $\boldsymbol{s}_t = (y_t, a_{t+1-L}, ..., a_{t-1})$, where $(a_{t+1-L}, ..., a_{t-1})$ are the orders placed $(L, ..., 1)$ time steps ago.

Action: The action is the amount to be ordered for future sales, given as $a_t = \{0 \leq a_t \leq a^{\max}, a_t \in \mathbb{N}\}$, where $a^{\max}$ is the maximum amount that can be ordered for this item.

Reward: Since the goal is to minimize the cumulative discounted cost, the reward at each time step t is defined as the opposite number of costs, which is $r_t = R(\boldsymbol{s}_t, a_t) = -f(\boldsymbol{s}_t, a_t)$.

Transition Function: The transition function is defined as $\boldsymbol{s}_t = (y_t, a_{t+1-L}, ..., a_{t-1}) \rightarrow \boldsymbol{s}_{t+1} = (y_{t+1}, a_{t+2-L}, ..., a_t)$. The inventory transition is defined as $y_{t+1} = [y_t - d_t]^+ + a_{t+1-L}$ due to the lead time. As a_{t+1-L} has been received, $a_{t+1+i-L}$ in $\boldsymbol{s}_{t+1}$ will replace a_{t+i-L} in $\boldsymbol{s}_t$ for $i = 1, ..., L-1$.

3.3 Reinforcement Learning with Feedback Graph

RLFG is proposed by Dann et al. (2020) to reduce the sample complexity of RL. In typical RL scenarios, an agent can only get one experience each time as feedback that can be used to update the policy. However, given prior knowledge about the environments, additional side experiences involving other states and actions may be observable. RLFG explores how RL leverages these side experiences by constructing a feedback graph[1] (FG). FG is denoted as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set $\mathcal{V} = \{\boldsymbol{v} | \boldsymbol{v} = (\boldsymbol{s}, a)\}$ and $\mathcal{E}$ is the edge set $\mathcal{E} = \{\boldsymbol{v} \rightarrow \hat{\boldsymbol{v}}\}$, formalized by the side information indicating that if the agent visits experience $\boldsymbol{v}$, then the corresponding side experiences $\hat{\boldsymbol{v}}$ are also observable. The total experiences $\mathcal{O}_t$ observed by the agent at timestep t from $\mathcal{G}$ is

$$\mathcal{O}_t(\mathcal{G}) = \{(\boldsymbol{s}_t, a_t, r_t, \boldsymbol{s}_{t+1})\} \cup \{\hat{\boldsymbol{s}}_t, \hat{a}_t, \hat{r}_t, \hat{\boldsymbol{s}}_{t+1}\}. \quad (4)$$

4 METHOD

4.1 IC Training Framework with RLFG and IME

We propose a training framework that incorporates RLFG and IME to enhance the sample efficiency of off-policy RL algorithms in IC problems. This framework relies on two primary assumptions causing sample scarcity. First, the real demand is a fixed distribution, which is unknown and may be censored. Second, the experiences collected in real-world operations are limited due to the cost of collecting online data. As illustrated in Figure 1, the FG module generates side experiences $\{(\hat{\mathbf{s}}_t, \hat{a}_t, \hat{r}_t, \hat{\mathbf{s}}_{t+1}, d_t^o)\}$ to improve the sample efficiency and the intrinsic reward module aids the exploration by guiding the RL algorithm to explore in state-action space with more side experiences. Here, the RL module can be any off-policy RL algorithm, such as DQN (Mnih et al., 2015), DDPG (Lillicrap, 2015), Rainbow (Hessel et al., 2018), or TD3 (Fujimoto et al., 2018). During model-updating steps, a batch

---

[1]Here "side experiences" bears the same meaning as "side observations" in Dann et al. (2020), and we use the term "side experiences" to avoid confusion with "observations" in partially observable MDPs.
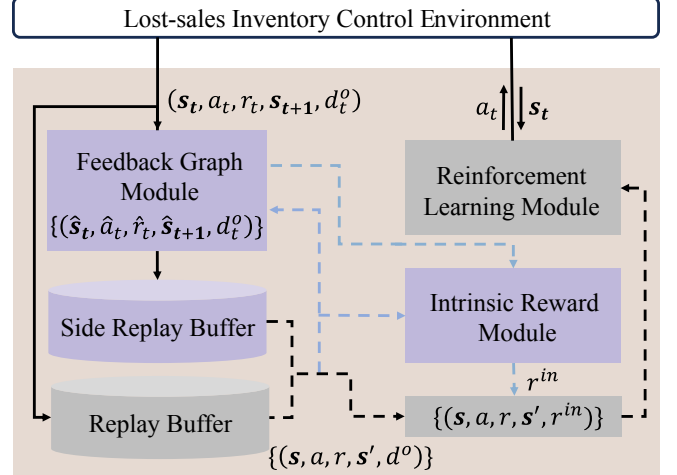

Figure 1: IC training framework based on RLFG and IME. Solid lines depict sampling steps and dashed lines represent model-updating steps.

of samples $\{(\mathbf{s}, \hat{a}, \hat{r}, \mathbf{s}', d_t^o)\}$ are sampled from both reply buffers. Subsequently, the FG module generates a new set of side experiences $\{(\hat{\mathbf{s}}, \hat{a}, \hat{r}, \hat{\mathbf{s}}', d_t^o)\}$, which are exclusively used to calculate the intrinsic rewards of the samples. Rainbow-FG is provided as an example in Algorithm 3 in Appendix B.

4.2 Feedback Graph in Inventory Control

Motivated by the ability of FG to reduce sample complexity (Dann et al., 2020), we incorporate it into the IC problems. FG is naturally suitable since IC problems follow structured MDP formation and have predictable transitions based on demand, typically independent of state-actions pairs. Observing the demand allows us to leverage the underlying properties of structured MDPs to obtain side experiences. Thus the key question is how to construct FG considering the potentially censored demand due to lost sales. To address this problem, we propose to construct FG dynamically based on the observed demand.

Algorithm 1 outlines the details of the FG module. When d_t^o is uncensored, FG forms a complete graph allowing all state-action pairs to generate side experiences based on d_t^o. When d_t^o is censored, only the state-action pairs with smaller inventory levels than $\boldsymbol{s}_t$ can be utilized to generate side experiences, resulting in a partially connected FG. Since d_t^o is censored, the store can only observe that all the inventory are sold out and assume that less inventory can also be sold out without any information for more inventory. Note that FG is still dynamically constructed under the censored case as the number of side experiences depends on d_t^o.

Algorithm 1 Feedback Graph Module.

1: **Input:** experience $(\boldsymbol{s}_t = (y_t, a_{t+1-L}, ..., a_{t-1}),$
   $a_t, r_t, \boldsymbol{s}_{t+1}, d_t^o)$
2: Initialize a temporary buffer $\mathcal{B} = \{\}$
3: **if** $d_t^o = y_t$ (censored case) **then**
4:    $y^{\text{bound}} = y_t$
5: **else**
6:    $y^{\text{bound}} = y^{\text{max}}$
7: **for** $\hat{y}_t = 0, ..., y^{\text{bound}}$ **do**
8:    **for** each term of $(\hat{a}_{t+1-L}, ..., \hat{a}_{t-1}) = 0, ..., a^{\text{max}}$
      do
9:      $\hat{s}_t = (\hat{y}_t, \hat{a}_{t+1-L}, ..., \hat{a}_{t-1})$
10:     **for** $\hat{a}_t = 0, ..., a^{\text{max}}$ **do**
11:       Get $\hat{r}_t$ and $\hat{\boldsymbol{s}}_{t+1}$ according to d_t^o
12:       $\mathcal{B} = \mathcal{B} \cup (\hat{\boldsymbol{s}}_t, \hat{a}_t, \hat{r}_t, \hat{\boldsymbol{s}}_{t+1}, d_t^o)$
13: Return $\mathcal{B}$

4.3 Theoretical Analysis

We conduct a quantitative analysis of how FG reduces sample complexity in the lost-sales IC environment. Here we provide an analysis based on Q-learning for simplicity. The qualitative analysis based on the property of RLFG is in Appendix C.

Without loss of generality, we restrict some definitions in Section 3.2 and define some new notations. We consider the reward function $R : \mathcal{S} \times \mathcal{A} \to (0, 1)$. The demand $d_t \sim P_d(d)$ follows any independent discrete distribution and $d^{\text{max}} < y^{\text{max}}$. We define π_b as the stationary behavior policy for easier analysis, $\mu(\boldsymbol{s}, a)$ as the stationary distribution of the Markov chain under π_b and $P(\boldsymbol{s}'|\boldsymbol{s}, a)$, and $\mu(\boldsymbol{s}, a|d)$ as the demand-conditioned stationary distribution. In standard RL, $\mu(s, a)$ can also be interpreted as the update probability since only visited state-action pairs can be updated. In RLFG, we denote $\tilde{\mu}(s, a)$ as the update probability since state-action pairs generated from FG can also be updated.

Scenario 1: Consider an edgeless graph $\mathcal{G}$ with all state-action pairs as the nodes.

Lemma 1: The sample complexity of the asynchronous Q-learning under scenario 1 is analyzed by Li et al. (2020), which is $\tilde{O}(\frac{1}{\mu_{\text{min}}(1-\gamma)^5\epsilon^2} + \frac{t_{\text{mix}}}{\mu_{\text{min}}(1-\gamma)})$, where $\mu_{\text{min}} = \min_{(s,a)\in\mathcal{G}} \mu(\boldsymbol{s}, a)$ and t_{mix} is the mixing time of the Markov chain.

Lemma 1 indicates that the sample complexity of Q-learning without FG is determined by μ_{min}. In the following parts, we will demonstrate: **Although the stationary distribution is unchanged, RLFG improves sample efficiency because of increased update probability ($\tilde{\mu}(\boldsymbol{s}, a) \geq \mu(\boldsymbol{s}, a)$) due to the side experiences.** From scenario 1 to scenario 6, we generalize the condition from RL to RLFG, which is

provided in Appendix D. Here we directly show the final condition of RLFG in scenario 6.

