Transportation Research Part E 187 (2024) 103576

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

A hybrid deep reinforcement learning approach for a proactive transshipment of fresh food in the online–offline channel system ☆

Junhyeok Lee [a], Youngchul Shin [b], Ilkyeong Moon [c,d,*]

[a] Memory Technology Innovation Team, Samsung Electronics, Saumsungjeonja-ro 1, Hwaseong-si, Gyeonggi-do, 18448, Republic of Korea
[b] Department of Industrial Engineering, Ajou University, World cup-ro 206, Yeongtong-gu, Suwon-si, Gyeonggi-do, 16499, Republic of Korea
[c] Department of Industrial Engineering, Seoul National University, Gwanak-ro 1, Gwanak-gu, Seoul 08826, Republic of Korea
[d] Institute of Engineering Research, Seoul National University, Gwanak-ro 1, Gwanak-gu, Seoul 08826, Republic of Korea

ARTICLE INFO

Keywords:
Perishable inventory management
Proactive transshipment
Online–offline channel system
Deep reinforcement learning
Soft actor–critic

ABSTRACT

To reduce the waste of fresh foods, one of the e-commerce companies in South Korea utilizes lateral transshipment in the network of online platforms and offline shops, which is called the online–offline channel system (OOCS). Even though the OOCS has achieved success in real practice, there is room for further study on this system with regard to deriving a transshipment policy. For this reason, this study aims to develop a solution approach that could derive a promising policy and analyze the impacts of transshipment in the OOCS. The main contributions are summarized as follows. First, we propose a model to deal with the proactive transshipment of perishable products in the OOCS. In particular, this is the first study that introduces the concept of the heterogeneous shelf life considering different properties of online and offline channels. Second, we develop the hybrid deep reinforcement learning (DRL) approach by combining the soft actor–critic algorithm with two novel acceleration methods. The developed method could obtain a promising policy without assumptions about demand distribution and mitigate computational burdens by reducing action spaces. On a set of experiments carried out on real-world demand data, the transshipment policy derived from the hybrid DRL approach could obtain the best profit compared to existing algorithms. Third, we examine the impacts of transshipment by differing types of demand and varying the unit transshipment cost parameter. We find that transshipment substantially reduces the outdating cost by allowing the offline channel to make good use of the old products that will be discarded in the online channel, which is new to the literature.

1. Introduction

The advance of information technology has contributed to the rapid rise of e-commerce platforms (Jiang et al., 2021). Especially in recent years, as customers have become health conscious, the quality of grocery service and the provision of fresh foods has only grown in importance (Li et al., 2021). Due to the ongoing increase in demand for fresh foods in South Korea, several e-commerce companies have emerged that provide delivery and logistics services for fresh foods (e.g., Market Kurly and Coupang Fresh). They offer a unique delivery service: *dawn deliveries*, which are guaranteed to arrive by the following early morning for orders placed at midnight (Hong and Lee, 2022a). It should be noted, however, that these leading companies have been suffering from large net losses,

☆ The authors are grateful for the valuable comments from the guest editor and three anonymous reviewers. This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No. RS-2023-00218913 and No. RS-2024-00337285).

* Corresponding author at: Gwanak-ro 1, Gwanak-gu, Seoul 08826, Republic of Korea.
E-mail addresses: ljh9533@snu.ac.kr (J. Lee), youngchul@ajou.ac.kr (Y. Shin), ikmoon@snu.ac.kr (I. Moon).

https://doi.org/10.1016/j.tre.2024.103576
Received 2 October 2023; Received in revised form 26 January 2024; Accepted 6 May 2024
Available online 22 May 2024

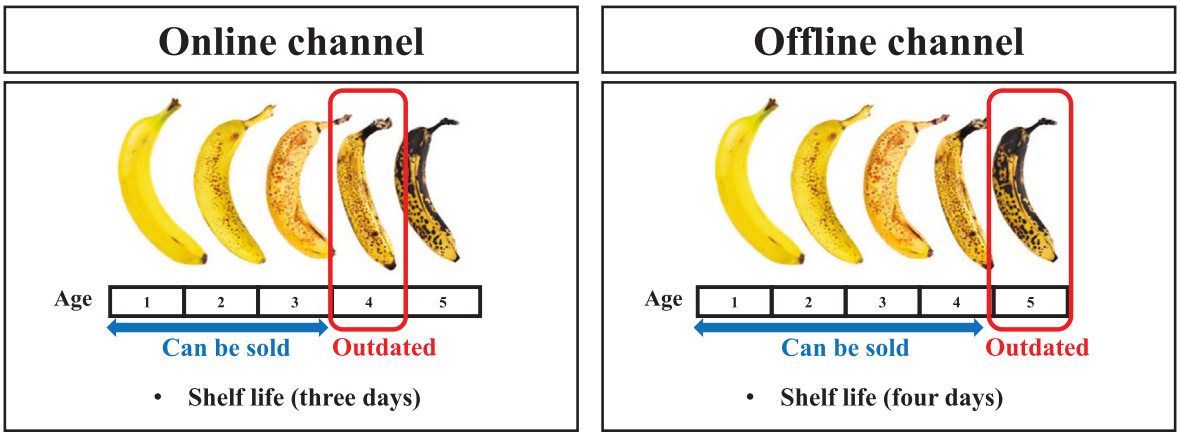


Fig. 1. Example of heterogeneous shelf life property in OOCS (Photo courtesy of *davidwolfe.com*).

and some have stopped providing dawn delivery services (Lee, 2022). A major factor contributing to these companies' net losses is the difficulty of managing perishable food inventories, which leads to substantial costs associated with product waste (Nakandala et al., 2017; Li et al., 2019).

Unlike the other companies, *Oasis Market* is the only company that stays in the black (Kang and Lee, 2022). The majority of companies strive to increase profitability by developing their online platforms, an outlet for selling fresh foods, as well as by improving logistics facilities, such as the cold chain system. Conversely, Oasis Market invests in both online platforms and offline shops simultaneously and has effectively connected the networks. We will use the term *online–offline channel system* (OOCS) to refer to a network of online platforms and offline shops in this paper. Oasis Market aims to reduce the risk of excess ordering by transshipping leftover products between channels.

Even though the OOCS has achieved success in real practice by Oasis Market, there is room for further study on this system. As far as we know, Oasis Market only addresses the movement of products from online to offline channels, not from offline to online channels (i.e., *unidirectional transshipment*) and implements transshipment in a simple manner in which the online channel's unsold products on that day are moved to the offline channel. Based on the above considerations, we further studied the OOCS to meet the following two goals. First, in order to minimize the operational costs in the OOCS, it is necessary to develop a method that could derive a near-optimal in policy determining the transshipment quantities. Second, we aim to study the *mutual transshipment* of products between online and offline channels.

The OOCS is related to research areas of lateral transshipment for perishable products. It is necessary to consider the key attributes of the corresponding business model in order to adopt lateral transshipment in OOCS. Before discussing these attributes, we would like to briefly discuss two types of lateral transshipment (i.e., reactive transshipment and proactive transshipment) and clarify the meaning of the term *shelf life*. The reactive transshipment takes place after observing the demand but before it must be satisfied. On the other hand, the proactive transshipment occurs before the demand has been realized (Paterson et al., 2011; Firouz et al., 2017). In this paper, the standard meaning of *shelf life* is used to indicate the length of periods for which fresh food can be sold. The three attributes of OOCS with lateral transshipment are as follows:

- *Heterogeneous shelf life:* Even though the lifetime of fresh food is homogeneous, the shelf life can be heterogeneous depending on the channel (Hong and Lee, 2022b). Based on the practice of *Oasis Market*, the shelf life of fresh food in the online channel is shorter than in the offline channel due to the risk of product deterioration during delivery (Please refer to Fig. 1). In practice, if fresh food remains unsold in the online channel up to the end of its shelf life, it is typically transshipped to the offline channel and then resold at a discount. Throughout the paper, we will use the term *heterogeneous shelf life* to refer to the property that the shelf life of fresh food is different depending on the channel where it is stored.
- *Proactive transshipment:* When the fresh food that customers want to purchase is out of stock in one retail store, they can easily purchase the same kind of fresh food in another store because fresh foods are sold in many stores. Consequently, customers are not apt to wait until the transshipped item arrives, making reactive transshipment inappropriate for fresh food operations.
- *Non-negligible transshipment time:* It is common for the online distribution center to be located a distance away from the retail store. Typically, an online distribution center is located in a suburban area, whereas an offline retail store is located downtown so that it is more convenient for customers to visit. Moreover, it requires several hours to package and handle inventories that will be transshipped to the other channel. Therefore, the transshipment time should not be negligible when considering inventory movements between online and offline channels.

Although lateral transshipment has been widely studied in the operations management field, a limited number of studies have taken into account perishable products. Depending on the types of lateral transshipment, several researchers focused on reactive transshipment for perishable products (Nakandala et al., 2017; Dehghani and Abbasi, 2018; Zhang et al., 2023), and others

concentrated on the proactive transshipment (Li et al., 2021; Cheong, 2013; Dehghani et al., 2021). Although previous studies have addressed lateral transshipment with perishable products in various aspects, there are two research gaps in existing studies. The first research gap is that no existing study has simultaneously addressed the three attributes of OOCS. To the best of our knowledge, this is the first study to attempt to analyze the heterogeneous shelf life of fresh food. A second research gap relates to assumptions about demand distribution. Most existing studies primarily focus on a specific demand distribution to determine the optimal policy, with the exception of Dehghani et al. (2021). If the gap between real demand and estimated distribution is large, the transshipment policy derived from relying on the estimated demand distribution could show poor performance in real practice. The distinctive differences between our study and Dehghani et al. (2021) will be presented in Section 2.2.

In order to fill the above two research gaps, our study deals with a lateral transshipment model of fresh foods in the OOCS by accommodating three attributes. Without any assumptions on the demand distribution, we develop the hybrid deep reinforcement learning (DRL) approach to directly utilize real-world demand data for deriving a promising transshipment policy. We aim to answer the following three research questions through this study:

1. Can the developed hybrid DRL approach increase average profit compared to existing methods in the setting of real-world demand data?
2. What positive effects does transshipment have in terms of profit depending on the variability of demand and value of unit transshipment cost?
3. How does a transshipment policy change the outdating cost of online and offline channels, respectively, compared to a 'no-transshipment' policy?

The main contributions of our research can be summarized as fourfold. First, we propose the Markov decision process (MDP) model to derive a transshipment policy for fresh foods in the OOCS. Furthermore, we accommodate key attributes of the OOCS in the mathematical model. Second, to derive a promising policy without assumptions about demand distribution, we customize the soft actor–critic (SAC) algorithm, which is one of the state-of-the-art DRL algorithms, for the proposed problem. However, we observe that the SAC algorithm is unstable during the training process due to large action spaces. To mitigate the issue, we propose a novel hybrid DRL approach by developing two acceleration methods. Third, we examine the tendency for transshipment to be effective in high demand variability by conducting computational experiments on various demand data sets. In addition, we analyze the impact of a unit transshipment cost parameter shelf life of online and offline channels through a sensitivity analysis. Fourth, by further analyzing the outdating cost regarding each channel, we discovered that the old product discarded in the online channel can be reutilized in the offline channel through transshipment.

The remainder of this paper is organized as follows: We review existing studies related to lateral transshipment on perishable products and a reinforcement learning (RL) approach for inventory problems in Section 2. In Section 3, we represent the problem description for the OOCS and the corresponding MDP model. In Section 4, we explain the solution methodologies, specifically the SAC algorithm and two accelerating methods. In Section 5, we present the results of computational experiments involving the comparison of algorithms and the analysis of transshipment effects in the OOCS. At last, we provide a summary of this paper, along with further research ideas in Section 6.

2. Literature review

This paper is related to two streams of research in operations management: lateral transshipment on perishable inventory management and an RL approach for inventory problems. In Section 2.1, we review existing studies considering lateral transshipment and perishable inventory and discuss the difference between this paper and other literature. Instead of reviewing all existing studies on the proactive transshipment for perishable products, we present a detailed literature review of several key papers related to our research topic in Section 2.2. Also, we present literature addressing inventory management problems using the RL approach in Section 2.3. Finally, we summarize the limitations of existing studies and explain the contributions of this paper in Section 2.4.

2.1. Comparison between this research and existing studies related to proactive transshipment

A lateral transshipment can be defined as the movement of inventory between several locations of the same echelon, which can result in a reduction in surplus and shortfall at the different locations by transferring inventory between them (Li et al., 2020; Wang et al., 2020). In spite of the complexity of modeling proactive transshipment over modeling reactive transshipment, several studies have focused on developing proactive transshipment models and promising policies (Abouee-Mehrizi et al., 2015; Meissner and Senicheva, 2018). The majority of studies, except for the work of Tagaras and Vlachos (2002), assumed that transshipment time was negligible in order to make the problem tractable. In addition, Tagaras and Vlachos (2002) developed a regular ordering policy and a lateral transshipment policy, as well as conducted extensive simulation studies to verify the advantages of transshipment.

Most studies focusing on proactive transshipment did not consider perishable products in the inventory model because the inclusion of product shelf life increases the state space dimension, making the problem solving more challenging (Li et al., 2013; Glazebrook et al., 2015; Abouee-Mehrizi et al., 2015). Let us assume that the shelf life of products is M and the positive lead time is L. Then the state space dimension becomes $M + L - 1$. Consequently, the inventory model becomes intractable by solving with exact solution methods, such as traditional dynamic programming approaches.

Nevertheless, in recent years, a few studies investigated both perishable products and proactive transshipment at the same time. The majority of research focusing on perishable products has assumed a specific demand distribution to ease the problem and derive

Table 1
Comparison of recent studies related to this research topic.

Author	Transshipment type	Replenishment decision	Heterogeneous shelf life	Non-negligible transshipment	Perishable inventory	Without assumptions on the demand	Solution methodology
Tagaras and Vlachos (2002)	Proactive	✓					Safety stock policy
Li et al. (2013)	Proactive	✓					DE[a]
Cheong (2013)	Proactive	✓			✓		IA[b]
Glazebrook et al. (2015)	Proactive	✓					DP[c]
Abouee-Mehrizi et al. (2015)	Proactive	✓					DP[c]
Nakandala et al. (2017)	Reactive				✓		DE[a]
Dehghani and Abbasi (2018)	Reactive				✓		DE[a]
Meissner and Senicheva (2018)	Proactive					✓	Approximate DP[c]
Dehghani et al. (2021)	Proactive	✓			✓	✓	RH-2SSP[d]
Li et al. (2021)	Proactive	✓			✓		DP[c]
Wei et al. (2022)	Proactive	✓					Approximate DP[c]
Zhang et al. (2023)	Reactive				✓		DP[c]
This research	Proactive	✓	✓	✓	✓	✓	Reinforcement learning

[a] Differential equation.
[b] Iterative algorithm.
[c] Dynamic programming.
[d] Rolling horizon algorithm based on the stochastic programming.

transshipment and ordering policies (Cheong, 2013; Li et al., 2021). On the other hand, two notable studies have addressed proactive transshipment without assuming any specific demand distribution (Meissner and Senicheva, 2018; Dehghani et al., 2021). Meissner and Senicheva (2018) addressed the proactive transshipment using the approximate dynamic programming to relax the demand distribution assumption. However, their method was limited to short planning horizons (i.e., 28 periods) and did not account for replenishment decisions. On the contrary, Dehghani et al. (2021), which is the most relevant study to our research, dealt with perishable inventory and a long planning horizon (i.e., more than 10,000 periods).

We present the distinctive features of our study compared to existing studies in Table 1. To the best of our knowledge, all the existing papers have assumed that the shelf life of a product is the same, regardless of the sales channel in which the product was stored. Additionally, most studies have assumed that transshipment time is negligible, except for the work of Tagaras and Vlachos (2002). However, this paper addresses the issue of heterogeneous shelf life for the first time and considers non-negligible transshipment time. In the following section, we present a detailed literature review of several key papers relevant to our research topic.

2.2. Lateral transshipment for perishable products

Our research is closely related to the stream of literature on lateral transshipment for perishable products. We found that there were only a few studies that investigated both perishable products and lateral transshipment at the same time. First, we review the literature concerning reactive transshipment of perishable products in detail. Nakandala et al. (2017) developed a periodic review perishable inventory model that considered reactive transshipment in the fresh food supply chain. They embodied five cost components and optimized the trade-off among these components. For the purpose of minimizing the total cost, they developed a simple decision rule that requires only information about the spoilage percentage and the parameters of other cost components. However, logistics practitioners may only use this model if they assume a compound Poisson demand distribution and negligible transshipment times.

It is common for research on lateral transshipment of perishable products to consider the periodic review model. Unlike other research, Dehghani and Abbasi (2018) addressed the continuous review perishable inventory model with transshipment. They examined the transshipment of perishable products in a reactive manner and determined stock levels in addition to considering transshipment strategies based on the age threshold of the perishable products. It should be noted, however, that the developed model is only available for Poisson demand distribution. Zhang et al. (2023) considered blood platelets inventory which is crucial for blood products and has a perishable nature. For the purposes of issuing and reactive transshipment, they assumed the first-in-first-out (FIFO) rule. A simple transshipment policy, which guarantees near-optimal results, was developed. An application of the policy developed was successfully implemented in a real hospital system, resulting in a substantial reduction in out-of-date platelets.