Scenario 6: Consider a graph $\mathcal{G}$ with all state-action pairs as the nodes. Consider all nodes satisfying $y \geq d_t$, if one is sampled by the policy, all can be sampled by the FG module and updated simultaneously. For nodes with $y < d_t$, only nodes with $y' \leq y$ can be sampled by the FG module and updated simultaneously.

Theorem 1: The update probability for Q-learning with FG under scenario 6 is Equation (5). The proof is provided in Appendix D.

$$
\tilde{\mu}(\boldsymbol{s}, a) = \underbrace{\sum_{d=0}^{d^{\text{max}}} P_d(d) \sum_{\substack{(\hat{\boldsymbol{s}}, \hat{a}) \in \mathcal{G} \\ \hat{y} \geq d}} \mu(\hat{\boldsymbol{s}}, \hat{a}|d)}_{\text{Uncensored term}}
$$
$$
+ \underbrace{\sum_{d=0}^{y-1} P_d(d) \sum_{\substack{(\hat{\boldsymbol{s}}, \hat{a}) \in \mathcal{G} \\ y \leq \hat{y} \leq d}} \mu(\hat{\boldsymbol{s}}, \hat{a}|d)}_{\text{Censored term}}. \quad (5)
$$

Remark: We can obtain the relationship between $\tilde{\mu}(\boldsymbol{s}, a)$ and $\mu(\boldsymbol{s}, a)$ in Equation (6), which shows that FG improves the update probability for all the state-action pairs and thus improves the μ_{min}, which is the state-action pair with minimum of update probability.

$$
\tilde{\mu}(\boldsymbol{s}, a) = \mathbb{E}_{d \sim P_d}[\tilde{\mu}(\boldsymbol{s}, a|d)] \geq \mathbb{E}_{d \sim P_d}[\mu(\boldsymbol{s}, a|d)] = \mu(\boldsymbol{s}, a). \quad (6)
$$

4.4 Intrinsically Motivated Exploration

To further leverage the benefits of FG, we design a curiosity-driven and FG-driven intrinsic reward to direct the RL algorithm towards the state-action space where more side experiences can be generated. For each state-action pair, the quantity and average uncertainty of the side experiences generated by this state-action pair are incorporated into the intrinsic reward.

This idea is motivated by the theoretical analysis based on Equation 5. The uncensored term reflects the contribution of the uncensored case to improve the update probability. Enhancing this term can improve the update probability of all state-action pairs. The censored term measures the benefits a state-action pair gains from the censored cases. In this scenario, the update probability can be improved from the state-action pairs with larger inventory levels. To satisfy both conditions, manually designing a behavior policy is difficult and not general enough. Instead, we propose an intrinsic

reward given in Equation 7 promoting side experience generation.

$$r_i^{in} = r_i^{in} + log_{10}(J) \times \frac{1}{J} \sum_{j=1}^{J} r_{i,j}^{in}, \tag{7}$$

where J is the number of the side experiences generated by an experience. Algorithm 2 shows the details of the intrinsic reward module. We utilize the M-head DQN (Nikolov et al., 2018) with each head trained by different mini-batches of experiences to get the prediction error as the curiosity. The final intrinsic reward, denoted in Equation 7, consists of the curiosity of the experience itself (r_i^{in}) and the averaged curiosity of all corresponding side experiences ($r_{i,j}^{in}$). The averaged value preserves the advantage of scale invariance but loses the quantity information compared with the sum value. To balance these two aspects, we multiply $log_{10}(J)$ with the averaged value to incorporate more quantity information into the intrinsic reward. In the censored case, where $J = y_t$, a larger J increases the intrinsic reward, which means more side experiences. In the uncensored case, J has no differences, and a larger intrinsic reward indicates higher curiosity. Across the censored and uncensored case, $J = y^{max}$ is larger than y_t, which means that the intrinsic reward in the uncensored case is more likely to be larger than that in the censored case.

Algorithm 2 Intrinsic Reward Module.
_____
1: **Input:** Mini-batch experiences $\{(\boldsymbol{s}, a, r, \boldsymbol{s}', d^o)\}$ and all side experiences $\{(\hat{\boldsymbol{s}}, \hat{a}, \hat{r}, \hat{\boldsymbol{s}}', d^o)\}$
2: **for** each experience $(\boldsymbol{s}_i, a_i, r_i, \boldsymbol{s}_i', d_i^o)$ in $\{(\boldsymbol{s}, a, r, \boldsymbol{s}', d^o)\}$ **do**
3: $\quad \bar{Q}^{dqn} = \frac{1}{M} \sum_{m=1}^{M} Q_m^{dqn}(\boldsymbol{s}_i, a_i)$
4: $\quad r_i^{in} = \frac{1}{M} \sqrt{\sum_{m=1}^{M} [Q_m^{dqn}(\boldsymbol{s}_i, a_i) - \bar{Q}^{dqn}]^2}$
5: $\quad \{(\hat{\boldsymbol{s}}_i, \hat{a}_i, \hat{r}_i, \hat{\boldsymbol{s}}_i', d_i^o)\}$ are all side experiences generated from $(\boldsymbol{s}_i, a_i, r_i, \boldsymbol{s}_i', d_i^o)$
6: $\quad J$ is the size of $\{(\hat{\boldsymbol{s}}_i, \hat{a}_i, \hat{r}_i, \hat{\boldsymbol{s}}_i', d_i^o)\}$
7: $\quad$ **for** each side experience $(\hat{\boldsymbol{s}}_{i,j}, \hat{a}_{i,j}, \hat{r}_{i,j}, \hat{\boldsymbol{s}}_{i,j}', d_i^o)$ in $\{(\hat{\boldsymbol{s}}_i, \hat{a}_i, \hat{r}_i, \hat{\boldsymbol{s}}_i', d_i^o)\}$ **do**
8: $\quad\quad \bar{Q}^{dqn} = \frac{1}{M} \sum_{m=1}^{M} Q_m^{dqn}(\hat{\boldsymbol{s}}_{i,j}, \hat{a}_{i,j})$
9: $\quad\quad r_{i,j}^{in} = \frac{1}{M} \sqrt{\sum_{m=1}^{M} [Q_m^{dqn}(\hat{\boldsymbol{s}}_{i,j}, \hat{a}_{i,j}) - \bar{Q}^{dqn}]^2}$
10: $\quad r_i^{in} = r_i^{in} + log_{10}(J) \times \frac{1}{J} \sum_{j=1}^{J} r_{i,j}^{in}$
11: $\quad$ Store r_i^{in} into $(\boldsymbol{s}_i, a_i, r_i, \boldsymbol{s}_i', d_i^o)$
12: Return $\{(\boldsymbol{s}, a, r, \boldsymbol{s}', d^o, r^{in})\}$
_____

5 EXPERIMENT

To verify the performance and sample efficiency of our training framework, we test our method, including (1)

Rainbow-FG, (2) **TD3-FG**, and (3) **Rainbow-FG (H)**, on the (a) **single-item**, (b) **multi-item**, and (c) **multi-echelon** lost-sales IC environment. Rainbow-FG (H) is a variant, which utilizes the heuristic knowledge to continue finetuning Rainbow-FG. It can be regarded as the upper bound of Rainbow-FG. All experiments are averaged on 20 random seeds with shaded areas representing the standard deviation.

5.1 Baselines

Heuristic Methods: We compare our method with widely recognized and well-performed heuristic methods with optimized parameters via brute force search.

- **Constant Order**: Fixed order $a_t = r^h$ with r^h being a parameter.

- **Myopic 1-period** (Morton, 1971): Assume the distribution of d_t is known. This method aims to minimize the expected cost at $t + L$. The order is $a_t = \min_{P(y_{t+L} - d_{t+L} < 0) \leq \frac{c+h}{p+h}} a_t$, where $0 \leq a_t \leq a^{max}$.

- **Myopic 2-period** (Zipkin, 2008a): This method considers 2-step expected costs from $t + L$ compared with Myopic 1-period method.

- **Base-Stock method** (Zipkin, 2008a): This method aims to maintain the inventory level with considering upcoming orders. The order is $a_t = (S^h - \boldsymbol{1} \cdot \boldsymbol{s}_t)^+$, where S^h is a parameter.

- **Capped Base-Stock method** (Xin (2021)): It combines Base-Stock and Constant Order method. The order is $a_t = \min[(S^h - \boldsymbol{1} \cdot \boldsymbol{s}_t)^+, r^h]$, where S^h and r^h are parameters.

- **Bracket method** (Bai et al. (2023)): A variant of the Constant Order method. The order is $a_t = \lfloor (t+1)r^h + \theta^h \rfloor - \lceil tr^h + \theta^h \rceil$, where r^h and θ^h are parameters.

Deep RL Methods:

We choose to compare the final results on (1) Rainbow; (2) TD3; (3) model-based policy optimization (MBPO); and (4) the A3C method. As for sample efficiency, we choose Rainbow-FG & Rainbow and TD3-FG & TD3 for comparison.

5.2 Case One: Single Item

We test our methods and baselines across different settings of the single-item lost-sales IC environment with demand obeying Poisson distribution. The parameters of the environment and the hyperparameters of our methods are in Appendix E. The parameters of the

Table 2: Optimal result comparison on single-item environment. For each method, the first row is the optimal average cost (↓) and the second row is the optimality gap (↓). The bold values indicates the values of the best five methods. p denotes the lost-sales penalty, and L denotes the lead time.

Method	$p=4$			$p=9$		
	$L=2$	$L=3$	$L=4$	$L=2$	$L=3$	$L=4$
Optimal	4.40	4.60	4.73	6.09	6.53	6.84
Constant Order	5.27	5.27	5.27	10.27	10.27	10.27
	19.8%	14.6%	11.4%	68.6%	57.3%	50.1%
Bracket	5.00	5.01	5.02	8.02	8.02	8.03
	13.6%	8.9%	6.1%	31.7%	22.8%	17.3%
Myopic 1-period	4.56	4.84	5.06	6.22	6.80	7.20
	3.7%	5.3%	7.1%	2.1%	4.1%	5.3%
Myopic 2-period	**4.41**	**4.64**	**4.82**	**6.10**	**6.57**	**6.92**
	0.2%	**0.8%**	**1.9%**	**0.2%**	**0.6%**	**1.2%**
Base-Stock	4.64	4.98	5.20	6.32	6.86	7.27
	5.5%	8.2%	9.9%	3.7%	5.1%	6.4%
Capped Base-Stock	**4.41**	**4.63**	**4.80**	**6.12**	**6.62**	**6.91**
	0.2%	**0.7%**	**1.5%**	**0.5%**	**1.4%**	**1.0%**
A3C	4.54	4.74	5.05	6.38	6.73	7.07
	3.2%	3.0%	6.7%	4.8%	3.1%	3.4%
MBPO	4.49	4.71	4.90	6.30	6.86	7.21
	2.0%	2.3%	3.5%	3.4%	5.1%	5.4%
Rainbow	4.52	4.72	4.91	6.28	6.72	7.09
	2.7%	2.6%	3.8%	3.1%	2.9%	3.7%
Rainbow-FG (ours)	**4.48**	**4.67**	**4.87**	**6.22**	**6.73**	**6.99**
	1.8%	**1.5%**	**2.9%**	**2.1%**	**3.1%**	**2.2%**
TD3	4.53	4.76	4.87	6.29	6.78	7.07
	2.9%	3.4%	2.9%	3.2%	3.8%	3.3%
TD3-FG (ours)	**4.42**	**4.62**	**4.76**	**6.14**	**6.62**	**6.90**
	0.4%	**0.4%**	**0.6%**	**0.8%**	**1.4%**	**0.9%**
Rainbow-FG (H) (ours)	**4.404**	**4.61**	**4.775**	**6.12**	**6.60**	**6.92**
	0.1%	**0.2%**	**0.9%**	**0.5%**	**1.1%**	**1.2%**

environment are based on standard settings in the existing SCM literature (Zipkin, 2008a; Gijsbrechts et al., 2022; Bai et al., 2023). For DRL hyperparameters, we use common settings without fine-tuning for individual experiments to demonstrate the generalizability of our method. In the following sections, default parameters are used unless stated otherwise. The hyperparameter analysis is demonstrated in Appendix H.

5.2.1 Optimal Results Comparison

Table 2 presents the average results and optimality gap compared to the optimal results. Without strong heuristics, Rainbow, TD3, MBPO, and A3C demonstrates similar performance whereas Rainbow-FG and TD3-FG achieves superior results. In particular, TD3-FG outperforms Myopic 2-period and Capped Base-Stock method and achieve SOTA performance in some settings without any strong heuristics. This notable improvement can be attributed to the ability to leverage a broader range of side experiences from FG, enabling TD3-FG to converge towards better solutions compared with TD3, and even existing best methods.

Although Rainbow-FG does not surpass the top-performing heuristic methods like Myopic 2-period and Capped Base-Stock, Myopic 2-period method assumes the distribution of the demand is known, which is a strong assumption, and the Capped Base-Stock method needs extensive parameter searches to attain this optimal result. These parameters vary across different scenarios without patterns. Conversely, Rainbow-FG consistently achieves near-optimal performance and possesses adaptability to diverse settings without these strong heuristics. Furthermore, when equipped with the heuristic knowledge, Rainbow-FG (H) attains equivalent or even superior performance compared to the best heuristic methods, particularly in settings ($p = 4, L = 2, 3, 4$). This result indicates that Rainbow-FG has the potential to perform better than heuristic
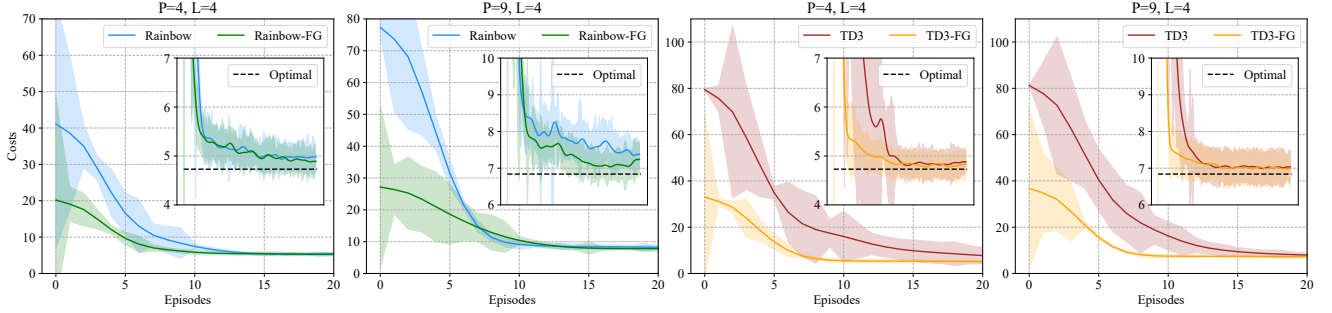
Figure 2: Single-item.

methods.

5.2.2 Sample Efficiency Comparison

Figure 2 illustrates the learning process of Rainbow versus Rainbow-FG and TD3 versus TD3-FG under two settings. The utilization of FG significantly enhances the sample efficiency. In the early stages of training, Rainbow-FG and TD3-FG show great sample efficiency compared with Rainbow and TD3. Towards the end, Rainbow-FG and TD3-FG converge faster and better compared to Rainbow and TD3, where Rainbow and TD3 fail to converge to the level of their FG version throughout the entire 100 episodes in some settings. Furthermore, FG contributes to improving the stability of the learning process. Notably, Rainbow-FG and TD3-FG exhibit lower standard deviation and more stable learning curves for the whole learning process. Experiments on more settings are shown in Appendix G.

5.3 Case Two: Multi Item

We further conduct experiments with TD3 and TD3-FG on the more challenging multi-item lost-sales IC problem with 2, 3, and 5 items in one store. Note that Rainbow and Rainbow-FG is not included in the comparison due to its exponential output growth with items. The parameters of the environment are in Appendix E.

5.3.1 Optimal Results Comparison

Figure 5 shows the final results in the multi-item environment. It demonstrates that TD3-FG can achieve better performance than TD3 in all cases. In particular, TD3-FG decreases the performance gap by 4.58%, 1.16%, and 2.56% for environments with 2, 3, and 5 items.

5.3.2 Sample Efficiency Comparison

Figure 3 demonstrates similar learning processes of TD3 and TD3-FG with different numbers of items like sing-item environment. TD3-FG shows great sample

efficiency and better stability throughout the training. As items increase, the efficiency gap between TD3 and TD3-FG widens, while TD3-FG maintains consistent gains, highlighting the scalability of the proposed methods.

5.4 Case Three: Multi Echelon

We compare TD3 and TD3-FG on multi-echelon lost-sales IC with 1 warehouse and 1, 2 or 4 retailers to evaluate generalization. The retailers order from the warehouse and satisfy the demand of customers, and the warehouse orders from the manufacturer and satisfies the demand of retailers. Both the warehouse and retailers need to find optimal decisions. The parameters of the environment are in Appendix E.

5.4.1 Optimal Results Comparison

Figure 5 shows the final results in the multi-echelon environment. It demonstrates that TD3-FG can achieve lower costs than TD3 with lower variance. In particular, TD3 performs very poorly when the number of retailers is 4. This results from failed warehouse decisions, as its demand comprises retailer orders. Since the demand for the warehouse is the sum of the orders from retailers, the mean and variance of the demand for the warehouse also increase as the number of retailers increases. Due to failed warehouse decisions, the retailers also cannot find optimal solutions to satisfy the demand of customers. Thus both the warehouse and retailers are in a greatly lost-sales condition.

5.4.2 Sample Efficiency Comparison

Figure 4 demonstrates similar learning curves of TD3 and TD3-FG with different numbers of retailers like multi-item environment. TD3-FG always achieves better performance with smaller variance under the same training budgets. Besides, TD3-FG converges to near-optimal solutions within around 20 episodes but TD3 converges to worse solutions with 80 to 100 episodes when the numbers of retailers are 1 and 4.
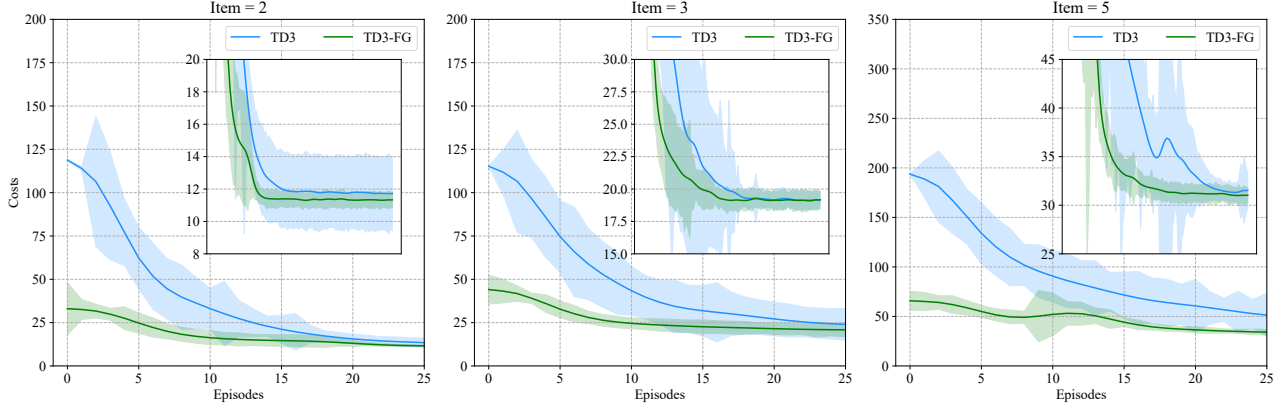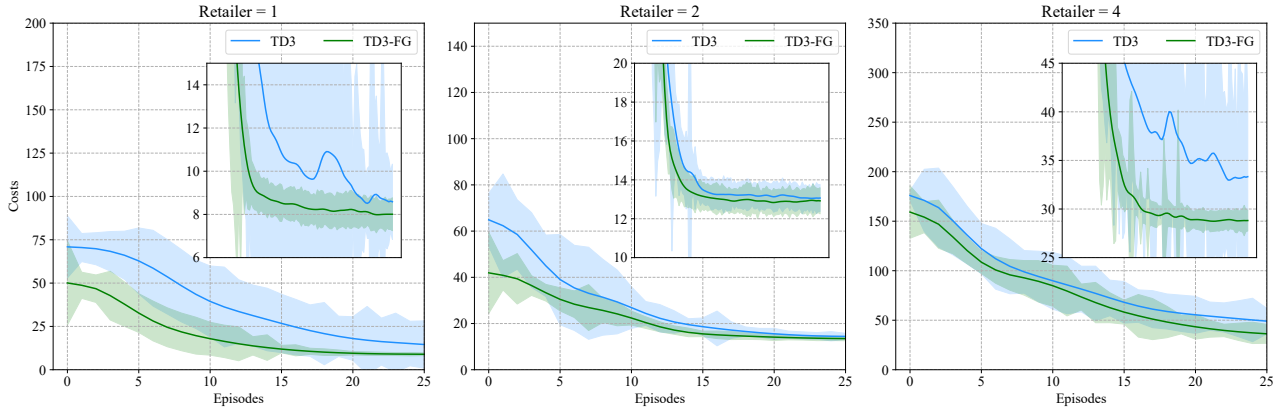
Figure 3: Multi-item.
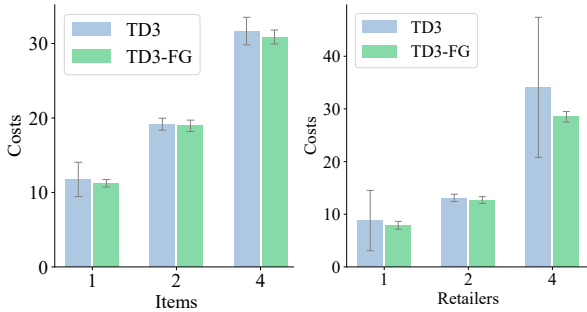