Additionally, there are very few studies that specifically deal with proactive transshipment of perishable products, which is the most relevant area for our research. Cheong (2013) presented a proactive transshipment model incorporating an inventory management system for perishable goods for multiple retail stores. The algorithm determines the optimal order and transshipment quantities on a single-period planning horizon. A perishable product with only a two-period lifecycle was assumed by the authors: old and fresh. Therefore, the proposed model cannot be applied to real-world scenarios such as fresh food supply chains or blood supply chains. Wei et al. (2022) emphasized the recycling of perishable products. The authors proposed an approximate dynamic

programming model for transshipment policies that included replenishment and recycling decisions. The developed model, however, should have addressed the shelf life of the perishable product, which is the key characteristic of perishable products. Moreover, since the authors dealt with a very short planning horizon (four months), replenishment and transshipment lead times were not taken into account. Li et al. (2021) considered offline retailing of perishable goods, which included replenishment decisions and proactive transshipments. Considering that customers usually select the freshest items first, the authors assumed the last-in-first-out (LIFO) rule for issuing. Through rigorous analysis, they showed two roles in transshipment: inventory balancing and inventory separation, which means that new inventory is put in one outlet and the older inventory is put in the other, and this research observed this effect for the first time.

The majority of research in lateral transshipment on perishable inventory management is focused on a specific demand distribution. On the other hand, Dehghani et al. (2021) developed a method that does not require an assumption about demand distribution. In addition, the authors assumed that perishable products would arrive immediately, which means the transshipment time is negligible. They utilized a mixed-integer linear programming (MILP) model, which is equivalent to the two-stage stochastic programming (2SSP) model, to decide on proactive transshipment in the blood supply chain. To accommodate a long planning horizon, they developed the rolling horizon algorithm based on the suggested 2SSP model, called *RH-2SSP*. However, RH-2SSP required a considerable amount of computational effort to solve one test instance as the MILP model must be solved for every period. For example, if the decision maker wishes to solve a 10,000 period problem, the MILP model must be solved 10,000 times for each instance. Similarly, the RL approach also required several hours to train the RL agent for one instance. In our computational experiments, we found that training both hybrid DRL and RH-2SSP for the same instance took a similar amount of time. Nevertheless, the trained neural networks of hybrid DRL are capable of reusing other test data sets. In Section 5.1 we will compare the performance of the hybrid DRL and the RH-2SSP in detail.

2.3. Reinforcement learning approach for inventory management

In the past decade, many optimization approaches have been developed to deal with deterministic decision problems. Mathematical programming, such as linear programming (LP) and MILP, has been utilized to model the problems into the mathematical form. In particular, various meta-heuristics have been developed to address large-scale optimization models and solve challenging decision problems (i.e., NP-hard problems) (Guo et al., 2023a; Zhou et al., 2023; Wu et al., 2023; Guo et al., 2023b). On the other hand, the sequential decision-making problem addresses the dynamic system that evolves over time depending on a decision maker's action and revealed information. An MDP is usually utilized to model the sequential decision-making problem. Recently, the RL approach, specifically the DRL algorithm, has been utilized to solve an MDP because it is a promising method for dealing with high-dimensional problems and does not require knowledge about the transition probability in the concerned system. However, because DRL requires a significant amount of data to learn an optimal policy, it is difficult to apply in the domains where obtaining data is difficult or expensive. Fortunately, because there is immense data for customer demand in inventory management, researchers studying inventory management problems have paid considerable attention to the RL and DRL approaches. Many researchers tried to employ the RL approach to solve intractable inventory problems: perishable inventory management (Kara and Dogan, 2018; De Moor et al., 2022), beer game (Oroojlooyjadid et al., 2022), joint replenishment problem (Vanvuchelen et al., 2020), multi-product and multi-node inventory management (Sultana et al., 2020), and joint pricing and inventory problem (Zhou et al., 2022). In addition, Boute et al. (2021) suggested a number of research avenues that may help to adopt the DRL approach to practical inventory management problems.

Instead of reviewing all existing studies on the RL approach to inventory management, we present a detailed literature review of four key papers related to our research topic. Gijsbrechts et al. (2022) applied the DRL algorithm, namely the asynchronous advantage actor–critic (A3C) (Mnih et al., 2016), to address three classic and intractable inventory problems: lost sales, dual-sourcing, and multi-echelon inventory management. The training of neural networks of DRL for each different inventory problem requires extensive tuning of hyperparameters, which is inevitable when it comes to designing neural networks of DRL. In order to mitigate this burden, the authors proposed a method for automatically tuning several types of hyperparameters. Compared with state-of-the-art heuristics and approximate dynamic programming, the developed A3C algorithm was able to achieve similar performance for each problem. This result implies that DRL could be a promising approach for inventory problems in which effective heuristics are lacking.

Oroojlooyjadid et al. (2022) proposed a Deep Q-Network (DQN) algorithm as part of a feedback scheme for the reward function applied to the beer game, a decentralized, multi-agent and cooperative problem. Although there are four players in the suggested beer game, the authors assumed that only one player could use the DQN, and the others would act irrationally. Without knowledge of the demand probability distribution, real demand data was employed to test the performance of the DQN algorithm. Furthermore, the transfer learning approach was developed to boost the learning speed of DQN in different cost instances. However, the developed DQN algorithm can only be applied in situations where there is only one player using the DQN algorithm. To practically employ the DRL approach to reduce the bullwhip effect, it is more reasonable that all players use DRL than in the above situation. The multi-agent DRL could be a more suitable approach to the beer game problem.

Kara and Dogan (2018) and De Moor et al. (2022) have both utilized the RL approach to manage perishable inventory. Kara and Dogan (2018) employed Q-learning and Sarsa algorithms to solve the problem. They defined two different states: the stock-age-based state and the quantity-based state. Computational experiments showed that Q-learning combined with a stock-age-based state showed the most promising performance. However, the authors should have compared the developed algorithm with state-of-the-art heuristics and evaluated performance systematically using an optimal policy or lower bound; thus, it is difficult to trust the RL algorithm.

In contrast, De Moor et al. (2022) evaluated the performance of the developed DQN algorithm in comparison with the optimal policy produced by value iteration (VI). They particularly implemented potential-based reward shaping, which can transfer knowledge of a state-of-the-art inventory policy (teacher policy) into the DQN algorithm, which is called shaped DQN. It was found that the shaped DQN algorithm outperformed the DQN algorithm without reward shaping for the small size problem (the product has a two-period shelf life). Occasionally, the shaped DQN showed better results than a teacher policy. However, for the practical size problem, the shaped DQN rarely outperformed the teacher policy, which means that the DQN algorithm with potential-based reward shaping is unlikely to surpass the performance of the state-of-the-art inventory policy. As Kara and Dogan (2018) and De Moor et al. (2022) employed basic reinforcement algorithms, Q-learning and DQN, respectively; thus, there may be room for improving the performance by implementing state-of-the-art reinforcement learning algorithms, such as SAC and proximal policy optimization (PPO).

2.4. Limitations of existing studies and contributions of this research

In summary, only a few research papers have addressed a proactive transshipment of perishable products. However, they heavily relied on the assumption of demand distribution to derive the transshipment policy. The transshipment policy derived from relying on the estimated distribution could perform poorly in real practice if the gap between the real and the estimated distribution is significant. Therefore, Dehghani et al. (2021) developed the RH-2SSP, which did not need any assumption about demand distribution. However, the RH-2SSP showed poor performance on our problem.

Many researchers tried to employ the DRL approach to solve intractable inventory problems. However, the majority of research only studied the ordering policy, and there was no research that studied DRL for determining ordering and transshipment policies at the same time. In order to validate the performance of the DRL approach, it is necessary to compare it with state-of-the-art heuristics or the lower bound (e.g., the objective value obtained from the perfect information setting). However, several studies failed to validate the developed DRL approach's performance systematically.

We present the main contributions of our study, which can fill the above research gaps, from the following two perspectives:

- *Modeling:* Our research deals with the proactive transshipment of perishable products in the OOCS. Specifically, we consider the following three attributes to accommodate key features of the OOCS for fresh foods, as mentioned in Section 1. These features are heterogeneous shelf life, proactive transshipment, and non-negligible transshipment time. In particular, we introduce the concept of the heterogeneous shelf life for the first time to consider the different properties of online and offline channels.
- *Solution approach:* We propose the DRL approach based on the SAC algorithm in order to use data directly for decision-making without making any assumptions about demand distribution. We develop novel acceleration methods to improve the performance of the DRL approach to our problem. First, we develop the hybrid method to reduce action spaces, which could mitigate computational burdens and stabilize the training process. This hybrid method could decouple transshipment decisions from replenishment decisions. The transshipment quantity is determined with the DRL, and replenishment is implemented through a newly developed ordering policy called the *SQLT policy*. Second, we propose the *reward shaping* (RS) method to maximize the average profit by incorporating the exterior knowledge of a teacher heuristic into the DRL. On a set of experiments carried out in various experimental settings, we could observe that the proposed algorithm, which combines the SAC with the two acceleration methods, surpassed the original SAC algorithm.

3. Problem description and mathematical model

3.1. Lateral transshipment for fresh foods in the online–offline channel system (OOCS)

We consider a periodic review, infinite horizon, perishable inventory problem with stochastic demand and fixed positive lead time. We assume that the unsatisfied demand will become lost sales. There are two heterogeneous outlets, online and offline channels, indexed by superscript ON and OF, owned by the same retailer as in Fig. 2. Each channel has its own demand for customers, and the demand is random and must be satisfied by the inventories of each channel. The online distribution center satisfies the online demand D_t^{ON}, and the offline store satisfies the offline demand D_t^{OF}. There are deterministic lead times for online and offline channels ($L^{ON} \geq 0$ and $L^{OF} \geq 0$). We focus on a single type of fresh food, and perishability exists due to the nature of fresh food. From here forward, we will use the term *product* to indicate fresh food. Furthermore, we accommodate the following assumptions to consider the properties of fresh food and the OOCS in real business:

- Shelf life of the product is different depending on the outlet. The shelf life of the product held in the online channel is shorter than in the offline channel.
- Even though online and offline channels sell the same item, the sale price of the offline channel is cheaper than that of the online channel.
- Demand distributions of online and offline channels are different.

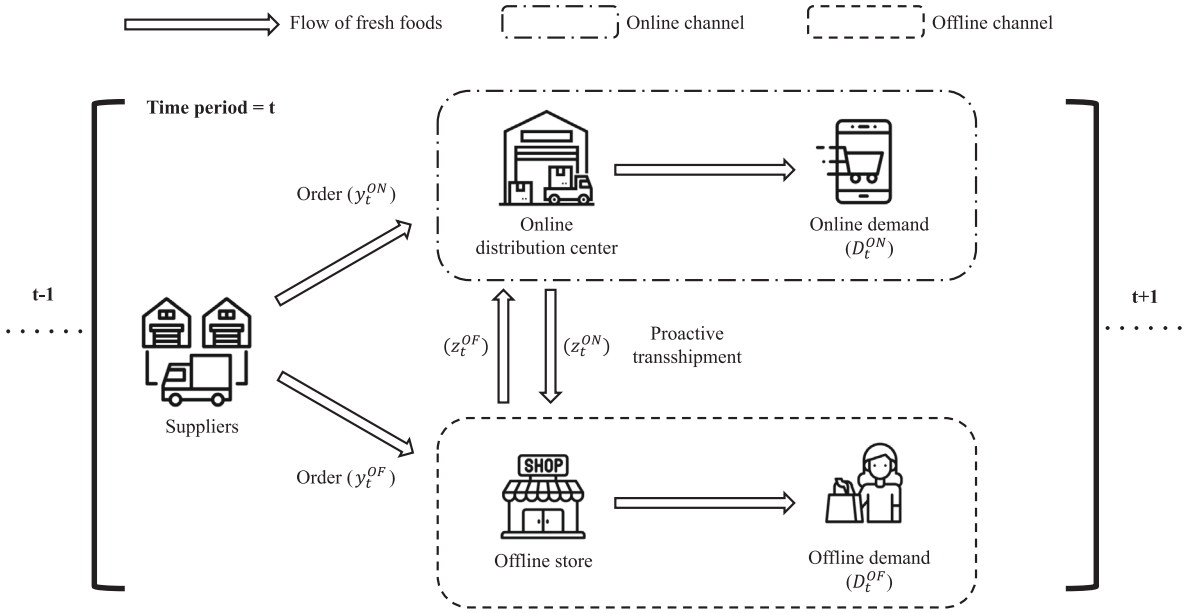


Fig. 2. Overview of the flow of fresh foods in the OOCS.

- Because online distribution centers and offline stores are located in different regions, the lead times for replenishment of the online and offline channels are different.

Note that we define the sale prices of the online and offline channels as the prices that customers have to pay for buying the products at the corresponding sales channel without considering the housing cost for utilizing offline stores or the labor cost. Based on this definition of the sale price, we accommodate the second assumption by referring to the real practice of the marketplace in South Korea and the article of Roerink (2021). According to Roerink (2021), online shoppers are less price-sensitive; thus, they spend more online than offline to buy fresh foods. In particular, in the case of the deli items, the average price per unit is 42 percent more expensive in the online channel than it is in the offline channel. In addition, additional packaging processes (e.g., cold chain packaging) are necessary to deliver products to customers while preserving freshness, which could incur a higher sale price for the online channel than for the offline channel.

The objective of the proposed problem is to maximize the average profit per period of the OOCS. Each channel orders products from suppliers, and the transshipment is mutually implemented between channels. The retailer makes four types of decisions at each period t: (1) order quantity of online channel, y_t^{ON}, (2) order quantity of offline channel, y_t^{OF}, (3) quantity of transshipped units from online channel to offline channel, z_t^{ON}, and (4) quantity of transshipped units from offline channel to online channel, z_t^{OF}. We consider that the issuing policy is FIFO, and the transshipment policy is first-in-first-transship (FIFT) in both outlets. In particular, the FIFT refers to the strategy of transshipping the older products ahead to the other channel while reserving the younger products (Wang and Ma, 2015). Note that the problem can easily be extended to the LIFO issuing policy by revising the logic for updating state variables. We consider the case that the transshipment time is non-negligible. Specifically, products from the origin channel are transshipped before demand is realized, and transshipped products arrive at the destination channel in the next period. We assume that realized demand and all variables are integers, to accommodate the situation that most e-commerce companies deal with packaged fresh foods, which counted as units.

To describe the evolution of the proposed system, we use the following notations and present the formal definition of several terminologies in the supplementary material:

Indices and sets

$\mathcal{T}$	Set of periods, $t \in \mathcal{T} = \{1, 2, \ldots, T\}$
$\mathcal{M}^{ON}$	Set of the age of product held in the online channel, $m \in \mathcal{M}^{ON} = \{1, 2, \ldots, M^{ON}\}$
$\mathcal{M}^{OF}$	Set of the age of product held in the offline channel, $m \in \mathcal{M}^{OF} = \{1, 2, \ldots, M^{OF}\}$

Parameters

p^{ON}	Sale price for a unit of product in online channel
p^{OF}	Sale price for a unit of product in offline channel ($p^{ON} > p^{OF}$)
M^{ON}	Shelf life of product held in online channel
M^{OF}	Shelf life of product held in offline channel ($M^{OF} > M^{ON}$)
y_{max}^{ON}	Maximum order quantity at each period in online channel
y_{max}^{OF}	Maximum order quantity at each period in offline channel
z_{max}^{ON}	Maximum transshipment quantity at each period from online channel to offline channel
z_{max}^{OF}	Maximum transshipment quantity at each period from offline channel to online channel
L^{ON}	Lead time of orders in online channel
L^{OF}	Lead time of orders in offline channel
c_h	Inventory holding cost for a unit of product
c_p	Lost sales cost for a unit of product
c_l	Transshipment cost for a unit of product
c_w	Outdating cost for a unit of product
c_o	Ordering cost for a unit of product

State variables

$IL_{m,t}^{ON}$	Starting inventory level of age m product at period t in online channel
$IL_{m,t}^{OF}$	Starting inventory level of age m product at period t in offline channel
$OT_{l,t}^{ON}$	Pipeline inventory that will arrive after $L^{ON}-l$ periods at period t in online channel
$OT_{l,t}^{OF}$	Pipeline inventory that will arrive after $L^{OF}-l$ periods at period t in offline channel
$LT_{m,t}^{ON}$	Transshipment quantity of age m product at period t from online channel to offline channel
$LT_{m,t}^{OF}$	Transshipment quantity of age m product at period t from offline channel to online channel

Concatenated vectors of state variables

IL_t	Concatenated vectors of the $\left(IL_{1,t}^{ON},\dots,IL_{M^{ON},t}^{ON},IL_{1,t}^{OF},\dots,IL_{M^{OF},t}^{OF}\right)$
OT_t	Concatenated vectors of the $\left(OT_{1,t}^{ON},\dots,OT_{L^{ON}-1,t}^{ON},OT_{1,t}^{OF},\dots,OT_{L^{OF}-1,t}^{OF}\right)$
LT_t	Concatenated vectors of the $\left(LT_{1,t}^{ON},\dots,LT_{M^{ON}-1,t}^{ON},LT_{1,t}^{OF},\dots,LT_{M^{ON}-2,t}^{OF}\right)$

For each period t, the following sequence of an event is repeated:

(a) At the start of period t, the starting inventory level, determined at the end of the previous period $t-1$, is observed. We consider $IL_{1,t}^{ON}$ and $IL_{1,t}^{OF}$ to be the youngest product and $IL_{M^{ON},t}^{ON}$ and $IL_{M^{OF},t}^{OF}$ to be the oldest product that expires at the end of the previous period $t-1$.

(b) Four types of decisions, $y_t^{ON}, y_t^{OF}, z_t^{ON}$, and z_t^{OF}, are implemented at the start of period t. The limit of order quantity exists for each channel, $0 \le y_t^{ON} \le y_{max}^{ON}$ and $0 \le y_t^{OF} \le y_{max}^{OF}$. Because of the lead time of each channel, the y_t^{ON} will be received at time $t+L^{ON}$, and the y_t^{OF} will be received at time $t+L^{OF}$. For each channel, the limit of transshipment quantity exists and is determined by the current inventory level, $0 \le z_t^{ON} \le \min\left\{z_{max}^{ON}, \sum_{m=1}^{M^{ON}-1} IL_{m,t}^{ON}\right\}$ and $0 \le z_t^{OF} \le \min\left\{z_{max}^{OF}, \sum_{m=1}^{M^{OF}-1} IL_{m,t}^{OF}\right\}$. The transshipped product is received at each channel at the start of period $t+1$.