Figure 4: Multi-echelon.

Figure 5: Final results on multi-item and multi-echelon environments.

6 CONCLUSION

This paper addresses the challenge of sample scarcity for RL methods in lost-sales IC problems. We design a novel training framework integrating RLFG and IME to boost the sample efficiency of RL methods. We first specially tailored FG using general lost-sales properties, environment-specific heuristics, to generate side experiences to aid RL updates. We then conduct a theoretical analysis to demonstrate our method's effec-

tiveness with Q-learning as an example. The analysis shows that FG decreases the sample complexity by improving the update probabilities for all state-action pairs. Additionally, we design an intrinsic reward to fully utilize FG for lost-sales IC problems based on the theoretical insights. To validate the performance and generalization ability of our proposed method, we have done experiments on single-item, multi-item, and multi-echelon lost-sales IC environments. Experimental results show that our approach greatly enhances RL's sampling efficiency in IC problems, which is consistent with the theoretical analysis.

Acknowledgements

References

N. Alon, N. Cesa-Bianchi, O. Dekel, and T. Koren. Online learning with feedback graphs: Beyond ban-

dits. In *Conference on Learning Theory*, pages 23–35. PMLR, 2015.

K. J. Arrow, S. Karlin, H. E. Scarf, et al. *Studies in the mathematical theory of inventory and production*. Stanford University Press, 1958.

X. Bai, X. Chen, M. Li, and A. Stolyar. Asymptotic optimality of open-loop policies in lost-sales inventory models with stochastic lead times. *Available at SSRN 4362329*, 2023.

M. Bijvank and S. G. Johansen. Periodic review lost-sales inventory models with compound poisson demand and constant lead times of any length. *European Journal of Operational Research*, 220(1):106–114, 2012.

M. Bijvank and I. F. Vis. Lost-sales inventory theory: A review. *European Journal of Operational Research*, 215(1):1–13, 2011.

M. Bijvank, W. T. Huh, and G. Janakiraman. Lost-sales inventory systems. In *Research Handbook on Inventory ManagemenT*, pages 2–26. Edward Elgar Publishing, 2023.

R. N. Boute, J. Gijsbrechts, W. Van Jaarsveld, and N. Vanvuchelen. Deep reinforcement learning for inventory control: A roadmap. *European Journal of Operational Research*, 298(2):401–412, 2022.

B. Chen, J. Jiang, J. Zhang, and Z. Zhou. Learning to order for inventory systems with lost sales and uncertain supplies. *Management Science*, 2024.

C. Cuartas and J. Aguilar. Hybrid algorithm based on reinforcement learning for smart inventory management. *Journal of intelligent manufacturing*, 34(1): 123–149, 2023.

C. Dann, Y. Mansour, M. Mohri, A. Sekhari, and K. Sridharan. Reinforcement learning with feedback graphs. *Advances in Neural Information Processing Systems*, 33:16868–16878, 2020.

B. J. De Moor, J. Gijsbrechts, and R. N. Boute. Reward shaping to improve the performance of deep reinforcement learning in perishable inventory management. *European Journal of Operational Research*, 301(2):535–545, 2022.

H. Dehaybe, D. Catanzaro, and P. Chevalier. Deep reinforcement learning for inventory optimization with non-stationary uncertain demand. *European Journal of Operational Research*, 314(2):433–445, 2024.

S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

J. Gijsbrechts, R. N. Boute, J. A. Van Mieghem, and D. J. Zhang. Can deep reinforcement learning improve inventory management? performance on lost

sales, dual-sourcing, and multi-echelon problems. *Manufacturing & Service Operations Management*, 24(3):1349–1368, 2022.

D. A. Goldberg, D. A. Katz-Rogozhnikov, Y. Lu, M. Sharma, and M. S. Squillante. Asymptotic optimality of constant-order policies for lost sales inventory models with large lead times. *Mathematics of Operations Research*, 41(3):898–913, 2016.

D. A. Goldberg, M. I. Reiman, and Q. Wang. A survey of recent progress in the asymptotic analysis of inventory systems. *Production and Operations Management*, 30(6):1718–1750, 2021.

J. Han, M. Hu, and G. Shen. Deep neural newsvendor. *arXiv preprint arXiv:2309.13830*, 2023.

B. Hao, T. Lattimore, and C. Qin. Contextual information-directed sampling. In *International Conference on Machine Learning*, pages 8446–8464. PMLR, 2022.

P. Harsha, A. Jagmohan, J. Kalagnanam, B. Quanz, and D. Singhvi. Math programming based reinforcement learning for multi-echelon inventory management. *Available at SSRN 3901070*, 2021.

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

G. Janakiraman and R. O. Roundy. Lost-sales problems with stochastic lead times: Convexity results for base-stock policies. *Operations Research*, 52(5):795–803, 2004.

J. Jiang and Y. Ye. Achieving instance-dependent sample complexity for constrained markov decision process. *arXiv preprint arXiv:2402.16324*, 2024.

S. Karlin. Inventory models of the arrow-harris-marschak type with time lag. *Studies in the mathematical theory of inventory and production*, 1958.

I. Kaynov, M. van Knippenberg, V. Menkovski, A. van Breemen, and W. van Jaarsveld. Deep reinforcement learning for one-warehouse multi-retailer inventory management. *International Journal of Production Economics*, 267:109088, 2024.

M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

G. Li, Y. Wei, Y. Chi, Y. Gu, and Y. Chen. Sample complexity of asynchronous q-learning: Sharper analysis and variance reduction. *Advances in neural information processing systems*, 33:7031–7043, 2020.

G. Li, C. Cai, Y. Chen, Y. Wei, and Y. Chi. Is q-learning minimax optimal? a tight sample complexity analysis. *Operations Research*, 72(1):222–236, 2024.

T. Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

F. Liu, S. Buccapatnam, and N. Shroff. Information directed sampling for stochastic bandits with graph feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

S. Mannor and O. Shamir. From bandits to experts: On the value of side-observations. *Advances in Neural Information Processing Systems*, 24, 2011.

T. V. Marinov, M. Mohri, and J. Zimmert. Stochastic online learning with feedback graphs: Finite-time and asymptotic optimality. *Advances in Neural Information Processing Systems*, 35:24947–24959, 2022.

H. Meisheri, V. Baniwal, N. N. Sultana, H. Khadilkar, and B. Ravindran. Using reinforcement learning for a large variable-dimensional inventory management problem. In *Adaptive learning agents workshop at AAMAS*, pages 1–9, 2020.

H. Meisheri, N. N. Sultana, M. Baranwal, V. Baniwal, S. Nath, S. Verma, B. Ravindran, and H. Khadilkar. Scalable multi-product inventory control with lead time constraints using reinforcement learning. *Neural Computing and Applications*, 34(3):1735–1757, 2022.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

T. E. Morton. Bounds on the solution of the lagged optimal inventory equation with no demand backlogging and proportional costs. *SIAM review*, 11(4):572–596, 1969.

T. E. Morton. The near-myopic nature of the lagged-proportional-cost inventory problem with lost sales. *Operations Research*, 19(7):1708–1716, 1971.

R. Nian, J. Liu, and B. Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers & Chemical Engineering*, 139:106886, 2020.

N. Nikolov, J. Kirschner, F. Berkenkamp, and A. Krause. Information-directed exploration for deep reinforcement learning. *arXiv preprint arXiv:1812.07544*, 2018.

A. Oroojlooyjadid, M. Nazari, L. V. Snyder, and M. Takáč. A deep q-network for the beer game: Deep reinforcement learning for inventory optimization. *Manufacturing & Service Operations Management*, 24(1):285–304, 2022.

H. Park, D. G. Choi, and D. Min. Adaptive inventory replenishment using structured reinforcement learning by exploiting a policy structure. *International Journal of Production Economics*, 266:109029, 2023.

M. I. Reiman. A new and simple policy for the continuous review lost sales inventory model. *Unpublished manuscript*, 2004.

M. Selukar, P. Jain, and T. Kumar. Inventory control of multiple perishable goods using deep reinforcement learning for sustainable environment. *Sustainable Energy Technologies and Assessments*, 52:102038, 2022.

F. Stranieri and F. Stella. Comparing deep reinforcement learning algorithms in two-echelon supply chains, 2023.

F. Stranieri, E. Fadda, and F. Stella. Combining deep reinforcement learning and multi-stage stochastic programming to address the supply chain inventory management problem. *International Journal of Production Economics*, 268:109099, 2024.

N. N. Sultana, H. Meisheri, V. Baniwal, S. Nath, B. Ravindran, and H. Khadilkar. Reinforcement learning for multi-product multi-node inventory management in supply chains. *arXiv preprint arXiv:2006.04037*, 2020.