(c) The random demand, D_t^{ON} and D_t^{OF}, is realized; as much as possible, it is satisfied from the inventory level, $IL_{m,t}^{ON}$ and $IL_{m,t}^{OF}$, and the rest is lost. The inventory level and pipeline inventory for the next period $t+1$ are updated.

(d) At the end of period t, revenue and inventory holding, shortage, outdating, ordering, and the transshipment costs are assessed.

We utilize the state variables $LT_{m,t}^{ON}$ and $LT_{m,t}^{OF}$ to implement the FIFT policy for transshipment. The $LT_{m,t}^{ON}$ and $LT_{m,t}^{OF}$ are decided as follows:

$$LT_{m,t}^{ON} = \min\left\{\left(z_t^{ON} - \sum_{k=m+1}^{M^{ON}-1} IL_{k,t}^{ON}\right)^+, IL_{m,t}^{ON}\right\}, \qquad \forall m \in \left\{1,\dots,M^{ON}-2\right\}, \tag{1}$$

$$LT_{M^{ON}-1,t}^{ON} = \min\left\{IL_{M^{ON}-1,t}^{ON}, z_t^{ON}\right\}, \tag{2}$$

$$LT_{m,t}^{OF} = \min\left\{\left(z_t^{OF} - \sum_{k=m+1}^{M^{ON}-2} IL_{k,t}^{OF}\right)^+, IL_{m,t}^{OF}\right\}, \qquad \forall m \in \left\{1,\dots,M^{ON}-3\right\}, \tag{3}$$

$$LT^{OF}_{M^{ON}-2,t} = \min\left\{ IL^{OF}_{M^{ON}-2,t}, z^{OF}_t \right\}. \tag{4}$$

where $x^+ = \max\{x, 0\}$. The index m of $LT^{ON}_{m,t}$ includes from one to $M^{ON}-1$, and the index m of $LT^{OF}_{m,t}$ includes from one to $M^{ON}-2$, because the shelf life of the product in the online channel is shorter than that in the offline channel, and the transshipment quantity will be received at the next period, $t+1$. Specifically, assume that the age $M^{ON}-1$ product is transshipped from the offline channel to the online channel in period t. This type of transshipment will become worthless because the transshipped product's age becomes M^{ON} when the product arrives at the online channel, i.e., the transshipped product cannot be sold to customers.

State variables for inventory level and pipeline inventory in the online channel are updated based on the following equations after D^{ON}_t is realized:

Online channel

$$OT^{ON}_{1,t+1} = y^{ON}_t, \tag{5}$$

$$OT^{ON}_{l,t+1} = OT^{ON}_{l-1,t}, \qquad \forall l \in \{2, \dots, L^{ON}-1\}, \tag{6}$$

$$IL^{ON}_{1,t+1} = OT^{ON}_{L^{ON}-1,t}, \tag{7}$$

$$IL^{ON}_{m+1,t+1} = \left(\left(IL^{ON}_{m,t} - LT^{ON}_{m,t} \right) - \left(D^{ON}_t - \sum_{k=m+1}^{M^{ON}-1} \left(IL^{ON}_{k,t} - LT^{ON}_{k,t} \right) \right)^+ \right)^+ + LT^{OF}_{m,t}, \qquad \forall m \in \{1, \dots, M^{ON}-2\}, \tag{8}$$

$$IL^{ON}_{M^{ON},t+1} = \left(\left(IL^{ON}_{M^{ON}-1,t} - LT^{ON}_{M^{ON}-1,t} \right) - D^{ON}_t \right)^+. \tag{9}$$

The inventory level for the period $t+1$ is updated by subtracting the current inventory level from the transshipment quantity (i.e., $IL^{ON}_t - LT^{ON}_t$).

State variables for inventory level and pipeline inventory in the offline channel are updated based on the following equations after D^{OF}_t is realized. The pipeline inventory in the offline channel updated as the same transitions in the online channel, Eqs. (5) and (6), as follows:

Offline channel (pipeline inventory)

$$OT^{OF}_{1,t+1} = y^{OF}_t, \tag{10}$$

$$OT^{OF}_{l,t+1} = OT^{OF}_{l-1,t}, \qquad \forall l \in \{2, \dots, L^{OF}-1\}. \tag{11}$$

However, because the shelf life of the online channel is shorter than that of the offline channel, the offline channel's inventory level is updated differently depending on the product's age m. If the index m is included in the set $\{1, \dots, M^{ON}-2\}$, the inventory level for the period $t+1$ is also updated by subtracting the current inventory level from the transshipment quantity (i.e., $IL^{OF}_t - LT^{OF}_t$) as follows:

Offline channel (inventory level)

$$IL^{OF}_{1,t+1} = OT^{OF}_{L^{OF}-1,t}, \tag{12}$$

$$IL^{OF}_{m+1,t+1} = \left(\left(IL^{OF}_{m,t} - LT^{OF}_{m,t} \right) - \left(D^{OF}_t - \sum_{k=m+1}^{M^{ON}-2} \left(IL^{OF}_{k,t} - LT^{OF}_{k,t} \right) - \sum_{k=M^{ON}-1}^{M^{OF}-1} IL^{OF}_{k,t} \right)^+ \right)^+ + LT^{ON}_{m,t}, \qquad \forall m \in \{1, \dots, M^{ON}-3\}, \tag{13}$$

$$IL^{OF}_{M^{ON}-1,t+1} = \left(\left(IL^{OF}_{M^{ON}-2,t} - LT^{OF}_{M^{ON}-2,t} \right) - \left(D^{OF}_t - \sum_{k=M^{ON}-1}^{M^{OF}-1} IL^{OF}_{k,t} \right)^+ \right)^+ + LT^{ON}_{M^{ON}-2,t}. \tag{14}$$

When the index m is included in the set $\{M^{ON}-1, \dots, M^{OF}\}$, there is no need to subtract current inventory from the transshipment quantity because the product cannot be transshipped from the offline channel to the online channel. In addition, if the index m is included in the set $\{M^{ON}, \dots, M^{OF}\}$, there is no transshipment quantity from online channel to offline channel, thus, $LT^{ON}_{m,t}$ is not considered as follows:

$$IL^{OF}_{M^{ON},t+1} = \left(IL^{OF}_{M^{ON}-1,t} - \left(D^{OF}_t - \sum_{k=M^{ON}}^{M^{OF}-1} IL^{OF}_{k,t} \right)^+ \right)^+ + LT^{ON}_{M^{ON}-1,t}, \tag{15}$$

$$IL^{OF}_{m+1,t+1} = \left(IL^{OF}_{m,t} - \left(D^{OF}_t - \sum_{k=m+1}^{M^{OF}-1} IL^{OF}_{k,t} \right)^+ \right)^+, \qquad \forall m \in \{M^{ON}, \dots, M^{OF}-2\}, \tag{16}$$

$$IL^{OF}_{M^{OF},t+1} = \left(IL^{OF}_{M^{OF}-1,t} - D^{OF}_t\right)^+. \tag{17}$$

At each period t, revenue and costs are defined as follows based on the above state and decision variables:

$$\text{Inventory holding } (HC_t) := c_h\left[\left(\sum_{m=1}^{M^{ON}-2}\left(IL^{OF}_{m,t} - LT^{OF}_{m,t}\right) + \sum_{m=M^{ON}-1}^{M^{OF}-1} IL^{OF}_{m,t} - D^{OF}_t\right)^+ + \left(\sum_{m=1}^{M^{ON}-1}\left(IL^{ON}_{m,t} - LT^{ON}_{m,t}\right) - D^{ON}_t\right)^+\right] \tag{18}$$

$$\text{Shortage } (SC_t) := c_p\left[\left(D^{OF}_t - \sum_{m=1}^{M^{ON}-2}\left(IL^{OF}_{m,t} - LT^{OF}_{m,t}\right) - \sum_{m=M^{ON}-1}^{M^{OF}-1} IL^{OF}_{m,t}\right)^+ + \left(D^{ON}_t - \sum_{m=1}^{M^{ON}-1}\left(IL^{ON}_{m,t} - LT^{ON}_{m,t}\right)\right)^+\right] \tag{19}$$

$$\text{Outdating } (WC_t) := c_w\left(IL^{ON}_{M^{ON},t+1} + IL^{OF}_{M^{OF},t+1}\right) \tag{20}$$

$$\text{Ordering } (OC_t) := c_o\left(y^{ON}_t + y^{OF}_t\right) \tag{21}$$

$$\text{Transshipment } (TC_t) := c_l\left(z^{ON}_t + z^{OF}_t\right) \tag{22}$$

$$\text{Revenue } (RV_t) := p^{ON}\left[\min\left\{\sum_{m=1}^{M^{ON}-1}\left(IL^{ON}_{m,t} - LT^{ON}_{m,t}\right), D^{ON}_t\right\}\right] + p^{OF}\left[\min\left\{\sum_{m=1}^{M^{ON}-2}\left(IL^{OF}_{m,t} - LT^{OF}_{m,t}\right) + \sum_{m=M^{ON}-1}^{M^{OF}-1} IL^{OF}_{m,t}, D^{OF}_t\right\}\right] \tag{23}$$

At last, the profit at period t, PF_t, is defined as: $PF_t := RV_t - HC_t - SC_t - WC_t - OC_t - TC_t$.

3.2. Markov decision process for the proposed lateral transshipment problem

The proposed problem can be formalized as the MDP. A MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, r, p, \gamma\rangle$, consisting of five components— a set of states, $\mathcal{S}$; a set of actions, $\mathcal{A}$; the reward function, r; the state transition probability function, p; and the discount factor, $\gamma \in [0, 1)$. Most existing studies of perishable inventory management defined the state at time t as inventory level and pipeline inventory, $s_t = \left(IL_t, OT_t\right)$ (Kara and Dogan, 2018; De Moor et al., 2022; Gijsbrechts et al., 2022). In addition to IL_t and OT_t, we include the transshipment information at previous period $t-1$, LT_{t-1}, in the state at time t, $s_t = (IL_t, OT_t, LT_{t-1})$. The state space $\mathcal{S}$ is thus $\left(3M^{ON} + M^{OF} + L^{ON} + L^{OF} - 5\right)$ dimensional.

We consider a discrete action space, and the action at time t is defined as $a_t = (y^{ON}_t, y^{OF}_t, z^{ON}_t, z^{OF}_t)$. Due to the maximum order and transshipment quantities in each channel, the size of action space $|\mathcal{A}|$ is $\left(y^{ON}_{max}+1\right)\times\left(y^{OF}_{max}+1\right)\times\left(z^{ON}_{max}+1\right)\times\left(z^{OF}_{max}+1\right)$. The valid actions for transshipment are different depending on the current state s_t, specifically the $IL^{ON}_{m,t}$ and $IL^{OF}_{m,t}$ as indicated in Section 3.1. The transition probability function is denoted as $p(s_{t+1}|s_t, a_t)$, which indicates the probability that the system is in state s_{t+1} at period $t+1$ when the action a_t is chosen under state s_t at period t. The transition probability can be computed if the demand distribution is known. The reward function quantifies how well the immediate action a_t and state s_t are chosen. Because the purpose of the proposed problem is to maximize the average profit per period, the reward function can be defined as follows: $r(s_t, a_t) := PF_t$.

The objective of solving the MDP is to find a policy $\pi : \mathcal{S} \to \mathcal{A}$, mapping each state to an action, that maximizes the expected cumulative reward:

$$\max_{\pi\in\Pi} \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right] \tag{24}$$

where Π is the set of all policies and $\mathbb{E}^{\pi}$ is the expectation operator when following policy π. The value function $V^{\pi}(s)$ of a policy π is the expected cumulative reward starting from state s under executing π:

$$V^{\pi}(s) := \mathbb{E}^{\pi}\left[\sum_{\tau=t}^{\infty}\gamma^{\tau-t} r(s_\tau, a_\tau)|s_t = s\right]. \tag{25}$$

The optimal policy can be derived from the optimal value function $V^*(s) := \max_{\pi\in\Pi} V^{\pi}(s)$, $\forall s \in \mathcal{S}$, which is the maximum value function overall policies. When the finite state and actions sets are assumed, the optimal value function can be obtained by solving the following Bellman equations recursively (i.e., VI algorithm Bertsekas, 2012):

$$V^*(s) := \max_{a\in\mathcal{A}}\left[r(s, a) + \gamma\sum_{s'\in\mathcal{S}} p(s'|s, a)V^*(s')\right]. \tag{26}$$

However, when dealing with the large-scale and complex MDP, it becomes challenging to solve with the VI algorithm, due to two issues (Gosavi, 2009). First, if several random variables should be manipulated in the MDP, it is generally difficult to compute the

transition probability. Furthermore, it is impossible to find transition probability if the probability distribution of random variables is not known. This issue is called the *curse of modeling*. Furthermore, obtaining the accurate transition probability is only possible if the true demand distribution is known. Second, it is challenging to store and handle the value function for all states, $\forall s \in S$, when the problem involves a large dimension. This issue is called the *curse of dimensionality*.

To solve the proposed problem, we first tried to solve the problem through the VI algorithm with the assumption that the demand distribution is known. However, the dimension of state and action is enormous because our problem deals with two outlets and the property of perishability. Even though we tested on small-size instances, it was impossible to obtain the value function because of the memory issue. Moreover, transition probability cannot be computed directly from real demand data. In order to mitigate the above issues, we adopt the DRL approach to solve the proposed problem. The capability of DRL lies in its ability to solve the complex MDP, which involves the large dimension of state and action, near-optimally. Furthermore, the Model-free DRL approach does not need to know or learn the transition probability. Instead, the policy is learned through interactions between the environment and agent utilizing the demand data set directly.

Among the various Model-free DRL algorithms, the SAC for discrete action settings is a state-of-the-art algorithm that deals with DRL problems with discrete action space (He et al., 2022; Christodoulou, 2019). In detail, we utilize the SAC algorithm because of the following three perspectives:

- *Tabular RL vs. DRL:* The tabular-based method, such as Q-learning, utilizes a look-up table to represent the state–action function. DRL involves neural networks as function approximators for addressing the state–action function to deal with the large dimension. The most practical problems have a large dimension property, and our model has a large state–action space because of perishability, dual-channel, and lead time properties. Therefore, instead of tabular RL, we utilize the DRL algorithm.
- *Off-policy vs. On-policy:* Off-policy algorithms can reuse past experience, which increases sample efficiency. On the other hand, on-policy algorithms (e.g., PPO Schulman et al., 2017 and A3C Mnih et al., 2016) require new samples at each gradient step, which requires significant training time compared to off-policy algorithms. Because of the nature of the DRL, it is necessary to train parameters in DRL again if the problem circumstances change. The circumstances of our problem can vary when the cost parameters or shelf life of each channel change. Therefore, we adopt the off-policy algorithm, which has a higher sample efficiency than the on-policy algorithm.
- *Exploration-exploitation dilemma:* There are various off-policy DRL algorithms, such as DQN, double DQN (Van Hasselt et al., 2016), and deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015). However, most off-policy algorithms struggle to balance the trade-off between exploration and exploitation because they are too sensitive to the hyperparameters. We first tried to solve the problem with DQN and DDQN, but they had poor performance because of the above issue. On the other hand, the SAC could balance the trade-off of exploration and exploitation by utilizing the maximum entropy framework. Therefore, we decided to utilize the SAC.

4. Solution methodology: hybrid deep reinforcement learning (DRL) approach

4.1. Soft actor–critic (SAC) algorithm

Haarnoja et al. (2018a) introduced the SAC algorithm, which is an off-policy actor–critic DRL algorithm based on the maximum entropy framework. The exploration and robustness are improved by using the maximum entropy framework. In the maximum entropy MDP problem, the concept of entropy of policy is used as follows:

$$\mathcal{H}\left(\pi\left(\cdot|s\right)\right) := \mathbb{E}_{a\sim\pi(\cdot|s)}\left[-\log\pi\left(a|s\right)\right]. \tag{27}$$

The goal of the maximum entropy MDP problem is to find a policy that maximizes the maximum entropy objective:

$$\max_{\pi\in\Pi}\sum_{t=0}^{\infty}\mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[\gamma^t\left(r(s_t,a_t)\right)+\alpha\mathcal{H}\left(\left(\pi(\cdot|s_t)\right)\right)\right] \tag{28}$$

where ρ_π is the distribution of trajectories induced by policy π, and α is the temperature parameter, which is utilized to control the relative importance of the reward and entropy.

From now on, we will introduce methods to derive optimal policy in the maximum entropy MDP in two different settings: (1) tabular setting, and (2) large spaces setting. First, in a tabular setting, the optimal policy can be found by repeating the implementation of soft policy evaluation and soft policy improvement (i.e., soft policy iteration) (Haarnoja et al., 2018a). For the soft policy evaluation, the following soft Q-function of a policy π can be computed by repeatedly applying the modified Bellman backup operator:

$$Q_\pi^{soft}\left(s_t,a_t\right) := r\left(s_t,a_t\right)+\gamma\mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t)}\left[V_\pi^{soft}\left(s_{t+1}\right)\right] \tag{29}$$

where

$$V_\pi^{soft}(s_t) := \mathbb{E}_{a_t\sim\pi(\cdot|s_t)}\left[Q_\pi^{soft}\left(s_t,a_t\right)-\alpha\log\pi\left(a_t|s_t\right)\right] \tag{30}$$

is called the soft value function. For the soft policy improvement, the policy is updated towards maximizing the rewards, which is the exponential of the new soft Q-function. The set of policies Π is considered to constrain policies to a parameterized family of distributions (e.g., Gaussian). To accommodate the above constraint, $\pi \in \Pi$, we utilize the Kullback–Leibler (KL) divergence for information projection; thus, the policy is updated as follows:

$$\pi_{new} = \underset{\pi \in \Pi}{\operatorname{argmin}} D_{KL}\left(\pi\left(\cdot|s_t\right) \middle\| \frac{\exp\left(\frac{1}{\alpha} Q^{soft}_{\pi_{old}}(s_t,\cdot)\right)}{Z_{\pi_{old}}(s_t)} \right) \tag{31}$$

where $Z_{\pi_{old}}(s_t)$ is the partition function used to normalize the distribution but can be ignored because it does not contribute to the gradient descent.