A. Tossou, C. Dimitrakakis, and D. Dubhashi. Thompson sampling for stochastic bandits with graph feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

L. Van Hezewijk, N. Dellaert, T. Van Woensel, and N. Gademann. Using the proximal policy optimisation algorithm for solving the stochastic capacitated lot sizing problem. *International Journal of Production Research*, 61(6):1955–1978, 2023.

G. Wu, M. Á. de Carvalho Servia, and M. Mowbray. Distributional reinforcement learning for inventory management in multi-echelon supply chains. *Digital Chemical Engineering*, 6:100073, 2023.

Y. Xie, W. Ma, and L. Xin. Vc theory for inventory policies. *arXiv preprint arXiv:2404.11509*, 2024.

L. Xin. Understanding the performance of capped base-stock policies in lost-sales inventory models. *Operations Research*, 69(1):61–70, 2021.

D. Yarats, I. Kostrikov, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=GY6-6sTvGaf`.

H. Zhang, C. Xiao, H. Wang, J. Jin, bo xu, and M. Müller. Replay memory as an empirical MDP:

Combining conservative estimation with experience replay. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=SjzFVSJUt8S`.

zhengyao jiang, T. Zhang, R. Kirk, T. Rocktäschel, and E. Grefenstette. Graph backup: Data efficient backup exploiting markovian data. In *Deep RL Workshop NeurIPS 2021*, 2021. URL `https://openreview.net/forum?id=0UQqmPGuL4n`.

P. Zipkin. Old and new methods for lost-sales inventory systems. *Operations research*, 56(5):1256–1263, 2008a.

P. Zipkin. On the structure of lost-sales inventory models. *Operations research*, 56(4):937–944, 2008b.

Checklist

1. For all models and algorithms presented, check if you include:

   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. **[Yes]**

   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. **[Yes]**

   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. **[Yes] The code is provided in Abstract.**

2. For any theoretical claim, check if you include:

   (a) Statements of the full set of assumptions of all theoretical results. **[Yes]**

   (b) Complete proofs of all theoretical results. **[Yes]**

   (c) Clear explanations of any assumptions. **[Yes]**

3. For all figures and tables that present empirical results, check if you include:

   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). **[Yes]**

   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). **[Yes]**

   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). **[Yes]**

   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). **[Yes]**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   Justification: We referred to the environmental model of Zipkin (2008a) and the code of Gijsbrechts et al. (2022). Both of them are cited in the text and reference.

   (a) Citations of the creator If your work uses existing assets. **[Yes]**

   (b) The license information of the assets, if applicable. **[Yes]**

   (c) New assets either in the supplemental material or as a URL, if applicable. **[Not Applicable]**

   (d) Information about consent from data providers/curators. **[Not Applicable]**

   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. **[Not Applicable]**

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. **[Not Applicable]**

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. **[Not Applicable]**

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. **[Not Applicable]**

A   EXTENDED RELATED WORK

A.1   Inventory Control Problem

The inventory control (IC) problem involves determining the optimal quantity of inventory to order to minimize costs while maintaining sufficient stock levels to meet customer demand. Based on different assumptions about the behavior of the customer, the IC problem can be divided into backlogging IC and lost-sales IC problems (Chen et al., 2024). The backlogging IC problem assumes that when the demand cannot be met due to insufficient inventory, the request of customers is accepted but delayed until inventory is replenished. Arrow et al. (1958) has proven that the base-stock policy, which aims to keep the sum of inventory level and upcoming orders constant, is optimal for single-source backlogging IC with constant lead time. Compared with backlogging IC, the lost-sales IC problem is more complex and relevant (Bijvank and Vis, 2011). the lost-sale IC problem assumes that when demand cannot be met due to insufficient inventory, the customer's request is lost and the exceeding request is unobservable, such as e-commerce. The base-stock policy can only be optimal when the cost of the lost-sales penalty is high. This motivates researchers to find better methods for the lost-sales IC problem.

A.2   Heuristic Methods in Lost-sales Inventory Control

The lost-sales IC problem is first simply studied by Karlin (1958), which assumes the lead time to place orders is one. Karlin (1958) proves that base-stock policy is not optimal because the inventory availability in future periods cannot be characterized by the inventory level and order quantity. Morton (1969) extends the former analysis to any positive and integral-value lead time and provides the upper and lower bounds of the optimal policy. Furthermore, Zipkin (2008b) generalizes to any lead times from the aspect of L-natural-convexity, and Janakiraman and Roundy (2004) studies the case when lead time is stochastic. The base-stock policy is sensitive to the demand due to its design. The opposite one is the constant order policy, whose decision has no relationship with the demand (Bijvank et al., 2023). As the lead time goes to infinity, Goldberg et al. (2016) proves that the constant-order policy can be asymptotically optimal and Reiman (2004) proves that the constant-order policy can be better than the base-stock policy. The contrary properties of these two methods motivate a better idea that combines both advantages (Bijvank and Johansen, 2012). Xin (2021) improves this idea to a new method named capped base-stock policy.

The above-introduced methods need to search for the best parameter for specific settings to perform well. Besides them, there is another series of heuristic method, also called approximate dynamic programming (APD). These methods assume known demand distribution rather than parameter searching. Morton (1971) proposes the Myopic method, which aims to minimize the expected cost when the placed order arrives. This method can be extended to the Myopic-T method, which considers the expected cost for T time steps.

A.3   Reinforcement Learning Methods in Inventory Control

The heuristic method needs either a parameter search or an assumption about the demand distribution, which are not general enough. This motivates the studies about applying reinforcement learning (RL) to lost-sales IC problems. RL has gained significant attention as a powerful data-driven technique for solving complex sequential decision-making problems (Han et al., 2023; Jiang and Ye, 2024; Xie et al., 2024). RL offers several advantages for addressing the challenges of IC problems. Firstly, RL allows for the discovery of optimal policies without relying on strong problem-specific assumptions, enabling more generalizable solutions (Nian et al., 2020). Secondly, combined with the deep neural network, Deep RL (DRL) can handle large state and action spaces, making it suitable for problems with high-dimensional variables (Dehaybe et al., 2024). Initially, the work mainly focuses on verifying the feasibility of RL in IC problems. Oroojlooyjadid et al. (2022) showcases the ability of a Deep Q-Network (DQN) to discover solutions that are close to optimal for the widely recognized beer distribution game. Gijsbrechts et al. (2022) introduces the A3C algorithm into IC problems and show that A3C can achieve acceptable performance but not better heuristic methods. Stranieri and Stella (2023) benchmarks various DRL methods such as A3C, PPO, and vanilla policy gradient (VPG) in IC problems. Recently, researchers tend to study how DRL can solve different situations of IC problems such as, non-stationary uncertain demand (Dehaybe et al., 2024; Park et al., 2023), multi-product (Sultana et al., 2020; Selukar et al., 2022), variable kinds of products (Meisheri et al., 2020, 2022), multi-echelon supply chains (Wu et al., 2023; Stranieri et al., 2024), one-warehouse multi-retailer (Kaynov et al., 2024), stochastic capacitated lot sizing problem (Van Hezewijk et al., 2023).

However, Two fundamental issues have not been resolved. First, the final performance of DRL is not optimal, sometimes even worse than heuristic methods. To address this problem, some papers (Cuartas and Aguilar, 2023; Stranieri et al., 2024; Harsha et al., 2021) explore the potential to combine RL algorithms with ex-

isting other methods rather than directly applying RL to IC problems. The other problem is the low sample efficiency nature of existing RL methods, which restricts further application to real-world IC problems (Boute et al., 2022; De Moor et al., 2022), especially when obtaining experiences is expensive or time-consuming. Besides, this sample inefficiency problem is enlarged in lost-sales IC because of censored demands (Chen et al., 2024), which refers to the phenomenon when the customer's real demand is unobservable due to insufficient inventory. Typically, if the order is placed daily, generating four hundred experiences needs over a year and part of them may be censored making them too hard to update RL policy. De Moor et al. (2022) tries to alleviate this problem by incorporating heuristic knowledge, such as base-stock policy, into DQN with reward-shaping. Although this method can improve the sample efficiency, it still relies on specific model heuristics, making it hard to generalize to different scenarios. Overall, Resolving the low sample efficiency of RL without strong heuristics is crucial in solving IC (especially lost-sales IC) problems.

A.4 Feedback Graph and its Application

Mannor and Shamir (2011) first proposes the feedback graph (FG) idea as a method to reduce the regret bound of the bandit problem when side observations can be obtained assuming the decision maker can also know the situations when other actions are taken besides the chosen action. Alon et al. (2015) extends the analysis of FG on bandit problems beyond the learning problems. Tossou et al. (2017) gives the first analysis of Thompson sampling for bandits with FG based on information theory. Building upon these foundations, Dann et al. (2020) combines FG with RL and shows how FG can reduce the regret bound and sample complexity of model-based RL algorithms. However, most existing work, such as Liu et al. (2018), Hao et al. (2022), and Marinov et al. (2022), mainly focus on the analysis of FG in some bandits problems. Seldom work tries to apply FG to real-world problems since side observations and the structure of FG are hard to define in applications.

A.5 Data Augmentation Methods

Data augmentation methods, similar to RLFG, aim to improve the sample efficiency of RL by more effectively utilizing available information. Specifically, Laskin et al. (2020) and Yarats et al. (2021) demonstrate how transformations like cropping, translating, rotating, or perturbing inputs can improve RL algorithms for visual tasks. These methods augment current states by applying transformations and use the augmented data as current states. Furthermore, some

later research efforts such as RM-MDP (Zhang et al., 2023) and Graph Backup (zhengyao jiang et al., 2021) leverage replay memory to extract additional information. In particular, RM-MDP, inspired by model-based RL, constructs an offline RL framework by reusing replay memory, while Graph Backup improves multi-step backups by constructing graph-based backups from the replay buffer. If RLFG is viewed as a data augmentation technique, it stands apart by focusing on generating unobserved state-action pairs from observed ones. Unlike traditional data augmentation, RLFG operates at the MDP level, where the augmented "side experiences" correspond to unobserved transitions within the MDP. This approach leverages the structure of the MDP itself, setting it apart from methods that augment data through transformations or trajectory-level modifications.