In order to practically employ the soft policy iteration in large spaces (e.g., continuous state), the function approximators, neural networks, are utilized for both the soft Q-function and the policy. A neural network with parameter θ is used for the soft Q-function, $Q^{soft}_{\theta}(s_t,a_t)$ and a neural network with parameter ϕ is used for the policy $\pi_{\phi}(a_t|s_t)$. To mitigate the issue of positive bias, two soft Q-functions and neural networks are utilized, $Q^{soft}_{\theta_i}(s_t,a_t)$, $\forall i \in \{1,2\}$ (Hasselt, 2010). Also, two target soft Q-networks are used to compute the shared target and improve the training stability, $Q^{soft}_{\bar{\theta}_j}(s_t,a_t)$, $\forall j \in \{1,2\}$. Both target soft Q-networks are updated through the soft update approach, which can be represented as:

$$\bar{\theta}_j \leftarrow \psi\theta_j + (1-\psi)\bar{\theta}_j \tag{32}$$

where $\psi \in [0,1]$ is the parameter for weight update.

In the large spaces setting, we call soft policy evaluation as *critic* and soft policy improvement as *actor*. We train each soft Q-function parameter θ_i to minimize the following critic cost function:

$$J_{Q^{soft}}\left(\theta_i\right) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\left(Q^{soft}_{\theta_i}\left(s_t,a_t\right) - \left(r(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim p(\cdot|s_t,a_t)}\left[V_{\bar{\theta}}(s_{t+1})\right]\right)\right)^2\right] \tag{33}$$

where:

$$V_{\bar{\theta}}(s_{t+1}) = \mathbb{E}_{a_{t+1}\sim\pi_{\phi}(\cdot|s_{t+1})}\left[\min_{j\in\{1,2\}} Q^{soft}_{\bar{\theta}_j}(s_{t+1},a_{t+1}) - \alpha\log\pi_{\phi}(a_{t+1}|s_{t+1})\right]. \tag{34}$$

The replay buffer storing trajectories of experience is denoted as D. Through sampling experiences from the replay buffer, the soft value function $V_{\bar{\theta}(s_{t+1})}$ is estimated through the Monte-Carlo method.

The policy parameter ϕ can be learned by directly minimizing the KL divergence in Eq. (31) as follows by multiplying α and ignoring the partition function:

$$J_{\pi}(\phi) = \mathbb{E}_{s_t\sim D}\left[\mathbb{E}_{a_t\sim\pi_{\phi}(\cdot|s_t)}\left[\alpha\log\left(\pi_{\phi}(a_t|s_t)\right) - \min_{j\in\{1,2\}} Q^{soft}_{\bar{\theta}_j}(s_t,a_t)\right]\right]. \tag{35}$$

Generally, the policy $\pi_{\phi}(a_t|s_t)$ outputs a mean $\mu_{\phi}(s_t)$ and a standard deviation $\sigma_{\phi}(s_t)$ thus the actions are distributed Gaussian distribution, $a_t \sim N(\mu_{\phi}(s_t),\sigma_{\phi}(s_t))$. However, because Eq. (35) cannot be backpropagated in the normal scheme to compute $J_{\pi}(\phi)$, the *reparameterization trick* is adopted. Given state s_t, the squashed Gaussian policy is used; thus, the action is sampled according to:

$$\tilde{a}^{\phi}_t(s_t,\xi_t) = \tanh\left(\mu_{\phi}(s_t) + \sigma_{\phi}(s_t)\odot\xi_t\right), \quad \xi_t \sim N(0,I) \tag{36}$$

where ξ is the noise following the standard normal distribution. By adopting the reparameterization trick, the policy π_{ϕ} is optimized by minimizing the following actor cost function:

$$J_{\pi}(\phi) = \mathbb{E}_{\substack{s_t\sim D\\ \xi_t\sim N}}\left[\alpha\log\left(\pi_{\phi}(\tilde{a}^{\phi}_t(s_t,\xi_t)|s_t)\right) - \min_{j\in\{1,2\}} Q^{soft}_{\bar{\theta}_j}(s_t,\tilde{a}^{\phi}_t(s_t,\xi_t))\right]. \tag{37}$$

Instead of deciding the fixed value for temperature parameter α, the α can be learned by optimizing the following objective (Haarnoja et al., 2018b):

$$J(\alpha) = \mathbb{E}_{\substack{s_t\sim D\\ a_t\sim\pi_{\phi}(\cdot|s_t)}}\left[-\alpha\left(\log\pi_{\phi}(a_t|s_t) + \bar{\mathcal{H}}\right)\right] \tag{38}$$

where $\bar{\mathcal{H}}$ is a constant representing the target entropy.

4.2. SAC for discrete action space and prioritized experience replay

The SAC algorithm was developed to derive a near-optimal policy in continuous action settings (Haarnoja et al., 2018a,b). Our problem considers discrete action settings because the decision variables are integers. Even though the SAC outputs continuous

actions, they can be discretized, either by splitting them into a set of bins or simply by rounding them into integer values (Kanervisto et al., 2020). However, if we adopt the above scheme, we will ignore the following advantages of a discrete action setting: discrete action space has finite available actions. Because of this property, the action probability for each action ($p(a_t|s_t)$) can be directly calculated; thus, the variance of estimating $V_{\bar{\theta}}(s_{t+1})$, $J_\pi(\phi)$, and $J(\alpha)$ the can be reduced. Therefore, we refine the SAC algorithm to adjust in a discrete action setting based on the approach introduced by Christodoulou (2019).

In a continuous setting, $\pi_\phi(\cdot|s_t)$ is a probability density function; however, it is now a probability mass function. To revise the SAC algorithm, the following two changes should be considered:

- $Q^{soft} : \mathcal{S} \times \mathcal{A} \to \mathbb{R} \implies Q^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$: Unlike continuous action spaces, which have infinitely many possible actions, there are a limited number of possible actions in discrete actions paces. Therefore, the soft Q-function can be changed as a mapping $Q^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$ from a state to a vector containing the Q-value of each possible action.
- $\pi : \mathcal{S} \to \mathbb{R}^{2|\mathcal{A}|} \implies \pi : \mathcal{S} \to [0,1]^{|\mathcal{A}|}$: In a continuous setting, π outputs the mean and variance of action distribution. On the other hand, the probability for each action can be directly computed because there are finite possible actions in a discrete setting. By applying a softmax function on the output layer in the neural network of π_ϕ, the policy outputs a vector containing the probability of each action.

Due to the above two changes, cost functions of critic $J_{Q^{soft}}(\theta)$, actor $J_\pi(\phi)$, and temperature $J(\alpha)$ should be revised. In terms of critic cost function $J_{Q^{soft}}(\theta)$, the expectation value of $V_{\bar{\theta}}(s_{t+1})$ (Eq. (34)) can be computed directly because the probability for each possible action can be obtained instead of forming a Monte-Carlo estimate. Through this modification, the variance for the estimate of critic cost function $J_{Q^{soft}}(\theta)$ can be reduced. The soft value function $V_{\bar{\theta}}(s_{t+1})$ can be obtained by applying the following equation:

$$V_{\bar{\theta}}(s_{t+1}) = \pi_\phi(s_{t+1})^\intercal \left(\min_{j\in\{1,2\}} Q^{soft}_{\bar{\theta}_j}(s_{t+1}, a_{t+1}) - \alpha \log \pi_\phi(a_{t+1}|s_{t+1}) \right). \tag{39}$$

In a continuous spaces setting, the reparameterization trick is used to optimize actor cost function $J_\pi(\phi)$, so the soft critic cost function is transformed from Eqs. (35) to (37). However, in a discrete spaces setting, the expectation can be calculated directly in Eq. (35) regarding the policy $\pi_\phi(\cdot|s_t)$. Therefore, we do not need the reparameterization trick, and the new actor cost function is defined as:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}\left[\pi_\phi(s_t)^\intercal \left(\alpha \log\left(\pi_\phi(s_t)\right) - \min_{j\in\{1,2\}} Q^{soft}_{\bar{\theta}_j}(s_t) \right) \right]. \tag{40}$$

Similarly, the temperature cost function $J(\alpha)$ is changed from Eq. (38) to the following equation because the probability for each action can be computed:

$$J(\alpha) = \mathbb{E}_{s_t\sim D}\left[\pi_\phi(s_t)^\intercal \left(-\alpha \left(\log \pi_\phi(a_t|s_t) + \bar{\mathcal{H}} \right) \right) \right]. \tag{41}$$

In summary, we utilize four neural networks for critic (i.e., θ_i and $\bar{\theta}_i, i = 1,2$) and one neural network for actor (i.e, ϕ). Each neural network has an input layer, at least one hidden layer and an output layer sequentially. Its input is the state vector, and the output is the $|\mathcal{A}|$-dimensional action vector consisting of unnormalized scores, which is called logits. In particular, an actor neural network converts the logits into an action probability distribution using the following softmax function:

$$f(a_i) = \frac{\exp(a_i)}{\sum_{j=1}^{|\mathcal{A}|} \exp(a_j)}. \tag{42}$$

As mentioned in Section 3.2, the valid actions (i.e., available transshipment quantities) in action spaces $\mathcal{A}$ are different depending on the current state s_t. To prevent sampling the invalid action in $\mathcal{A}$, we employ the following *action masking* technique (Huang and Ontañón, 2020):

1. The current inventory level in online and offline channels, $IL^{ON}_{m,t}$ and $IL^{OF}_{m,t}$, are observed. We check the invalid transshipment quantity by comparing the sum of the current inventory level, $\sum_{m=1}^{M^{ON}-1} IL^{ON}_{m,t}$ and $\sum_{m=1}^{M^{OF}-1} IL^{OF}_{m,t}$, and maximum transshipment quantity, z^{ON}_{max} and z^{OF}_{max}.
2. A large negative number replaces the logit of actions corresponding to the invalid transshipment quantity.
3. The action probability can be obtained by inputting the logit of actions into the softmax function, and the probability of invalid actions will become ϵ, which should be a minimal number.

Due to the property of the off-policy algorithm, the SAC can use the past experiences, (s_t, a_t, r_t, s_{t+1}), which are stored in replay buffer D. Experiences can be sampled uniformly from a replay buffer without considering the importance of each experience. Even though this scheme stabilized the training process of DRL, it could impede sampling efficiency because important and unimportant experiences are replayed at the same frequency. Therefore, we employ the *prioritized experience replay* (PER), the method that prioritizes more important experiences by measuring the priority value of each experience using the magnitude of its temporal-difference (TD) error (Schaul et al., 2015). The TD error of experience $d \in D$, $|\delta_d|$, is defined using the soft value function, $V_{\bar{\theta}}$, and two soft Q-networks, $Q^{soft}_{\theta_1}$ and $Q^{soft}_{\theta_2}$, as follows:

$$|\delta_d| = \min\left\{ \left(Q^{soft}_{\theta_1}(s) - \left(r + \gamma V_{\bar{\theta}}(s')\right)\right)^2, \left(Q^{soft}_{\theta_2}(s) - \left(r + \gamma V_{\bar{\theta}}(s')\right)\right)^2 \right\}. \tag{43}$$

The priority value of each experience, p_d, is defined according to:

$$p_d = |\delta_d| + \epsilon_{per} \tag{44}$$

where ϵ_{per} is a small positive constant that prevents the case that the probability of revisiting the experience is zero. Then, the probability of sampling experience d is defined as:

$$P(d) = \frac{p_d^{\eta}}{\sum_{k=1}^{N_D} p_k^{\eta}} \tag{45}$$

where η is the prioritization factor that determines how much prioritization is used, and N_D is the size of replay buffer $\mathcal{D}$.

Usually, the PER can cause inevitable bias because it changes the distribution of expectations in an uncontrolled way. Therefore, we correct this bias by utilizing the following importance sampling weights, w_d:

$$w_d = \left(\frac{1}{N_D} \times \frac{1}{P(d)}\right)^{\beta} \tag{46}$$

where $\beta \in [0, 1]$. In addition, we normalize the above weights by $1/\max_d w_d$, due to stability reasons, and apply the importance of sampling weights to update neural networks. Finally, the SAC algorithm with PER for discrete action setting (SACDPE) is presented in Appendix A.

4.3. Two acceleration methods in the hybrid DRL approach: SQLT policy and reward shaping

Even though the SACDPE could get a promising policy, it suffers from unstable performance because of relatively large action spaces. The output layer of critic ($\theta_i, \bar{\theta}_i,\ i = 1, 2$) and actor neural networks (ϕ) composed of $|\mathcal{A}|$ nodes correspond to the number of available actions. Therefore, the large action spaces lead to neural networks with many parameters to train and many nodes in the output layer, resulting in considerable training time (Boute et al., 2021). In addition, the large action space increases the computational burden for the exploration strategy.

To mitigate the above issues and improve the performance of SACDPE, we developed the following two methods to accelerate the SACDPE: (1) SQLT policy and (2) RS. First, in the SQLT policy, we split decision-making for ordering (y_t^{ON} and y_t^{OF}) and transshipment (z_t^{ON} and z_t^{OF}) quantity into two stages, as shown in Fig. 3. In the original SACDPE, these four types of decisions are made simultaneously, which makes the action space extraordinarily large. Instead, we decide the transshipment quantity by relying on the DRL algorithm (SACDPE) in the first stage, and then the order quantity is decided through a specific ordering policy in the second stage. Note that this two-stage approach does not violate the sequence assumption (b). Even though the developed approach decouples transshipment decisions from ordering decisions, four types of decisions are made before the random demand is realized (i.e., the start of the period t).

Some readers may wonder why the decision on the order quantity is not made before the decision on the transshipment quantity. This sequence of decisions could result in excessive orders because the order quantity is determined solely without considering transshipment quantity. On the other hand, if the decision on the transshipment quantity is made before the order quantity, as shown in Fig. 3, the transshipment information can be reflected to determine the order quantity by utilizing the developed new ordering policy (i.e., SQLT policy). By separating the ordering decision from actions in the SACDPE, the size of the action space is reduced from $\left(y_{max}^{ON}+1\right) \times \left(y_{max}^{OF}+1\right) \times \left(z_{max}^{ON}+1\right) \times \left(z_{max}^{OF}+1\right)$ to $\left(z_{max}^{ON}+1\right) \times \left(z_{max}^{OF}+1\right)$. Consequently, the number of nodes at the output layer in the neural network is reduced; thus, the training time could be reduced significantly.

In the first stage decision, the current state s_t is observed, and the action for transshipment a_t is decided by policy $\pi_{\phi}(\cdot|s_t)$. In the second stage, any ordering policy, such as the base-stock policy, can be used for the second stage decision. However, we developed the SQLT policy by improving the SQmax policy (Haijema and Minner, 2016) and reflecting the information about transshipment decisions in the prior stage. The order quantity of SQLT policy is decided as follows:

$$y_t^{ON} = \min\left\{\left(S^{ON} - q_t^{ON}\left(z_t^{ON}, z_t^{OF}\right)\right)^+, Q_{max}^{ON}, y_{max}^{ON}\right\}, \tag{47}$$

$$y_t^{OF} = \min\left\{\left(S^{OF} - q_t^{OF}\left(z_t^{ON}, z_t^{OF}\right)\right)^+, Q_{max}^{OF}, y_{max}^{OF}\right\} \tag{48}$$

where S^{ON} and S^{OF} are parameters for the base-stock level, and Q_{max}^{ON} and Q_{max}^{OF} are parameters for a maximum order quantity in online and offline channels. Because our problem deals with integer values for demand and variables, we find the optimal value of these parameters through a *grid search*. The functions $q_t^{ON}\left(z_t^{ON}, z_t^{OF}\right)$ and $q_t^{OF}\left(z_t^{ON}, z_t^{OF}\right)$ are defined according to:

$$q_t^{ON}\left(z_t^{ON}, z_t^{OF}\right) = \sum_{m=1}^{M^{ON}-1} IL_{m,t}^{ON} + \sum_{l=1}^{L^{ON}-1} OT_{l,t}^{ON} - z_t^{ON} + z_t^{OF}, \tag{49}$$

$$q_t^{OF}\left(z_t^{ON}, z_t^{OF}\right) = \sum_{m=1}^{M^{OF}-1} IL_{m,t}^{OF} + \sum_{l=1}^{L^{OF}-1} OT_{l,t}^{OF} - z_t^{OF} + z_t^{ON}. \tag{50}$$

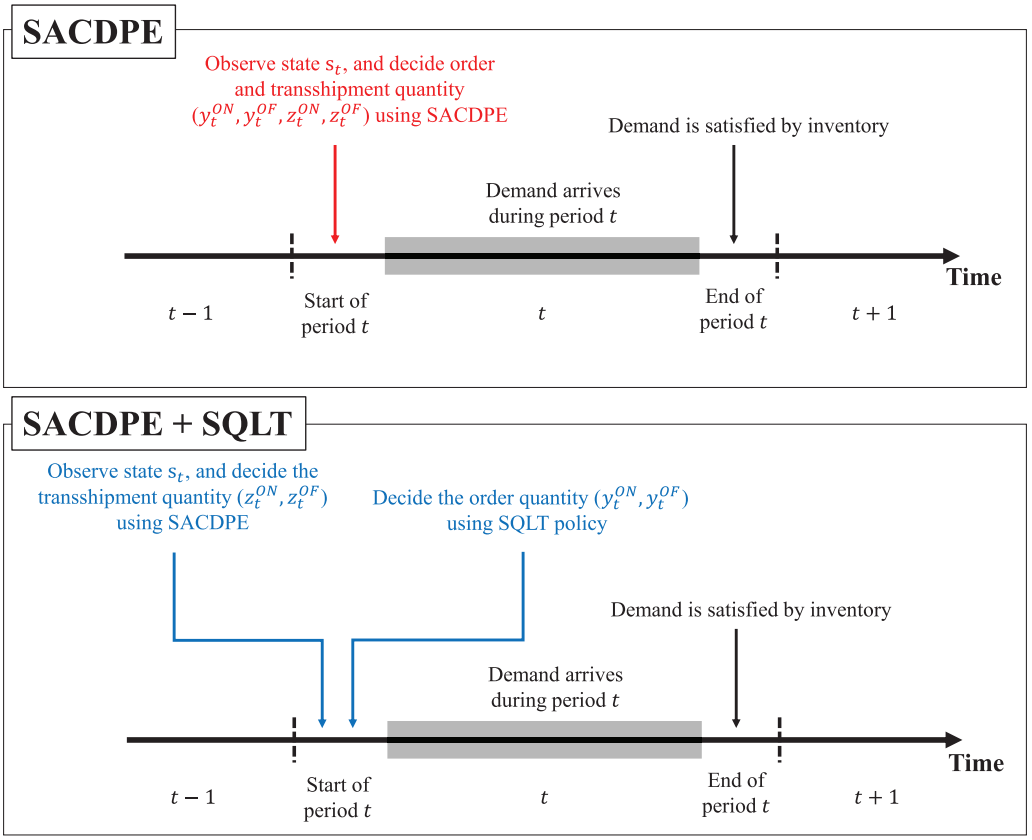


Fig. 3. Differences between original SACDPE and SACDPE+SQLT regarding the decision process. The differences between the two algorithms are highlighted in red and blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In Fig. 4, we present the architectures of neural networks utilized in SACDPE and SACDPE+SQLT, and these architectures are used for both critic and actor neural networks. The neural network architecture presented by Christodoulou (2019) is used for SACDPE. The presented neural networks consist of input, hidden, and output layers. There are $|\mathcal{S}|$ nodes in the input layer, and each node inputs the value of each state variable. Between the input and output layers, hidden layers exist in the networks, and the number of layers and nodes are determined by referring to the settings of the related studies (Schaul et al., 2015; Vanvuchelen et al., 2020; Sun et al., 2022). Output layers consist of $|\mathcal{A}|$ nodes, which correspond to the number of available actions, and have different activation functions for critic and actor neural networks. A critic neural network outputs the value of the input value of each output node without converting processes. In contrast, an actor neural network converts the input value into an action probability $(p(a_t|s_t))$ by utilizing the softmax function. Because SACDPE and SACDPE+SQLT have different sizes of $|\mathcal{A}|$, the number of output nodes in neural networks are different for each algorithm (i.e., SACDPE $\rightarrow (y_{max}^{ON}+1)\times(y_{max}^{OF}+1)\times(z_{max}^{ON}+1)\times(z_{max}^{OF}+1)$ and SACDPE+SQLT $\rightarrow (z_{max}^{ON}+1)\times(z_{max}^{OF}+1)$). Regarding the architecture of neural networks, the difference between both algorithms exists in the output layer, as presented in Fig. 4.

In addition to the above SQLT policy, we implement RS to define the more appropriate reward function to maximize the average profit. The RS is a technique to incorporate the exterior knowledge of a teacher heuristic into RL; thus, agents are guided towards more promising policies (De Moor et al., 2022; Zhu et al., 2020). In this research, we employ the SQmax policy as a teacher heuristic. The two same environments are declared; one for the RL, ENV_{RL}, and the other for a teacher heuristic, the SQmax policy, ENV_{SQmax}. At each time step of the training process, the values of realized demand in two environments are equal However, the current state and next state are different (i.e., (s_t, s_{t+1}) is obtained from ENV_{RL}, and $(\hat{s}_t, \hat{s}_{t+1})$ is obtained from ENV_{SQmax}). Even though several methods exist in the RS research area, we could get better solutions by just subtracting the profit of the SQmax policy from the profit of the RL as follows:

$$r(s_t, a_t) = PF_t^{RL} - PF_t^{SQmax} \tag{51}$$

where PF_t^{RL} is obtained profit at period t by the RL approach, and PF_t^{SQmax} is obtained profit at period t by the SQmax policy. Intuitively, the value of PF_t^{SQmax} is used as the criteria for assessing the quality of decisions implemented by RL at period t. The hybrid DRL approach, SACDPE combining the SQLT policy and RS (SACDPE+SQLT+RS), is presented in Algorithm 1.

Algorithm 1 Hybrid DRL approach (SACDPE+SQLT+RS)

Initialize $Q_{\theta_1}^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, $Q_{\theta_2}^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, $\pi_\phi : \mathcal{S} \to [0,1]^{|\mathcal{A}|}$
Initialize $Q_{\bar{\theta}_1}^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, $Q_{\bar{\theta}_2}^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, $\mathcal{D} \leftarrow \emptyset$
$\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$
Declare the environment for SACDPE (ENV_{RL})
Declare the environment for SQmax policy (ENV_{SQmax})
$e \leftarrow 1$
for *each episode* $e = 1, \cdots, E$ **do**
  $t \leftarrow 1$
  for *each timestep* $t = 1, \cdots, T$ **do**
    Observe s_t and choose action for transshipment $a_t \sim \pi_\phi(\cdot|s_t)$
    Determine y_t^{ON} and y_t^{OF} by SQLT policy
    Observe PF_t^{RL} and s_{t+1} from ENV_{RL}
    Observe state $\hat{s}_t$
    Determine y_t^{ON} and y_t^{OF} by SQmax policy
    Observe PF_t^{SQmax} and $\hat{s}_{t+1}$ from ENV_{SQmax}
    $r_t = PF_t^{RL} - PF_t^{SQmax}$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$ with maximal priority $p_t = \max_{i<t} p_i$
    Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$ according to probability $P(d) = p_d^\eta / \sum_{k=1}^{N_D} p_k^\eta, \forall d \in \mathcal{D}$
    $\Delta\theta_1, \Delta\theta_2, \Delta\phi, \Delta\alpha = 0$
    for $b \in \mathcal{B}$ **do**
      $w_b = \left(\frac{1}{N_D} \times \frac{1}{P(d)}\right)^\beta / \max_{i\in\mathcal{B}} w_i$
      $|\delta_b| = \min\left\{\left(Q_{\theta_1}^{soft}(s) - (r + \gamma V_{\bar{\theta}}(s'))\right)^2, \left(Q_{\theta_2}^{soft}(s) - (r + \gamma V_{\bar{\theta}}(s'))\right)^2\right\}$
      $p_b \leftarrow |\delta_b| + \epsilon_{per}$
      $\Delta\theta_i \leftarrow \Delta\theta_i + w_b \nabla_{\theta_i} J_{Q^{soft}}(\theta_i)$, for $i \in \{1,2\}$
      $\Delta\phi \leftarrow \Delta\phi + w_b \nabla_\phi J_\pi(\phi)$
      $\Delta\alpha \leftarrow \Delta\alpha + \nabla_\alpha J(\alpha)$
    end
    Update soft Q networks (critic) $\theta_i \leftarrow \theta_i - \lambda\Delta\theta_i$, for $i \in \{1,2\}$
    Update policy network (actor) $\phi \leftarrow \phi - \lambda\Delta\phi$
    Adjust temperature $\alpha \leftarrow \alpha - \lambda\Delta\alpha$
    Update target soft Q networks $\bar{\theta}_i \leftarrow \psi\theta_i + (1-\psi)\bar{\theta}_i$, for $i \in \{1,2\}$
    $t \leftarrow t + 1$
  end
  $e \leftarrow e + 1$
end
Return: θ_1, θ_2, π_ϕ

5. Computational experiments

Throughout this section, we conduct three types of computational experiments to address research questions 1, 2, and 3. In Section 5.1, we evaluate the performance of the developed hybrid DRL approach by conducting ablation studies to assess the effectiveness of two acceleration methods and by comparing it with existing algorithms on the real-world demand data set. In Section 5.2, we demonstrate the advantages of transshipment in the OOCS by examining different types of demand and varying the unit transshipment cost parameter. In Section 5.3, we examine the outdating costs associated with online and offline channels, respectively. On the basis of the results of the experiment, we suggest several managerial insights in Section 5.4. All computational experiments were implemented on a PC with an AMD Ryzen 7 PRO 4750G with a Radeon Graphics 3.60 GHz processor and 16 GB of RAM with Windows 10 64-bit. In addition, all experiments for the hybrid DRL approach were coded in Python 3.8 and Pytorch 1.12.1.

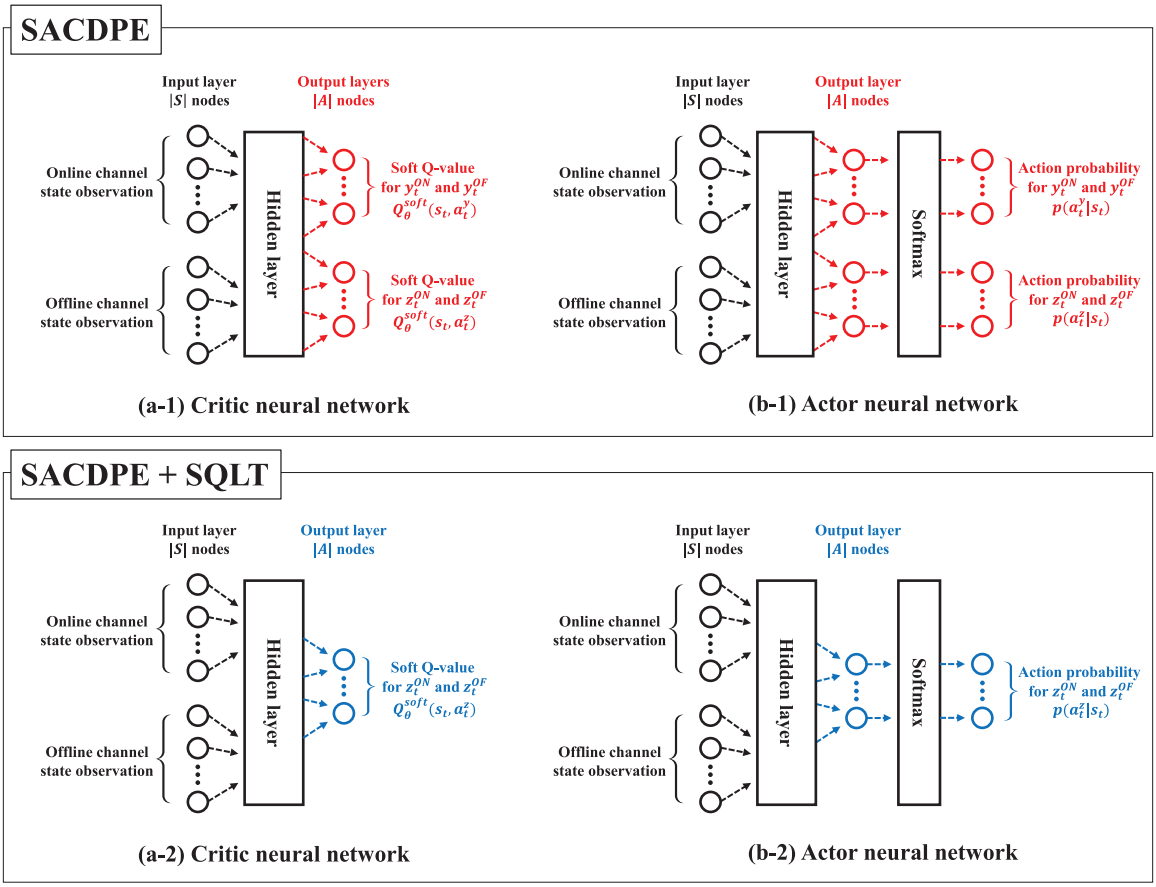


Fig. 4. The architectures of critic and actor neural networks. The differences between the SACDPE and SACDPE+SQLT are highlighted in red and blue. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.1. Performance analysis of the developed hybrid DRL approach for real-world data set

Two types of experiments were conducted in this section to validate the hybrid DRL approach, SACDPE+SQLT+RS: (1) validating the effects of accelerating approaches through ablation studies, and (2) comparing this hybrid DRL approach to existing algorithms. By referring the cost parameters in De Moor et al. (2022), we set the $c_p = 5$, $c_h = 1$, $c_w = 10$, and $c_o = 3$ for all instances. In this section, we consider the negligible transshipment cost, $c_l = 0$. The sale price in each channel p^{ON} and p^{OF} are set as 10 and 8 by accommodating a property that the sale price online is more expensive than offline. To address the practical size problem, we set the $M^{ON} = 5$, $M^{OF} = 7$, $L^{ON} = 2$, and $L^{OF} = 3$, which determines the dimensions of state. Moreover, we set the $y_{max}^{ON} = 10$, $y_{max}^{OF} = 10$, $z_{max}^{ON} = 5$, and $z_{max}^{OF} = 5$, which determines the size of action spaces. We consider 10,000 periods for the planning horizon to reflect the infinite horizon setting.

In this section, we examine the real-world data set presented in Oroojlooyjadid et al. (2022) and Kaggle (2018) for demands in online and offline channels, and the demand data set is used directly for training the DRL approach without any assumptions about demand distribution. A total of six instances were considered based on three different types of demand (i.e., Category A, B, and C) for each channel. In each instance, there are two types of demand data sets: training data and test data. As presented in Fig. B.12, the training data consists of 5000 episodes, and the test data consists of 20 episodes, and each episode contains the demand information within the planning horizon (i.e., 10,000 periods). The training data is utilized for the training process in DRL, and the test data is utilized to assess the performance of the developed algorithms. In detail, the parameters for the actor (ϕ) and critic (θ) neural networks are updated iteratively on the training data set. After the training, the parameters in actor and critic neural networks are not updated again, and the trained actor neural network is used on the test data set with parameter values obtained in the training process.

It has been found that the VI algorithm has not been able to obtain optimal policy as a result of the high complexity of the proposed problem. Therefore, to evaluate the quality of the solution systematically, we used the average profit obtained from the optimal objective value of the integer programming (IP) model under perfect information conditions (i.e., the deterministic demand setting). Due to the fact that the demand for the planning horizon is already known in advance, all costs, with the exception of the

ordering cost, are close to zero. Based on Dehghani et al. (2021), we developed the IP model using transshipment and replenishment as decision variables. The average profit obtained from perfect information is regarded as the upper bound. We used Python 3.8 and the FICO Xpress Optimizer library to solve the IP model. Detailed explanations about the developed IP model are presented in the supplementary material.

5.1.1. Ablation study: Validating the effectiveness of the hybrid DRL approach

In the first experiment, we conducted ablation studies to verify the effectiveness of the proposed acceleration methods in our hybrid DRL approach. We compared four DRL algorithms: SACDPE, SACDPE+RS, SACDPE+SQLT, and SACDPE+SQLT+RS. The SACDPE did not accommodate any proposed acceleration methods, and SACDPE+SQLT and SACDPE+RS adopted the SQLT policy and RS, respectively. The SACDPE+SQLT+RS accommodated both acceleration methods. In order to analyze the robustness of DRL algorithms, each algorithm was implemented five times, which means that five actor neural networks π_ϕ per instance were trained with random weight initialization. We informally conducted hyperparameter tuning instead of conducting the advanced search proposed by previous studies (Gijsbrechts et al., 2022). We determined the values of hyperparameters by referring to the settings of the related studies (Schaul et al., 2015; Vanvuchelen et al., 2020; Sun et al., 2022). All experiments were conducted using the same values of hyperparameters as stated in Table B.10. It should be noted that the SAC algorithm, which is the base algorithm of our study, has the advantage of mitigating the brittleness of hyperparameter tuning compared to other RL algorithms (Haarnoja et al., 2018a,b). We implemented extensive experiments with varying hyperparameter values, but the performance was not affected significantly. In particular, the SACDPE+SQLT+RS was the most robust to different hyperparameter values compared to other algorithms.

Fig. 5 depicts the learning curves of different DRL algorithms in the training process. We focus our analysis on the performance of the hybrid DRL approach, SACDPE+SQLT+RS (red), compared with the SACDPE (blue), SACDPE+RS (orange), and SACDPE+SQLT (green). The shaded areas around learning curves describe a 95 percent confidence interval for five multiple runs. We trained each DRL algorithm for 5000 episodes, and one episode consists of 100 time periods. The SACDPE obtained the worst average profit and had the widest confidence interval among every DRL algorithm. The SACDPE+RS shows more stability during training than the SACDPE. Also, the average profit of SACDPE+RS converges to a higher value than SACDPE. The SACDPE+RS requires more than 2000 episodes for convergence of average profit. On the other hand, SACDPE+SQLT can learn a promising policy within 500 episodes. The SACDPE+SQLT+RS also requires about 500 episodes for convergence and obtains the best policy resulting in the highest profit among every DRL algorithm. Because the action space size was reduced by adopting SQLT, the SACDPE+SQLT and SACDPE+SQLT+RS could learn a promising policy within relatively short episodes compared to SACDPE and SACDPE+RS.