B RAINBOW-FG ALGORITHM

Algorithm 3 shows the example of utilizing FG, called Rainbow-FG.

Algorithm 3 Rainbow-FG Algorithm.

1: Initialize the distributional Q-network Q with parameter θ, the target Q-network Q' with parameter ϕ, the replay buffer $\mathcal{R}$, and the side replay buffer $\mathcal{R}^s$
2: **for** episode $= 1, ..., M$ **do**
3:     Initialize the environment state s_0
4:     **for** $t = 0, ..., T$ **do**
5:         Get the order $a_t = \begin{cases} \text{argmax}_a\ Q(\boldsymbol{s}_t, a), & w.p\ \ 1 - \epsilon \\ \text{a random action}, & w.p\ \ \epsilon \end{cases}$
6:         Execute action a_t and get next state $\boldsymbol{s}_{t+1}$ and reward r_t
7:         Get the observed demand d_t^o
8:         Get the side experiences $\{(\hat{\boldsymbol{s}}_t, \hat{a}_t, \hat{r}_t, \hat{s}_{t+1}, d_t^o)\} \rightarrow$ Algorithm 1
9:         $\mathcal{R} = \mathcal{R} \cup (\boldsymbol{s}_t, a_t, r_t, \boldsymbol{s}_{t+1}, d_t^o)$ and $\mathcal{R}^s = \mathcal{R}^s \cup \{(\hat{\boldsymbol{s}}_t, \hat{a}_t, \hat{r}_t, \hat{s}_{t+1}, d_t^o)\}$
10:         **Model Update:**
11:         Randomly sample a batch of experiences $\{(\boldsymbol{s}, a, r, \boldsymbol{s}', d^o)\}$ from $\mathcal{R} \cup \mathcal{R}^s$
12:         Get all side experiences $\{(\hat{\boldsymbol{s}}, \hat{a}, \hat{r}, \hat{\boldsymbol{s}}', d^o)\}$ for each experience in $\{(\boldsymbol{s}, a, r, \boldsymbol{s}', d^o)\} \rightarrow$ Algorithm 1
13:         Get the intrinsic reward r^{in} of this batch of experiences $\rightarrow$ Algorithm 2
14:         Get the final reward $r = (1 - \beta) \times r + \beta \times r^{in}$
15:         Update parameters

C ANALYSIS BASED ON THE PROPERTY OF GRAPH

Based on the feedback graph $\mathcal{G}$, Dann et al. (2020) defines three concepts to measure the sample complexity. ω, α, and ζ measure the minimum vertices to be sampled to observe the whole $\mathcal{G}$. We will talk about ω, α, and ζ for the IC environment in section 4.2.

Mas-number(ω): The maximum size of $\mathcal{V}' \subseteq \mathcal{V}$ forming an acyclic subgraph of $\mathcal{G}$ is the mas-number.

Independence number(α): The size of the largest $\mathcal{V}' \subseteq \mathcal{V}$ with no edge in $\mathcal{G}$ within $\mathcal{V}'$ is the independence number.

Domination number(ζ): A set of vertices $\mathcal{V}' \subseteq \mathcal{V}$ is a dominating set if there always $\exists v' \in \mathcal{V}'$ and $\forall v \in \mathcal{V}$ such that $v' \to v$. The smallest size of $\mathcal{V}'$ is called the domination number ζ.

For any feedback graph $\mathcal{G}$, the inequality (8) is always satisfied, where $|\mathcal{S}|$ is the size of the state set and $|\mathcal{A}|$ is the size of the action set. The mas-number ω and independence number α quantify the worst-case connectivity by measuring the maximum number of distinct vertices an algorithm can traverse before encountering a repeated vertex. On the other hand, the domination number represents a best-case scenario and indicates the minimum number of vertices that an algorithm must visit to observe every vertex. Besides, Dann et al. (2020) also shows that with feedback graph $\mathcal{G}$, the regret bound/sample complexity of a model-based RL algorithm can be reduced from $|\mathcal{S}||\mathcal{A}|$ scale to ω or even ζ scale.

$$\zeta \leq \alpha \leq \omega \leq |\mathcal{V}| = |\mathcal{S}| \times |\mathcal{A}|. \qquad (8)$$

In this paragraph, we will simply show how sample complexity is reduced based on ω, α, and ζ. If we define all state-action pairs as the nodes of the feedback graph $\mathcal{G}$, $\mathcal{G}$ can be divided into two parts $\mathcal{G}_1$ and $\mathcal{G}_2$, where $\mathcal{G}_1$ is a connected graph, $\mathcal{G}_2$ is a graph without any edge, and $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$.

For the uncensored case, $\mathcal{G} = \mathcal{G}_1$ is a complete graph and $\mathcal{G}_2 = \emptyset$. The complete feedback graph has a good property which is $\omega = \alpha = \zeta = 1 \ll |\mathcal{S}||\mathcal{A}|$. For the censored case, $\mathcal{G}$ consists of $\mathcal{G}_1$ for $\hat{y}_t \leq y_t$ and $\mathcal{G}_2$ for $\hat{y}_t > y_t$. Thus the feedback graph's property should be $\alpha = \zeta = (y^{\max} - y_t) \times (a^{\max})^L + 1(y_t \neq 0)$ and $\omega = (y^{\max} - y_t) \times (a^{\max})^L + y_t$. We can obtain that $\alpha = \zeta \leq \omega \leq |\mathcal{S}||\mathcal{A}|$, where $\alpha = \zeta = \omega$ holds iff $y_t = 0 \& 1$ and $\omega = |\mathcal{S}||\mathcal{A}|$ holds iff $y_t = 0$. Overall, sample complexity after using FG can be reduced.

D THEORETICAL ANALYSIS

Let us analyze the sample complexity and update probability scenario by scenario from the simplest case to real case. Let us start with **scenario 2** since **scenario 1** is analyzed in section 4.3.

Scenario 2: Consider a graph $\mathcal{G}$ with all state-action pairs as the nodes. Assume all nodes, once one node is sampled, all of these nodes can be sampled and updated at the same time. Thus G can be regarded as $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$, where $\mathcal{G}_1$ is a complete graph and $\mathcal{G}_2 = \emptyset$. This case is the same as Synchronous Q-Learning in Li et al. (2024).

Lemma 2: The sample complexity of the Q-learning with feedback graph under scenario 2 is $\tilde{O}(\frac{1}{(1-\gamma)^4\epsilon^2})$ with learning rate being $(1-\gamma)^3\epsilon^2$.

Scenario 3: Consider a graph $\mathcal{G}$ with all state-action pairs as the nodes. Assume for some nodes, once one node is sampled, all of these nodes can be sampled and updated at the same time. For other nodes, when one node is sampled, only itself can be updated. Thus $\mathcal{G}$ can be regarded as $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$, where $\mathcal{G}_1$ is a complete graph and $\mathcal{G}_2$ consists of nodes without any edge.

Lemma 3: The sample complexity of the Q-learning with FG under scenario 3 is $\tilde{O}(\frac{1}{\tilde{\mu}_{\min}(1-\gamma)^5\epsilon^2} + \frac{t_{\mix}}{\tilde{\mu}_{\min}(1-\gamma)})$, where $\tilde{\mu}_{\min} = \min[\min_{(\boldsymbol{s},a)\in G_2} \mu(\boldsymbol{s},a), \sum_{(\boldsymbol{s},a)\in\mathcal{G}_1} \mu(\boldsymbol{s},a)]$

Scenario 1 indicates the IC environment without feedback graph and scenario 2 indicates the uncensored case of the IC environment with feedback graph. As for the censored case of the IC environment, we simplify it in scenario 3 by assuming $\mathcal{G}_1$ is a complete graph. We can see that the sample complexity order is $\tilde{O}(\frac{1}{\mu_{\min}(1-\gamma)^5\epsilon^2} + \frac{t_{\mix}}{\mu_{\min}(1-\gamma)}) \geq \tilde{O}(\frac{1}{\tilde{\mu}_{\min}(1-\gamma)^5\epsilon^2} + \frac{t_{\mix}}{\tilde{\mu}_{\min}(1-\gamma)}) > \tilde{O}(\frac{1}{(1-\gamma)^4\epsilon^2})$, since $\min_{(\boldsymbol{s},a)\in\mathcal{G}} \mu(\boldsymbol{s},a) < \min[\min_{(\boldsymbol{s},a)\in G_2} \mu(\boldsymbol{s},a), \sum_{(\boldsymbol{s},a)\in\mathcal{G}_1} \mu(\boldsymbol{s},a)] < 1$, where 1 indicates $\mu_{\min} = 1$ in scenario 2.

Now, Let us loosen the assumption that $\mathcal{G}_1$ is a complete graph in scenario 3.

Scenario 4: Consider a graph $G = G_1 \cup G_2$ with all state-action pairs as the nodes. We define $v_i < v_j$ and $s_i < s_j$ if $s_i[0] < s_j[0]$. For nodes in G_1, once one node v_i is sampled, $\forall v_j < v_i$ can be sampled and updated at the same time. For nodes in G_2, when one node is sampled, only itself can be updated. Thus G can be regarded as $G = G_1 \cup G_2$, where G_1 is a connected graph and $G_2 = \emptyset$.

Lemma 4: The sample complexity of the Q-learning with feedback graph under Assumption 4 is $\tilde{O}(\frac{1}{\tilde{\mu}'_{min}(1-\gamma)^5\epsilon^2} + \frac{t_{mix}}{\tilde{\mu}'_{min}(1-\gamma)})$, where $\hat{\mu}'_{min} =$

$\min[\min_{(s,a)\in G_2}\mu(s,a),\sum_{\{s,a|s\in G_1;\forall \hat{s}\in G_1,s>\hat{s}\}}\mu(s,a)]$.

Based on the above scenarios and lemmas, let us consider the lost-sales IC environment with constant demand d.

Scenario 5: Consider a graph $\mathcal{G}$ with all state-action pairs as the nodes. Assume for nodes with $y \geq d$, once one node is sampled, all of these nodes can be sampled and updated simultaneously. For nodes with $y < d$, only nodes with $y' \leq y$ can be sampled and updated simultaneously. Thus $\mathcal{G}$ can be regarded as $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$.