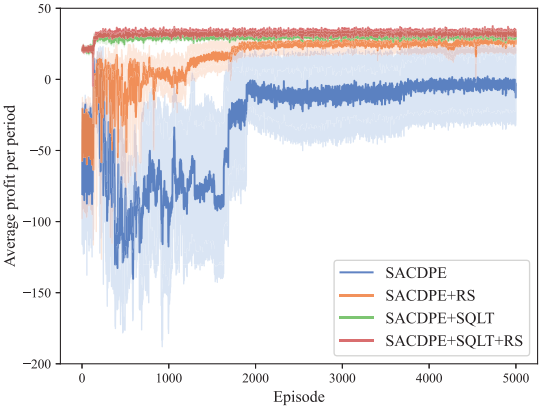
As shown in Table 2, experimental results are presented regarding the quality of solutions obtained with the developed DRL algorithms. The performance of the trained five actor neural networks was evaluated for each instance and DRL algorithm. Each actor neural network was tested for 20 different test demand data sets, and the sample mean and sample standard deviation of 20 runs were computed. The average of the sample mean and the average of standard deviation obtained from five actor neural networks were used for performance measures. As anticipated, the SACDPE+SQLT+RS outperformed other DRL algorithms, and the gap of average profit between SACDPE+SQLT+RS and the perfect condition is within 20 percent for all instances.

Through the ablation study, we confirmed that each acceleration method in our hybrid DRL approach has the following positive effects when combined with a pure DRL algorithm (i.e., SACDPE):

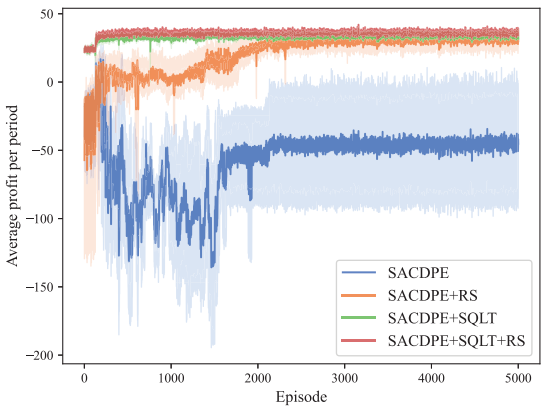
- *SQLT:* Because adopting the SQLT in the SACDPE reduces action spaces of MDP, the neural network architecture in DRL is changed. In detail, the SQLT reduces the number of output nodes in the output layer of neural networks; thus, the number of parameters required to be trained is also reduced. Because of this property, combining SQLT with SACDPE could derive high-quality policy within a shorter length of episodes and require less computation time for the training process on the given episode length than could DRL algorithms without SQLT. Furthermore, the SQLT has a small confidence interval for multiple runs, which means the stability of the training process is improved.
- *RS:* Adopting the RS in the SACDPE also improves the quality of policy derived from the DRL algorithms and the stability of the training process. In contrast to SQLT, DRL algorithms with RS have the same neural network architecture as that of the pure DRL algorithm because RS only changes the logic of the reward function. Therefore, the RS does not reduce the computation time for the training process and the required length of episodes to obtain a high-quality policy compared to the pure DRL algorithm.

5.1.2. Comparison between the hybrid DRL approach and existing approaches

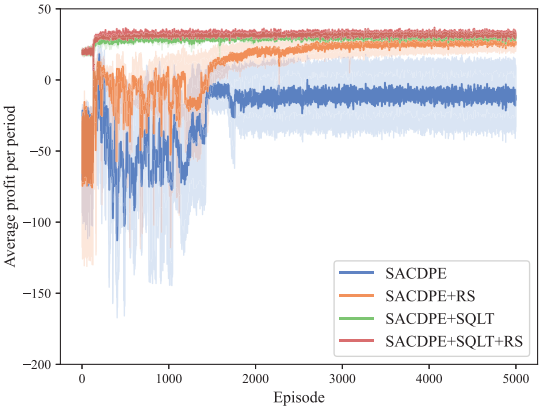
In the second experiment, we utilize four existing approaches that do not require prior knowledge of demand distribution to compare with the hybrid DRL approach. We began by adopting RH-2SSP presented by Dehghani et al. (2021). We revised the 2SSP model in Dehghani et al. (2021) to be suitable for our proposed problem, and the training data set was used for scenario samples. Like the developed DRL algorithm, RH-2SSP considers replenishment and transshipment decisions simultaneously. We present explanations about how we implemented RH-2SSP in the supplementary material. Second, we adopted three ordering policies based on an estimate of product waste to the order quantity presented in Haijema and Minner (2019): BSP-EW, BSPlow-EW, and SQmax-EW. To the best of our knowledge, BSPlow-EW is the state-of-the-art ordering policy for perishable products. Because only replenishment is considered as a decision in these three ordering policies, the order quantity in each channel is determined separately by predefined ordering policies.



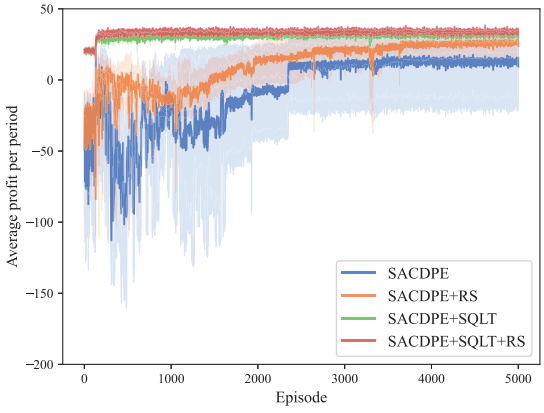
(a) Instance 1 (Online: A, Offline: B)



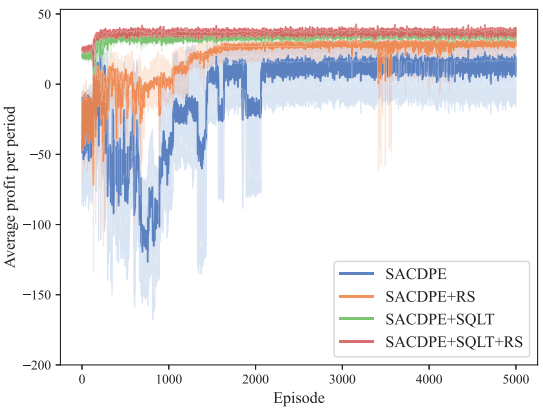
(b) Instance 2 (Online: A, Offline: C)



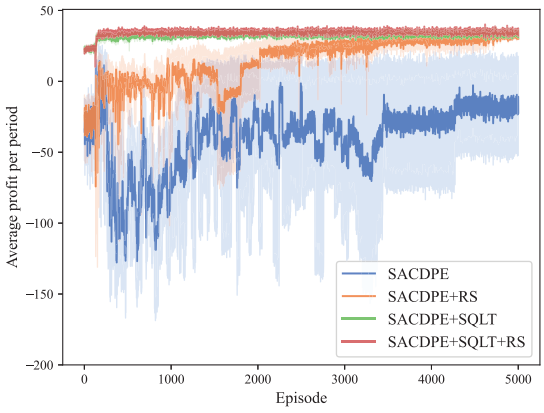
(c) Instance 3 (Online: B, Offline: A)



(d) Instance 4 (Online: B, Offline: C)



(e) Instance 5 (Online: C, Offline: A)



(f) Instance 6 (Online: C, Offline: B)

Fig. 5. Learning curves of different DRL algorithms in training process. The sold lines describe the average learning curves of multiple runs. The shaded areas around learning curves describe a 95% confidence interval. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In Table 3, we can see how existing approaches perform in comparison with each other. Even though RH-2SSP considers transshipment as a decision, the performance of RH-2SSP was poorer than the performance of other ordering policies. The solution

Table 2
Comparison between DRL algorithms considering the real-world data set.

Instance	Demand category		Perfect	SACDPE			SACDPE+RS			SACDPE+SQLT			SACDPE+SQLT+RS		
	Online	Offline		Mean	Std	Gap[a]	Mean	Std	Gap[a]	Mean	Std	Gap[a]	Mean	Std	Gap[a]
1	A	B	42.05	−3.65	27.44	108.68	28.76	19.53	31.60	33.68	18.47	19.89	35.08	18.03	**16.56**
2	A	C	45.21	−47.86	36.43	205.86	32.88	19.68	27.28	36.32	19.00	19.66	37.96	18.73	**16.04**
3	B	A	41.26	−11.13	26.28	126.99	28.11	17.85	31.88	32.93	17.16	20.19	34.41	16.99	**16.60**
4	B	C	42.46	13.08	20.65	69.19	27.32	18.98	35.65	34.34	17.00	19.11	35.70	16.81	**15.91**
5	C	A	45.69	13.31	26.74	70.87	30.78	18.83	32.62	37.41	18.66	18.13	38.60	18.54	**15.51**
6	C	B	43.72	−18.98	31.27	143.41	34.38	18.57	21.36	35.84	17.75	18.02	37.10	17.46	**15.15**

[a] Gap: (Perfect − Mean) × 100/Perfect.

Table 3
Comparison between existing approaches considering the real-world data set.

Instance	Demand category		Perfect	RH-2SSP			BSP-EW			BSPlow-EW			SQmax-EW		
	Online	Offline		Mean	Std	Gap[a]	Mean	Std	Gap[a]	Mean	Std	Gap[a]	Mean	Std	Gap[a]
1	A	B	42.05	32.77	16.30	22.06	33.49	19.01	20.35	33.67	18.48	**19.92**	33.67	18.48	**19.92**
2	A	C	45.21	34.13	16.28	24.51	36.01	19.90	20.36	36.37	18.71	**19.55**	36.37	19.31	**19.55**
3	B	A	41.26	31.01	14.97	24.83	32.61	18.39	20.97	32.97	17.21	**20.10**	32.97	17.20	**20.10**
4	B	C	42.46	31.63	14.33	25.50	34.05	17.88	19.79	34.41	16.44	**18.95**	34.41	17.13	**18.95**
5	C	A	45.69	35.17	16.12	23.03	36.51	19.76	20.09	36.89	18.71	**19.25**	36.89	18.71	**19.25**
6	C	B	43.72	34.64	15.54	20.77	35.44	18.36	18.94	35.64	17.78	**18.49**	35.64	17.78	**18.49**

[a] Gap: (Perfect − Mean) × 100/Perfect.

quality of RH-2SSP cannot be guaranteed because RH-2SSP is also one of the approximation methods for solving the problem with a long planning horizon. When compared to existing approaches, BSPlow-EW and SQmax-EW provided the same quality of solution and the best performance. However, we observed that the developed hybrid DRL approach, SACDPE+SQLT+RS, outperformed BSPlow-EW and SQmax-EW.

In terms of computational efficiency, SQLT reduces the computational burden of the training process. DRL algorithms without SQLT (SACDPE and SACDPE+RS) required about thirteen and a half hours to be trained in under 5000 episodes. In contrast, DRL algorithms that employ SQLT as an acceleration method (SACDPE+SQLT and SACDPE+SQLT+RS) required approximately six hours to be trained in under the same number of episodes. Despite the fact that DRL algorithms require several hours to train the first time, the trained neural networks can be reused and tested on a variety of demand data sets. In particular, every trained DRL algorithm required less than a second to be tested on one test data set. Existing rule-based approaches, BSP-EW, BSPlow-EW, and SQmax-EW, can be implemented on one test data set within a second. On the other hand, in the case of RH-2SSP, the 2SSP model is solved at every period because the algorithm is based on the rolling horizon approach. Thus, RH-2SSP required solving the 2SSP model 10,000 times to test on one data set, and it has low computational efficiency because it takes three and a half hours to implement just one time.

5.2. Advantages of transshipment on profit in the OOCS

In this section, we aim to analyze the advantages of transshipment by comparing it with no-transshipment policy. We adopt the SACDPE+SQLT+RS for a transshipment policy and the SQmax-EW for a no-transshipment policy. We set $L^{ON} = 3$, $L^{OF} = 3$, $y^{ON}_{max} = 20$, and $y^{OF}_{max} = 20$, and other parameters are equal to the parameter setting in Section 5.1. It should be noted that L^{ON} and L^{OF} were set as the same value because the differences between the lead time of online and offline channels could affect the average profit. The effectiveness of transshipment was evaluated by varying three key factors: (1) demand variability, (2) unit transshipment cost c_l, and (3) shelf life of product held in online and offline channels, M^{ON} and M^{OF}.

5.2.1. Effectiveness of transshipment on different types of demand

To begin with, we compare the transshipment and no-transshipment policies in terms of average profit for different types of demand. In this experiment, we assume that the transshipment cost is negligible. To demonstrate the effects of transshipment based on demand variability, we generated nine demand data sets as shown in Table 4. The determined parameters of the distributions are intended to cover cases of low, medium and high demand variability for each discrete probability distribution. For each demand data set, we trained the SACDPE+SQLT+RS for 2500 episodes three times (i.e., three actor neural networks). Based on the performance of three neural networks, the neural network that exhibited the most promising performance was selected for analysis.

Table 5 shows a comparison of average profits per period with respect to a transshipment policy compared to a no-transshipment policy. For every nine demand data sets, transshipment between online and offline channels resulted in a higher profit than no-transshipment. We use the performance measure ‘Gap(diff)’ to evaluate the effectiveness of a transshipment policy compared to a no-transshipment policy (i.e., the higher value of Gap(diff) represents that transshipment is more effective than no-transshipment). Under conditions of equal means, the variances of Uniform, Negative binomial, and Poisson distributions are listed in descending

Table 4
Information about distributions utilized to generate demand data sets.

	Distributions								
	Discrete uniform			Poisson			Negative binomial		
Notation	$U\{a_i, b_i\}$			$Pois(\lambda_i)$			$NB(n_i, p_i)$		
Parameters	$a_i = 0$			$\lambda_i = 5$	$\lambda_i = 8$	$\lambda_i = 10$	$p_i = 0.5$		
	$b_i = 10$	$b_i = 16$	$b_i = 20$				$n_i = 5$	$n_i = 8$	$n_i = 10$
Mean (μ)	5	8	10	5	8	10	5	8	10
Variance (σ^2)	10	24	36.67	5	8	10	10	16	20
CV[a]	0.63[b]	0.61[c]	0.60[d]	0.45[b]	0.35[c]	0.32[d]	0.63[b]	0.50[c]	0.45[d]

[a] CV: coefficient of variation (σ/μ).
[b] High demand variability.
[c] Medium demand variability.
[d] Low demand variability.

Table 5
Average profit of transshipment and no-transshipment policies for different types of demand data sets.

	Average profit per period								
	U{0, 10}	U{0, 16}	U{0, 20}	Pois(5)	Pois(8)	Pois(10)	NB(5,0.5)	NB(8,0.5)	NB(10,0.5)
No-transshipment	30.41	51.30	64.75	42.25	74.77	97.05	31.93	63.10	84.43
Transshipment	34.31	56.89	71.62	45.70	79.31	101.99	35.39	68.11	90.64
Gap(diff)[a]	**12.83**	**10.90**	**10.61**	**8.16**	**6.07**	**5.09**	**10.86**	**7.94**	**7.35**

[a] Gap(diff): (Transshipment – No-transshipment) × 100/No-transshipment.

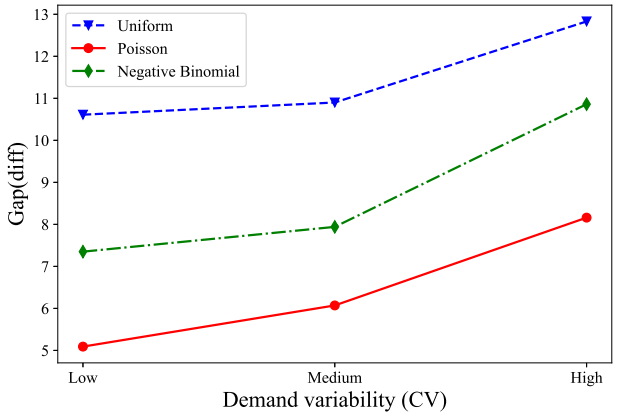


Fig. 6. Correlation between Gap(diff) and demand variability for three demand distributions.

order. Also, the Gap(diff) of Uniform, Negative binomial, and Poisson follows the same descending order, which indicates that transshipment is more effective when the variance of demand is greater.

In addition, we examined the correlation between Gap(diff) and demand variability for each distribution. Referring to several existing studies, we utilized the coefficient of variation (CV) to measure demand variability (Tagaras and Vlachos, 2002). For every distribution, Fig. 6 shows a tendency that the higher the value of CV, the more effective the transshipment is.

From the perspective of revenue and inventory holding, lost sales, outdating, and ordering costs, Fig. 7 illustrates how transshipment improves profitability. The y-axis represents the share of the total improvement according to different components (i.e., improvement percentage). Among all the components of the cost structure, transshipment contributes the most to reducing the outdating cost for every demand distribution. Also, the improvement percentage of inventory holding cost accounts for a relatively large share of total improvement. Due to the fact that the transshipment is implemented before the demand is realized, the holding cost can be saved instead of the transshipment cost, as shown in Eq. (18).

5.2.2. *Effectiveness of transshipment on different values of transshipment cost c_l*

In the second experiment, a sensitivity analysis on the unit transshipment cost parameter, c_l, was implemented. For ease of the expositions, we only consider a demand data set generated from $U\{0, 20\}$. As with experiments for different demand data sets, we also trained the SACDPE+SQLT+RS for 2500 episodes three times for every value of c_l, and we utilized the best one to assess the average profit of the transshipment policy. Table 6 shows the average profit of transshipment and no-transshipment policies

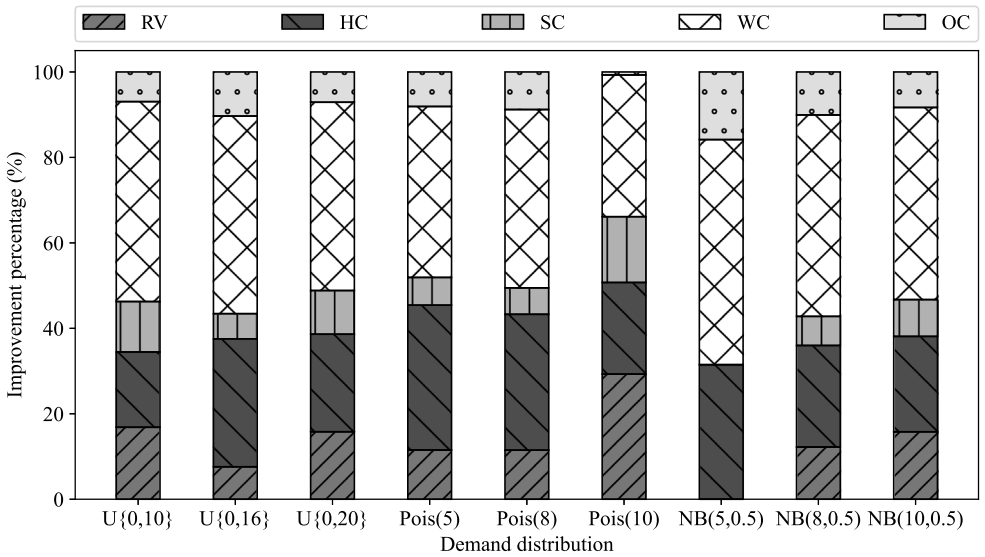


Fig. 7. Share of total improvement for revenue and each cost component for different demand distributions.