Lemma 5: The sample complexity of the Q-learning with feedback graph under scenario 5 is $\tilde{O}(\frac{1}{\tilde{\mu}_{\min}(1-\gamma)^5 \epsilon^2} + \frac{t_{\text{mix}}}{\tilde{\mu}_{\min}(1-\gamma)})$, where $\tilde{\mu}_{\min} = \sum_{(s,a)\in\mathcal{G};y\geq d}\mu(s,a)$.

Proof:

For $\{(s,a)|(s,a) \in \mathcal{G}; y \geq d\}$, we have $\tilde{\mu}_{\min}^{y \geq d} = \sum_{(s,a)\in\mathcal{G};y\geq d}\mu(s,a)$.

For $\{(s,a)|(s,a) \in \mathcal{G}; y < d\}$, we have $\tilde{\mu}_{\min}^{y < d} = \sum_{y=d}\mu(s,a) + \mu_{\min}^{y \geq d}$.

Thus we have $\tilde{\mu}_{\min} = \tilde{\mu}_{\min}^{y \geq d}$.

If $\mu_{\min}$ appears in $\{(s,a)|(s,a) \in \mathcal{G}; y \geq d\}$, we have $\tilde{\mu}_{\min} = \tilde{\mu}_{\min}^{y \geq d} = \sum_{(s,a)\in\mathcal{G};y\geq d}\mu(s,a) > (y^{\max} - d)|A|^L \min_{(s,a)\in\mathcal{G};y\geq d}\mu(s,a) = (y^{\max} - d)|A|^L\mu_{\min}$.

If $\mu_{\min}$ appears in $\{(s,a)|(s,a) \in \mathcal{G}; y < d\}$, which means $\mu_{\min} < \min_{(s,a)\in\mathcal{G};y\geq d}\mu(s,a)$, we have $\tilde{\mu}_{\min} = \tilde{\mu}_{\min}^{y \geq d} = \sum_{(s,a)\in\mathcal{G};y\geq d}\mu(s,a) > (y^{\max} - d)|A|^L \min_{(s,a)\in\mathcal{G};y\geq d}\mu(s,a) > (y^{\max} - d)|A|^L\mu_{\min}$.

Thus $\tilde{\mu}_{\min}$ under scenario 5 improves at least $(y^{\max} - d)|A|^L$ times than that under scenario 1.

Proof done.

Now we lossen the assumption of the constant demand d to $d_t \sim P_d(d)$.

Scenario 6: Consider a graph $\mathcal{G}$ with all state-action pairs as the nodes. Consider all nodes satisfying $y \geq d_t$, if one is sampled by the policy, all can be sampled by the FG module and updated simultaneously. For nodes with $y < d_t$, only nodes with $y' \leq y$ can be sampled by the FG module and updated simultaneously. Thus $\mathcal{G}$ can be regarded as $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$.

Proof of Theorem 1:

For each $(\boldsymbol{s}, a)$ and each possible value of d, we have

$$\tilde{\mu}^{y\geq d}(\boldsymbol{s},a|d) = \sum_{\substack{(\hat{\boldsymbol{s}},\hat{a})\in\mathcal{G} \\ \hat{y}\geq d}}\mu(\hat{\boldsymbol{s}},\hat{a}|d) \geq \mu^{y\geq d}(\boldsymbol{s},a|d), \tag{9}$$

$$\{(\boldsymbol{s},a)|(\boldsymbol{s},a) \in \mathcal{G}; y \geq d\}.$$

$$\tilde{\mu}^{y<d}(\boldsymbol{s},a|d) = \sum_{\substack{(\hat{\boldsymbol{s}},\hat{a})\in\mathcal{G} \\ y\leq\hat{y}\leq d}}\mu(\hat{\boldsymbol{s}},\hat{a}|d) + \sum_{\substack{(\hat{\boldsymbol{s}},\hat{a})\in\mathcal{G} \\ \hat{y}\geq d}}\mu(\hat{\boldsymbol{s}},\hat{a}|d)$$

$$\geq \mu^{y\leq d}(\boldsymbol{s},a|d), \ \{(\boldsymbol{s},a)|(\boldsymbol{s},a) \in \mathcal{G}; y < d\}. \tag{10}$$

Thus we have

$$\tilde{\mu}(\boldsymbol{s},a) = E_{d\sim P_d}[\tilde{\mu}(\boldsymbol{s},a|d)] \geq E_{d\sim P_d}[\mu(\boldsymbol{s},a|d)] = \mu(\boldsymbol{s},a). \tag{11}$$

Then we formulate $\tilde{\mu}(\boldsymbol{s},a)$ in details:

$$\tilde{\mu}(\boldsymbol{s},a) = E_{d\sim P_d}[\tilde{\mu}(\boldsymbol{s},a|d)]$$

$$= \sum_{d=y}^{d^{\max}}P_d(d)\sum_{\substack{(\hat{\boldsymbol{s}},\hat{a})\in\mathcal{G} \\ \hat{y}\geq d}}\mu(\hat{\boldsymbol{s}},\hat{a}|d)$$

$$+ \sum_{d=0}^{y-1}P_d(d)\sum_{\substack{(\hat{\boldsymbol{s}},\hat{a})\in\mathcal{G} \\ y\leq\hat{y}\leq d}}\mu(\hat{\boldsymbol{s}},\hat{a}|d)$$

$$+ \sum_{\substack{(\hat{\boldsymbol{s}},\hat{a})\in\mathcal{G} \\ \hat{y}\geq d}}\mu(\hat{\boldsymbol{s}},\hat{a}|d) \tag{12}$$

$$= \sum_{d=0}^{d^{\max}}P_d(d)\sum_{\substack{(\hat{\boldsymbol{s}},\hat{a})\in\mathcal{G} \\ \hat{y}\geq d}}\mu(\hat{\boldsymbol{s}},\hat{a}|d)$$

$$+ \sum_{d=0}^{y-1}P_d(d)\sum_{\substack{(\hat{\boldsymbol{s}},\hat{a})\in\mathcal{G} \\ y\leq\hat{y}\leq d}}\mu(\hat{\boldsymbol{s}},\hat{a}|d).$$

Proof done.

E PARAMETERS OF EXPERIMENTS

Table 3 shows the parameters for Rainbow-FG and TD3-FG. Table 4 shows the parameters for the single-item environment. As for multi-item environment, we set $P = [4, 9]$ and $L = [4, 3]$ for 2 items, $P = [4, 9, 19]$ and $L = [4, 3, 2]$ for 3 items, $P = [4, 9, 4, 9, 19]$ and $L = [4, 4, 3, 3, 2]$ for 5 items. As for multi-echelon environment, we set the warehouse with $P = 4$ and $L = 4$; $P = 4$ and $L = 4$ for 1 retailer, $P = [4, 9]$ and $L = [4, 4]$ for 2 retailers, $P = [4, 9, 4, 9]$ and $L = [4, 4, 3, 3]$ for 4 retailers. Other parameters are kept the same as the single-item environment.

F COMPUTE RESOURCES

Experiments are carried out on Intel (R) Xeon (R) Platinum 8375C CPU @ 2.90GHz and NVIDIA GeForce RTX 3080 GPUs.

Table 3: Parameters of Rainbow-FG and TD3-FG.

Parameter	Value	Parameter	Value
Training Episodes	100	Training Steps/Episode	1000
Testing Frequency	1	Testing Steps/Episode	400
Batch Size	128	Batch Size for FG	256
Replay Buffer Size	12000	Replay Buffer Size for FG	192000
ϵ	0.1	γ	0.995
lr	1e-4	Target Update Frequency	100
$V^{\min}$	-200	$V^{\max}$	0
N^{atom}	51	Hidden Layer Size	512
Intrinsic Reward Weight	0.01	Intrinsic Reward Discount Factor	0.9

Table 4: Parameters of the environment. L and p are two important parameters where L reflects the dimensionality of the state and p reflects the seriousness of lost sales.

Parameter	L	p	$y^{\max}$	$a^{\max}$	c	h	d^m	$d^{\max}$
Value	4	4	100	20	0	1	5	20

The computational complexity is given as $O(T \cdot \prod_{i=0}^{m} S_i A_i)$, where it scales linearly with the product of state and action size of each item/echelon $i = 1, \ldots, m$. The time cost increases linearly on the feedback graph size and the lead time. The running time for each experiment is within 2 hours for RLFG without intrinsic rewards and 3 hours with intrinsic rewards.

G MORE EXPERIMENTS FOR FG

G.1 Sample Efficiency Comparison for Intrinsic Reward

This section investigates the effect of intrinsic reward designed for the IC problem. Figure 6 illustrates the learning process of Rainbow-FG with and without the intrinsic reward, referred to as Rainbow-FG w/ inr and Rainbow-FG w/o inr, respectively. The results demonstrate that the designed intrinsic reward significantly improves sample efficiency during the initial stages of training in the environment with $p = 4, L = 4$, and $d^m = 5$. However, as the learning process reaches the constant order level, the intrinsic reward does not exhibit a substantial impact. We attribute it to the simplicity of the environment, which weakens the effect of the intrinsic reward. To further evaluate its effectiveness, we conduct experiments in more complex settings with $p = 19, L = 8$, and varying values of $d^m = (5, 10, 15)$. In these environments, we observe a clear effect of the intrinsic reward during the whole training process. Moreover, Rainbow-FG w/ inr demonstrates a more

stable training process compared to Rainbow-FG w/o inr, achieving slightly lower cost.

H HYPERPARAMETER ANALYSIS

H.1 Sample Efficiency under Different Exploration Level

Figure 7 shows the learning process of Rainbow and Rainbow-FG with different exploration parameters in more settings. The learning processes in these settings show similar properties in section 5.2.2.

H.2 Feedback Graph Size

Theoretically, we should get enough side experiences by considering every term of S and A to get the largest size of G_1. However, in practice, the time or resources may be limited so that only part of the side information can be obtained. Based on this question, this section focuses on the effect of different sizes of the feedback graph.