Table 6
Average profit of transshipment and no-transshipment policies varying the unit transshipment cost parameter c_l.

	Average profit per period									
	$c_l = 0$ (0.00)[b]	$c_l = 1$ (0.33)[b]	$c_l = 2$ (0.67)[b]	$c_l = 3$ (1.00)[b]	$c_l = 4$ (1.33)[b]	$c_l = 5$ (1.67)[b]	$c_l = 6$ (2.00)[b]	$c_l = 7$ (2.33)[b]	$c_l = 8$ (2.67)[b]	$c_l = 9$ (3.00)[b]
No-transshipment	64.75									
Transshipment	71.62	68.70	67.01	66.15	65.40	65.21	65.01	64.81	64.75	64.75
Gap(diff)[a]	**10.61**	**6.10**	**3.49**	**2.16**	**1.00**	**0.71**	**0.40**	**0.09**	**0.00**	**0.00**

[a] Gap(diff): (Transshipment – No-transshipment) × 100/No-transshipment.

[b] (c_l/c_o): the ratio of the unit transshipment cost to the unit order cost parameters.

varying the value of c_l. Transshipment can contribute to an increase in the average profit compared to no-transshipment when the c_l is between zero and seven. However, there is no advantage to using transshipment for maximizing profit if the c_l is bigger than seven.

Based on the experiment results in Table C.11, we depict Fig. 8 to show the improvement effect of transshipment on average profit varying the value of c_l. The sum of the improvement of revenue and cost will be the same as the improvement of profit. Similar to the results of Fig. 7, the transshipment improves the outdating cost the most compared to revenue and other cost components. If the transshipment cost is non-negligible ($c_l > 0$), transshipment could not contribute to saving inventory holding cost significantly, unlike the results of the first experiment considering negligible transshipment cost ($c_l = 0$). When the unit transshipment and inventory holding cost parameter is equal ($c_l = 1$), the inventory holding cost is reduced. However, in the case that c_l is bigger than c_h, the effect of transshipment to reduce the inventory holding cost was insignificant.

5.2.3. Effectiveness of transshipment on different shelf life of online and offline channels

In the third experiment, a sensitivity analysis on the shelf life of online and offline channels, M^{ON} and M^{OF}, was implemented. The experiment setting is equivalent to the second experiment except that the value of c_l is zero. Table 7 shows the average profit of transshipment and no-transshipment policies by varying the value of M^{ON} and M^{OF}. To analyze the effects of shelf life on the profit, we define two settings for the shelf life: "Short shelf life" ($M^{ON} = 3,\ M^{OF} = 5, 6, 7$) and "Long shelf life" ($M^{ON} = 5,\ M^{OF} = 7, 8, 9$). The transshipment was more effective in benefiting the average profit in the setting of Short shelf life than the it was in affecting the Long shelf life. Also, if the difference in the shelf life between channels was slight (i.e., $M^{OF} - M^{ON} = 2$), the positive effect of transshipment was insignificant compared to the case in which the difference was more considerable (i.e., $M^{OF} - M^{ON} = 3, 4$). In contrast, in the setting of Long shelf life, the variation of Gap (diff) was insignificant, even though the value of $M^{OF} - M^{ON}$ was changed. These results could be expected, because if the $M^{OF} - M^{ON}$ was equal to two, the transshipped product had a high risk of being outdated as indicated in Table D.12, owing to the non-negligible transshipment time.

5.3. Cost savings for outdated inventory by transshipment and heterogeneous shelf life

Observing the experiment results in Section 5.2, we found that transshipment reduces the outdating cost compared to a no-transshipment policy. To examine the impacts of transshipment on the OOCS, we first analyzed how many products were

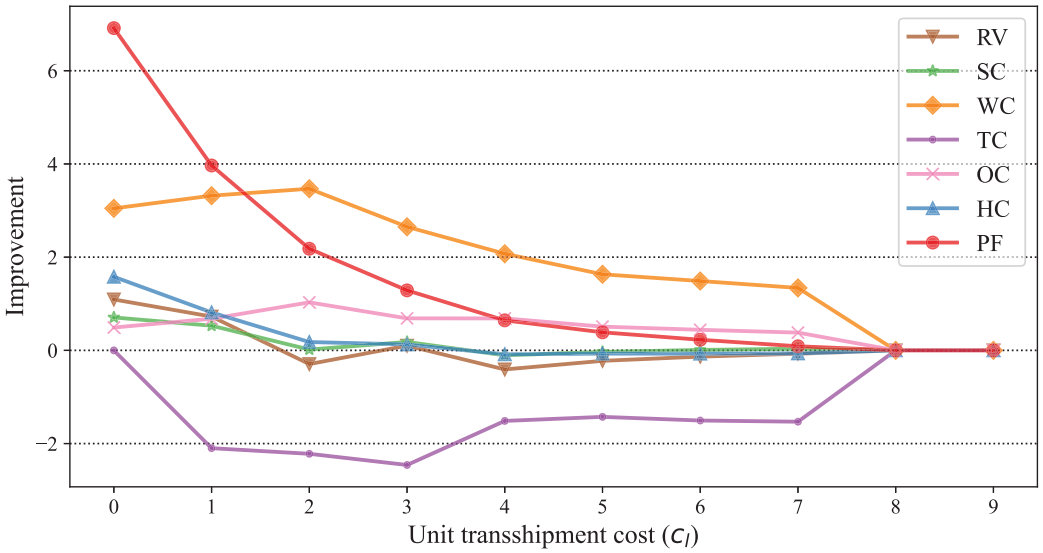


Fig. 8. Improvement effect of transshipment on average profit varying the unit transshipment cost c_l.

Table 7
Average profit of transshipment and no-transshipment policies varying the shelf life of product held in online and offline channels, M^{ON} and M^{OF}.

	Short shelf life			Long shelf life		
M^{ON}	3	3	3	5	5	5
M^{OF}	5	6	7	7	8	9
$M^{OF} - M^{ON}$	2	3	4	2	3	4
No-transshipment	64.85	66.35	67.07	78.51	78.81	78.89
Transshipment	71.54	73.73	74.35	85.61	86.10	85.96
Gap (diff)[a]	**10.32**	**11.13**	**10.86**	**9.03**	**9.24**	**8.96**

[a] Gap(diff): (Transshipment − No-transshipment) × 100/No-transshipment.

transshipped from one channel to the other. As a consequence, we identified the outdating cost that can be saved for each channel by using the transshipment on the OOCS. The same experiment setting is applied in Section 5.2 for the different types of demand and different values of c_l that will be analyzed.

Fig. 9 illustrates boxplots of the transshipment quantity according to different types of demand. The planning horizon of 10,000 periods results in 10,000 samples per boxplot. We add a mark for the mean values on boxplots using the white circle. For every type of demand, more products were transshipped from the online channel to the offline channel than transshipped from the offline channel to the online channel. As indicated in Table 5, there was no significant difference between the mean values of two types of transshipment quantities for the demand of Poisson distribution, where transshipment is the least effective among three distribution types. In contrast, we can observe that the mean value gap was large for uniform and negative binomial distributions, in which the transshipment is effective due to a high degree of demand variability.

Fig. 10 presents boxplots of transshipment quantity for varying c_l values generated from the demand data set $\mathrm{U}\{0, 20\}$. As the value of c_l increases, the total transshipment quantity decreases due to a high transshipment cost. If the value of c_l is smaller than eight, more products were transshipped from the online channel to the offline channel than transshipped from the offline channel to the online channel. When the value of c_l is larger than eight, the transshipment did not occur in both channels. Consequently, the average profit of transshipment and no-transshipment policies is equal, as shown in Table 6.

We inspect the outdating cost of each channel for transshipment and no-transshipment policies through Tables 8 and 9. For every experiment setting, we could observe that the outdating cost increased in the offline channel when utilizing transshipment compared to no-transshipment. However, the outdating cost decreased substantially in the online channel compared to the offline channel; thus, the transshipment can save the outdating cost from the standpoint of the total system. It has been found that this tendency is the result of two different factors:

- The shelf life of the online channel is shorter than that of the offline channel (heterogeneous shelf life property).
- It is evident from Figs. 9 and 10 that a greater number of products are transshipped from an online channel to an offline channel, rather than from an offline channel to an online channel.

We observe that products that must be discarded in the online channel were transshipped to the offline channel. Most of them were used to satisfy demand in the offline channel, and few products were abandoned in the offline channel. As can be observed

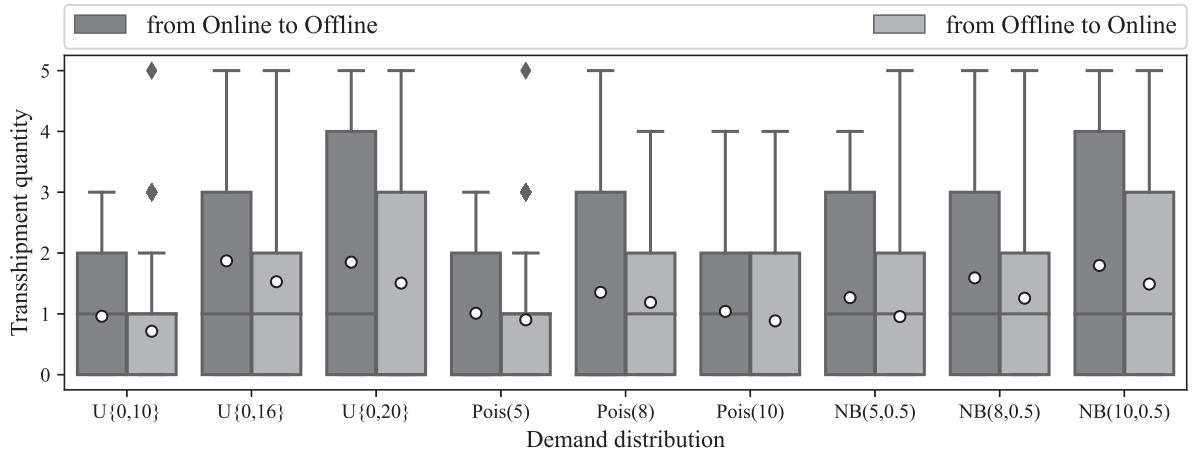


Fig. 9. Boxplots of transshipment quantity for different types of demand.

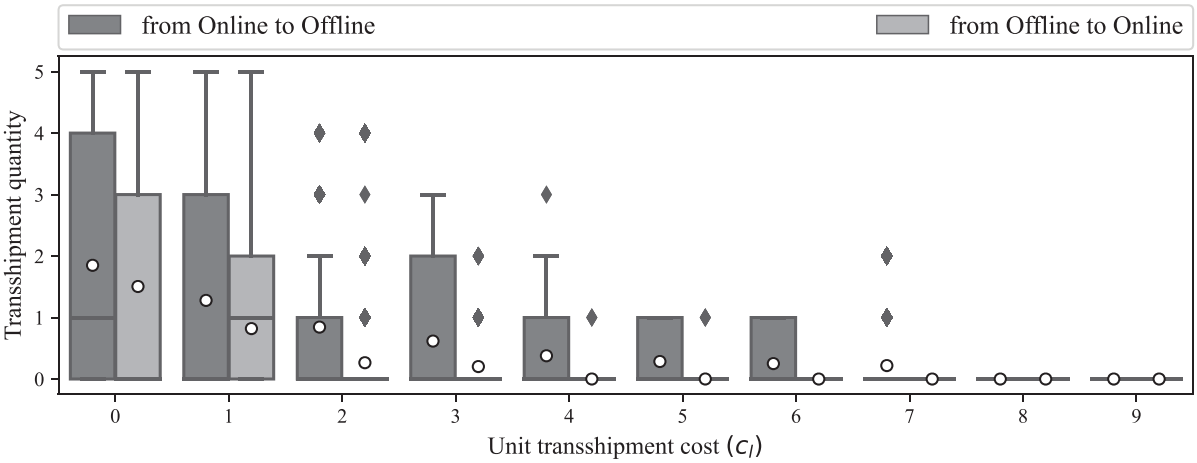


Fig. 10. Boxplots of transshipment quantity for varying the value of unit transshipment cost parameter c_l.

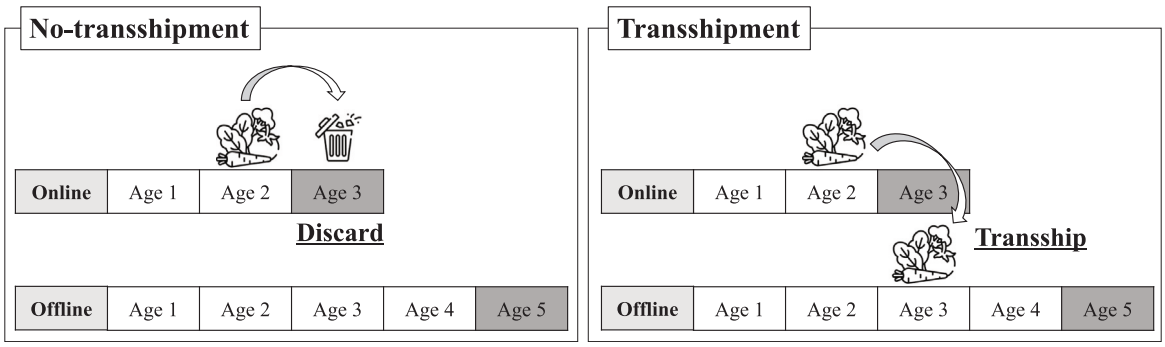


Fig. 11. Example of saving outdating cost in the OOCS through transshipment.

from Fig. 11, in the case of this experiment setting, $M^{ON} = 3$ and $M^{OF} = 5$, it is necessary to discard the product in the online channel once the product reaches the age of three. However, this product can be used in the offline channel because the product will be disposed of when the age is five in the offline channel. Therefore, when utilizing transshipment in the OOCS, we found that the offline channel, which has a longer shelf life, plays the role of making good use of the old product that will be discarded in the online channel if not transshipped to the offline channel.

Table 8
Analysis about the outdating cost of each channel in different types of demand data sets.

	Channel	U{0, 10}	U{0, 16}	U{0, 20}	Pois(5)	Pois(8)	Pois(10)	NB(5,0.5)	NB(8,0.5)	NB(10,0.5)
WC_{NT}[a]	ON	5.85	8.76	11.54	2.44	2.55	1.95	4.58	4.43	5.02
	OF	1.09	1.71	2.34	0.24	0.03	0.01	0.85	0.59	0.34
	ON+OF	6.95	10.47	13.88	2.68	2.57	1.95	5.43	5.02	5.35
WC_T[b]	ON	3.50	5.00	7.49	0.89	0.55	0.37	1.97	1.42	1.73
	OF	1.78	2.90	3.33	0.46	0.08	0.03	1.66	1.19	0.77
	ON+OF	5.27	7.90	10.83	1.35	0.63	0.40	3.63	2.61	2.50
Diff[c]	ON	**2.36**	**3.76**	**4.05**	**1.55**	**2.00**	**1.58**	**2.61**	**3.01**	**3.29**
	OF	**−0.68**	**−1.19**	**−1.00**	**−0.22**	**−0.06**	**−0.02**	**−0.81**	**−0.60**	**−0.43**
	ON+OF	1.67	2.57	3.05	1.33	1.94	1.55	1.80	2.41	2.85
Saving(%)[d]	ON+OF	**24.07**	**24.57**	**21.96**	**49.70**	**75.48**	**79.55**	**33.20**	**47.96**	**53.32**

[a] WC_{NT} : Outdating cost of no-transshipment policy (SQmaxEW).
[b] WC_T : Outdating cost of transshipment policy (SACDPE+SQLT+RS).
[c] Diff: Effects of transshipment on outdating cost ($WC_{NT} - WC_T$).
[d] Saving(%) : $\left(WC_{NT} - WC_T\right) \times 100/WC_{NT}$.

Table 9
Analysis about the outdating cost of each channel varying the unit transshipment cost parameter c_l.

	Channel	$c_l = 0$	$c_l = 1$	$c_l = 2$	$c_l = 3$	$c_l = 4$	$c_l = 5$	$c_l = 6$	$c_l = 7$	$c_l = 8$	$c_l = 9$
WC_{NT}[a]	ON	11.54									
	OF	2.34									
	ON+OF	13.88									
WC_T[b]	ON	7.49	7.32	7.30	8.33	8.96	9.56	9.77	9.99	11.54	11.54
	OF	3.33	3.24	3.11	2.90	2.85	2.69	2.61	2.54	2.34	2.34
	ON+OF	10.83	10.56	10.41	11.22	11.80	12.24	12.39	12.54	13.88	13.88
Diff[c]	ON	**4.05**	**4.22**	**4.24**	**3.21**	**2.58**	**1.98**	**1.77**	**1.55**	**0.00**	**0.00**
	OF	**−1.00**	**−0.90**	**−0.77**	**−0.56**	**−0.51**	**−0.35**	**−0.28**	**−0.21**	**0.00**	**0.00**
	ON+OF	3.05	3.32	3.47	2.65	2.07	1.63	1.49	1.34	0.00	0.00
Saving(%)[d]	ON+OF	**21.96**	**23.92**	**24.99**	**19.11**	**14.94**	**11.76**	**10.72**	**9.66**	**0.00**	**0.00**

[a] WC_{NT} : Outdating cost of no-transshipment policy (SQmaxEW).
[b] WC_T : Outdating cost of transshipment policy (SACDPE+SQLT+RS).
[c] Diff: Effects of transshipment on outdating cost ($WC_{NT} - WC_T$).
[d] Saving (%) : $\left(WC_{NT} - WC_T\right) \times 100/WC_{NT}$.