The default setting only constructs the feedback graph considering the current inventory y_t, which is the first term of s_t, and action a_t. Each comparison group adds one following term of s_t when constructing the feedback graph. Figure 8 illustrates the learning process of Rainbow-FG with different sizes of FG. The result shows that the sample efficiency is more sensitive to the size of FG at the initial stage than at the final stage. As the size of FG increases, the improvement of the sample efficiency mainly occurs at the initial stage. At the final stage, Rainbow-FG w/ s[0:1] & a and s[0:4] & a first reach the final level and then follows Rainbow-FG with s[0:3] & a.

H.3 Intrinsic Reward Weight

To test the sensitivity of our intrinsic reward design, we test our method with different intrinsic reward weights. Figure 9 shows the detailed learning process. A larger intrinsic reward weight tends to have higher sample effi-

(a) p=4, L=4, d^m=5 (b) p=19, L=8, d^m=5
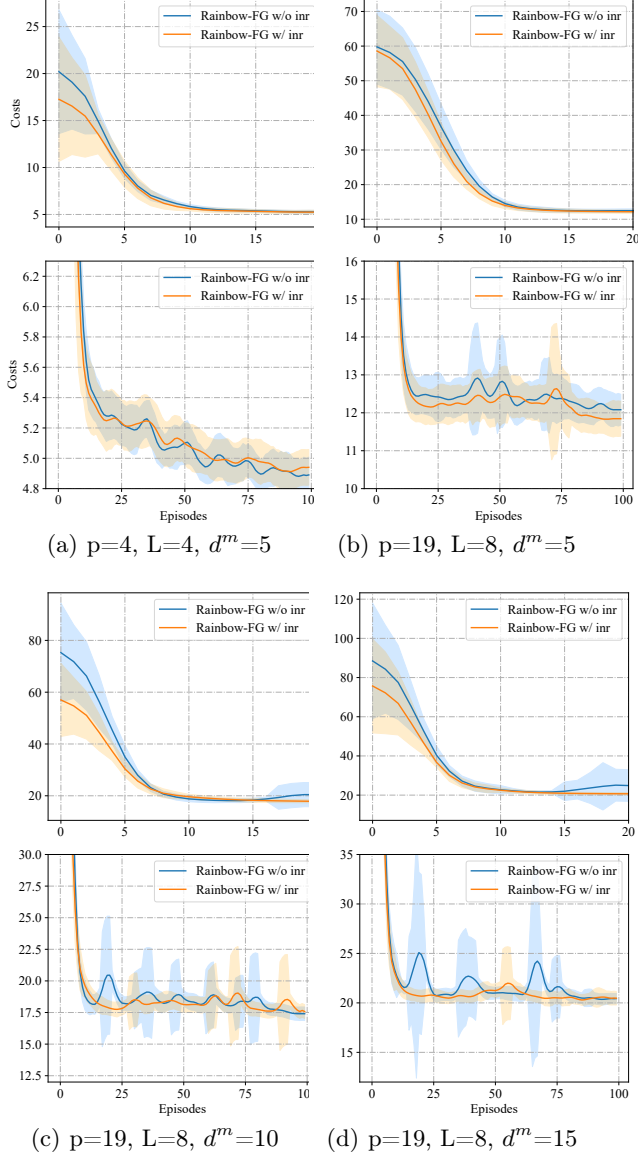

(c) p=19, L=8, d^m=10 (d) p=19, L=8, d^m=15

Figure 6: Learning process of Rainbow-FG with and without intrinsic reward. The two rows have the same meaning with Figure 2.

ciency during the initial stages of training. This result demonstrates the benefits of improving sample efficiency for the intrinsic reward method. However, large intrinsic reward weights can affect the performance of the final stage. The main reason is that adding the intrinsic reward to the extrinsic reward changes the original objective to be optimized.
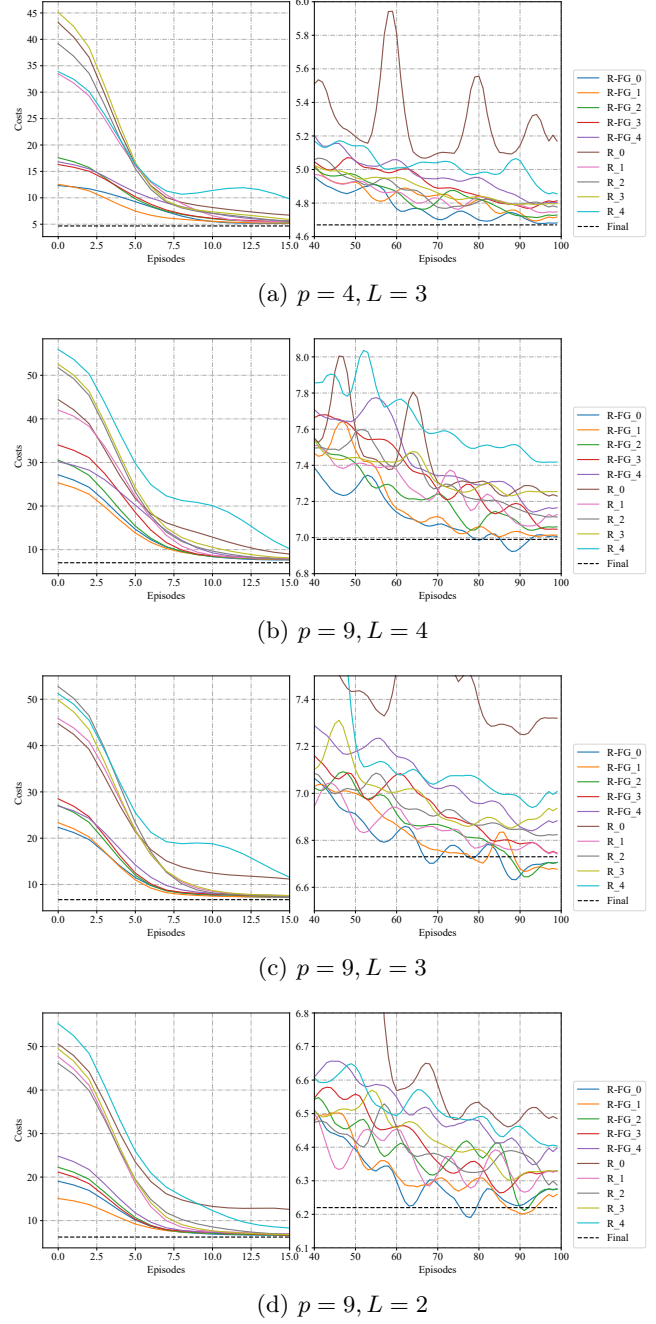

(a) $p = 4, L = 3$

(b) $p = 9, L = 4$

(c) $p = 9, L = 3$

(d) $p = 9, L = 2$

Figure 7: Learning process of Rainbow and Rainbow-FG under the IC environment. "R_x" denotes Rainbow with $\epsilon = 0.x$ and "R-FG_x" denotes Rainbow-FG with $\epsilon = 0.x$. "Final" denotes the optimal result of Rainbow-FG. The first column is the initial-stage view and the second column is the near-convergence view of the learning process.
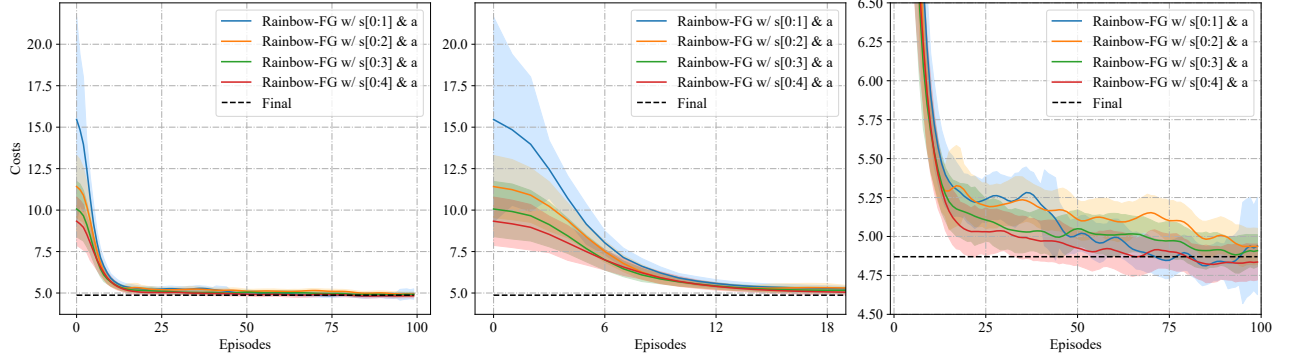
Figure 8: Learning process of Rainbow-FG with different feedback graph size. The first column is the full view of the learning process. The second and third columns are different partial views of the first column.
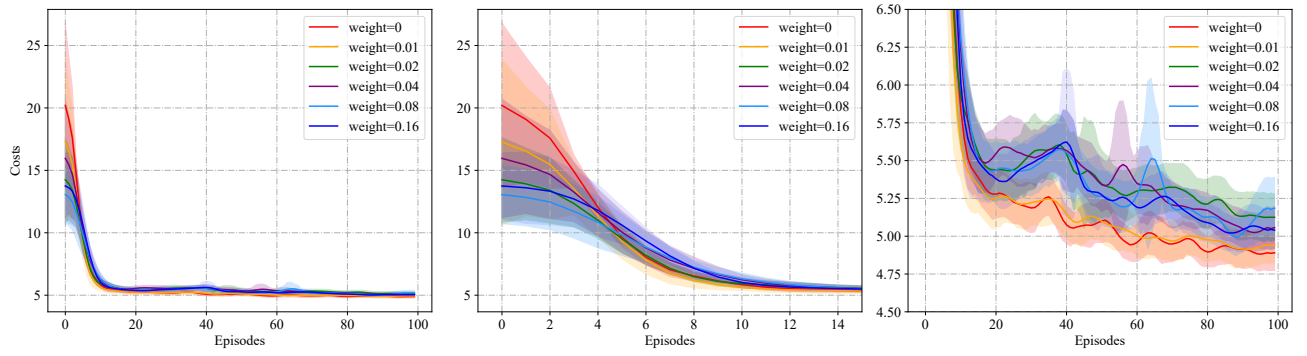

Figure 9: Learning process of Rainbow-FG with different intrinsic reward weights. The intrinsic reward weight is multiplied by 2 each time. The first column is the full view of the learning process. The second and third columns are different partial views of the first column.