5.4. Managerial insights

According to the results of the experiment, we suggest the following managerial insights that are relevant to logistics practitioners who are concerned about setting up an effective transshipment policy within the OOCS:

- As a result of the rapid development of computational technology in recent years, multiple e-commerce companies have been able to secure lots of data about the historical demand for fresh foods. For companies that have an abundance of demand data, it is recommended that they utilize the data directly with the developed DRL approach to obtain a practical transshipment policy since the neural network of the DRL can be trained more accurately as more data is gathered. On the other hand, when the company does not have enough data to train a neural network of DRL, it is necessary to generate artificial data utilizing estimated demand distribution in order to train the neural network. Alternatively, it is also possible to obtain a transshipment policy using traditional methods, such as VI and heuristics, although they are difficult to apply in practice.
- In a condition in which the unit transshipment cost, c_l, is relatively small compared to the unit outdating cost, c_w, a transshipment policy is effective in increasing the average profit from the perspective of the total system by saving the outdating cost. In comparison with a no-transshipment policy, the transshipment could not contribute to ramping up the average profit if the value of c_l is not much smaller than the value of c_w (i.e., $c_l/c_w \approx 1$). Thus, we recommend that logistics managers estimate the accurate value of c_l and c_w before deciding whether to implement the transshipment policy within the OOCS.
- Although the hybrid DRL approach developed has proved to be effective in maximizing profit, training the DRL once takes several hours. A further reason for not relying on DRL's transshipment policy is that it does not follow a simple rule and is difficult to interpret. Therefore, several logistics practitioners have not been able to rely on DRL's transshipment policy in their business practices. Hence, if logistics managers do not have time to train the DRL from scratch and require an interpretable policy, a simple decision rule for transshipment could be considered. According to Figs. 9 and 10, in the OOCS with a heterogeneous shelf life, we can observe that more products are transshipped from the online channel (short shelf

life) to the offline channel (long shelf life) than from the offline channel to the online channel. As a result of this trend, if logistics practitioners do not have enough time or desire a more interpretable policy, developing a simple policy that only covers transshipments from online to offline could be acceptable in business practice.

6. Conclusions

Due to an increase in demand for fresh foods, e-commerce companies provide fresh food delivery services through their logistics platforms. Instead of relying solely on the online channel, the OOCS has attracted logistics practitioners who are responsible for managing fresh foods. The OOCS can contribute to reducing the risks of excess ordering and waste of fresh food through lateral transshipment between channels.

Motivated by the above real-world practice, we developed the lateral transshipment model for fresh food by accommodating the key attributes of the OOCS: heterogeneous shelf life, proactive transshipment, and non-negligible transshipment time. In the field of lateral transshipment research, the majority of studies focus on a specific distribution of demand to determine the policy of transshipment. Conversely, we seek to directly derive a transshipment policy based on demand data by developing the DRL approach based on the SAC algorithm, which does not need any assumptions about demand distribution.

Unfortunately, the action space of the proposed model is extraordinarily large because four types of decisions must be made simultaneously. In our experience, the DRL approach suffers from unstable performance during training, which is due to the difficult task of computing large action spaces in the DRL approach, and therefore requires considerable computation time. As a way to mitigate these issues, we developed a hybrid DRL approach that combines two novel acceleration methods: SQLT and RS, to create a hybrid DRL approach. First, we split the decision-making process into two stages. Transshipment decisions are handled by the DRL approach, while replenishment decisions are handled by the SQLT approach. Second, to enhance the performance of DRL, we implement the RS by adopting the SQmax policy as a teacher heuristic into DRL. By conducting computational experiments, we observed that adopting two acceleration methods enabled the training process to be stabilized and the average profit to be maximized.

We analyzed the impacts of transshipment in the OOCS by differing types of demand and varying the unit transshipment cost parameter and shelf life of online and offline channels. In line with our expectations, transshipment was more effective when demand variability was high. Transshipment could lead to an increase in average profit as a result of a substantial reduction in outdating cost, as compared to revenue and other components of the cost. Transshipment resulted in a slight increase in the outdating cost in the offline channel, compared to the case where there was no-transshipment. However, the outdating cost in the online channel was reduced substantially by implementing transshipment. Also, we found that more fresh foods are transshipped from online to offline channels than from offline to online channels. These findings suggest that the offline channel could be utilized to resell old products planned to be discarded in the online channel. Finally, we presented several managerial insights instructive to logistics practitioners who require a transshipment policy with the OOCS.

This study could serve as a starting point for future research related to the DRL approach to lateral transshipment of perishable products. Even though this study has focused on proactive transshipment, we expect that the proposed DRL approach could be applied to reactive transshipment by adding the observed demand to the state in the MDP. Moreover, by differing the types of demand and cost parameters, analyzing the effectiveness between proactive and reactive transshipment strategies for each case could provide better guidance and managerial insights for real-business operators.

However, we are aware that this research has several limitations to address in future research, as follows:

- The current study only considered two outlets. On the other hand, leading companies with OOCS commonly operate multiple online distribution centers and offline retail stores. Reflecting the distances of multiple outlet locations in cost parameters and developing the appropriate DRL approach could be important lines of future research.
- The DRL requires massive computational efforts to train neural networks for one instance. In order to systematically analyze the impacts of the proposed model, the trained neural networks of DRL need to be evaluated in different instances by varying values of parameters. However, several weeks or months are required to train neural networks from scratch for every different combination of parameters. Instead of training neural networks for DRL from scratch for each instance, future studies should target developing methods that reutilize neural networks completed training for another instance. By using this scheme, substantial reductions in the computational burden would be achieved (Oroojlooyjadid et al., 2022; Zhu et al., 2020).
- Under the given sale price scenario, the current study decided the order and transshipment quantities, which are general decisions in inventory management and transshipment research areas. However, including pricing as one of the decision variables will be more instructive to logistics practitioners and suitable for real-world operations. If pricing decisions for online and offline channels are included in decisions, our action dimension will increase from four to six (i.e., $a_t = (p_t^{ON}, p_t^{OF}, y_t^{ON}, y_t^{OF}, z_t^{ON}, z_t^{OF})$). Also, an action consists of both continuous and integer decisions because the sale price is typically considered as a continuous value. Therefore, utilizing DRL algorithms with discrete action settings is unsuitable for this future research because it has continuous decisions and a large size of action space. To overcome these issues, future studies should target the development of DRL algorithms with continuous action settings, which is a promising strategy for a large multi-discrete action space. Furthermore, continuous action values derived from the DRL algorithms can be used to determine the sale price as it is, and order and transshipment quantities can be determined by discretizing the continuous action value.

CRediT authorship contribution statement

Junhyeok Lee: Conceptualization, Data curation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Youngchul Shin:** Conceptualization, Investigation, Writing – review & editing. **Ilkyeong Moon:** Conceptualization, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Pseudocode for SACDPE

Algorithm 2 SACDPE.

Initialize $Q_{\theta_1}^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, $Q_{\theta_2}^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, $\pi_\phi : \mathcal{S} \to [0,1]^{|\mathcal{A}|}$
Initialize $Q_{\bar{\theta}_1}^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, $Q_{\bar{\theta}_2}^{soft} : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$, $\mathcal{D} \leftarrow \emptyset$
$\bar{\theta}_1 \leftarrow \theta_1$, $\bar{\theta}_2 \leftarrow \theta_2$
Declare the environment for SACDPE (ENV_{RL})
$e \leftarrow 1$
for *each episode* $e = 1, \cdots, E$ **do**
  $t \leftarrow 1$
  for *each timestep* $t = 1, \cdots, T$ **do**
    Observe s_t and choose action $a_t \sim \pi_\phi(\cdot|s_t)$
    Observe $r_t = PF_t^{RL}$ and s_{t+1} from ENV_{RL}
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1})\}$ with maximal priority $p_t = \max_{i<t} p_i$
    Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$ according to probability $P(d) = p_d^\eta / \sum_{k=1}^{N_D} p_k^\eta, \forall d \in \mathcal{D}$
    $\Delta\theta_1, \Delta\theta_2, \Delta\phi, \Delta\alpha = 0$
    for $b \in \mathcal{B}$ **do**
      $w_b = \left(\frac{1}{N_D} \times \frac{1}{P(d)}\right)^\beta / \max_{i \in \mathcal{B}} w_i$
      $|\delta_b| = \min\left\{\left(Q_{\theta_1}^{soft}(s) - \left(r + \gamma V_{\bar{\theta}}(s')\right)\right)^2, \left(Q_{\theta_2}^{soft}(s) - \left(r + \gamma V_{\bar{\theta}}(s')\right)\right)^2\right\}$
      $p_b \leftarrow |\delta_b| + \epsilon_{per}$
      $\Delta\theta_i \leftarrow \Delta\theta_i + w_b \nabla_{\theta_i} J_{Q^{soft}}(\theta_i)$, for $i \in \{1,2\}$
      $\Delta\phi \leftarrow \Delta\phi + w_b \nabla_\phi J_\pi(\phi)$
      $\Delta\alpha \leftarrow \Delta\alpha + \nabla_\alpha J(\alpha)$
    end
    Update soft Q networks (critic) $\theta_i \leftarrow \theta_i - \lambda\Delta\theta_i$, for $i \in \{1,2\}$
    Update policy network (actor) $\phi \leftarrow \phi - \lambda\Delta\phi$
    Adjust temperature $\alpha \leftarrow \alpha - \lambda\Delta\alpha$
    Update target soft Q networks $\bar{\theta}_i \leftarrow \psi\theta_i + (1-\psi)\bar{\theta}_i$, for $i \in \{1,2\}$
    $t \leftarrow t + 1$
  end
  $e \leftarrow e + 1$
end
Return: θ_1, θ_2, π_ϕ

Appendix B. Information about hyperparameters of the hybrid DRL approach and data description

See Table B.10 and Fig. B.12.

Appendix C. Improvement effects of transshipment varying the unit transshipment cost parameter

See Table C.11.

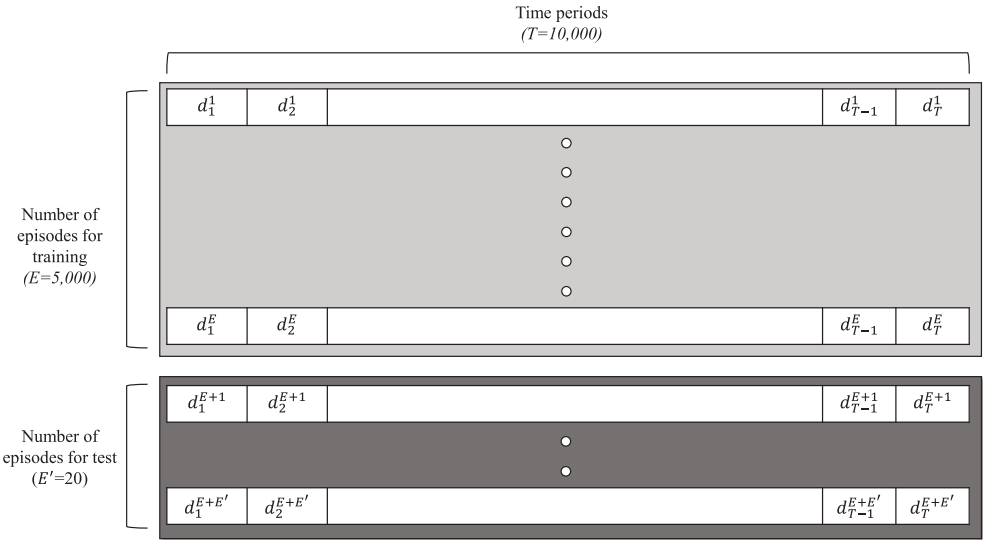


Fig. B.12. Data description.

Table B.10
Hyperparameters used for the hybrid DRL approach.

Hyperparameter name	Hyperparameter used
Size of the replay buffer N_D	200,000
Size of the minibatch $\|\mathcal{B}\|$	128
Soft update factor ψ	0.002
Prioritization factor η	0.6
Compensation factor β	0.4
Discount factor γ	0.99
Learning rate λ	0.0001
Hidden layer of neural networks	[64,64]
Optimizer	Adam
Activation function hidden layers	Relu
Activation function output layers	Softmax

Table C.11
Improvement effects of utilizing transshipment varying the unit transshipment cost parameter c_l.

	c_l	Measure	RV	HC	SC	WC	TC	OC	PF
No-transshipment	–	Average	162.53	16.62	9.40	13.88	0.00	58.49	64.14
Transshipment	0	Average	163.62	15.04	8.69	10.83	0.00	58.00	71.06
		Improvement[a]	1.09	1.58	0.71	3.05	0.00	0.49	6.91
	1	Average	163.25	15.80	8.87	10.56	2.10	57.81	68.11
		Improvement[a]	0.72	0.81	0.53	3.32	−2.10	0.68	3.97
	2	Average	162.23	16.44	9.38	10.41	2.22	57.46	66.33
		Improvement[a]	−0.29	0.18	0.02	3.47	−2.22	1.03	2.18
	3	Average	162.63	16.49	9.22	11.22	2.46	57.80	65.43
		Improvement[a]	0.10	0.13	0.18	2.65	−2.46	0.69	1.29
	4	Average	162.12	16.70	9.51	11.80	1.51	57.81	64.79
		Improvement[a]	−0.41	−0.08	−0.11	2.07	−1.51	0.69	0.64
	5	Average	162.30	16.69	9.43	12.24	1.43	57.98	64.53
		Improvement[a]	−0.22	−0.07	−0.03	1.63	−1.43	0.51	0.39
	6	Average	162.39	16.69	9.39	12.39	1.51	58.05	64.37
		Improvement[a]	−0.13	−0.07	0.01	1.49	−1.51	0.44	0.23
	7	Average	162.46	16.68	9.36	12.54	1.53	58.11	64.23
		Improvement[a]	−0.07	−0.07	0.04	1.34	−1.53	0.38	0.09
	8	Average	162.53	16.62	9.40	13.88	0.00	58.49	64.14
		Improvement[a]	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	9	Average	162.53	16.62	9.40	13.88	0.00	58.49	64.14
		Improvement[a]	0.00	0.00	0.00	0.00	0.00	0.00	0.00

[a] Improvement: (Transshipment – No-transshipment) for revenue and profit, and (No-transshipment – Transshipment) for cost components.

Table D.12
Improvement effects of utilizing transshipment varying the shelf life of product held in online and offline channels, M^{ON} and M^{OF}.

		Short shelf life			Long shelf life		
	M^{ON}	3	3	3	5	5	5
	M^{OF}	5	6	7	7	8	9
	$M^{OF} - M^{ON}$	2	3	4	2	3	4
		Average value per period					
RV	No-transshipment	161.40	162.31	162.84	168.34	167.79	167.79
	Transshipment	162.15	162.78	163.41	169.77	169.76	169.54
	Improvement[a]	0.75	0.47	0.57	1.42	1.97	1.76
HC	No-transshipment	15.99	17.24	17.96	23.43	23.08	23.10
	Transshipment	14.04	16.03	16.69	20.30	20.19	20.12
	Improvement[a]	1.95	1.21	1.26	3.12	2.90	2.98
SC	No-transshipment	10.03	9.46	9.12	6.37	6.65	6.65
	Transshipment	9.48	9.08	8.69	5.63	5.69	5.68
	Improvement[a]	0.55	0.37	0.43	0.74	0.96	0.97
WC	No-transshipment	13.23	11.91	11.36	3.02	2.53	2.48
	Transshipment	10.13	7.46	7.50	1.43	1.12	1.04
	Improvement[a]	3.10	4.45	3.86	1.60	1.41	1.44
OC	No-transshipment	57.92	57.87	57.90	57.05	56.73	56.71
	Transshipment	57.32	56.76	57.00	57.01	56.89	56.86
	Improvement[a]	0.60	1.11	0.90	0.03	−0.16	−0.15
PF	No-transshipment	64.24	65.84	66.51	78.48	78.79	78.85
	Transshipment	71.19	73.46	73.53	85.39	85.88	85.84
	Improvement[a]	6.95	7.61	7.02	6.92	7.08	7.00

[a] Improvement: (Transshipment − No-transshipment) for revenue and profit, and (No-transshipment − Transshipment) for cost components.

Appendix D. Improvement effects of transshipment varying the shelf life of online and offline channels

See Table D.12.

Appendix E. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.tre.2024.103576.

References

Abouee-Mehrizi, H., Berman, O., Sharma, S., 2015. Optimal joint replenishment and transshipment policies in a multi-period inventory system with lost sales. Oper. Res. 63 (2), 342–350.

Bertsekas, D., 2012. Dynamic Programming and Optimal Control: Volume I, vol. 1, Athena scientific.

Boute, R.N., Gijsbrechts, J., van Jaarsveld, W., Vanvuchelen, N., 2021. Deep reinforcement learning for inventory control: A roadmap. European J. Oper. Res..

Cheong, T., 2013. Joint inventory and transshipment control for perishable products of a two-period lifetime. Int. J. Adv. Manuf. Technol. 66 (9), 1327–1341.

Christodoulou, P., 2019. Soft actor-critic for discrete action settings. arXiv preprint arXiv:1910.07207.

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

Dehghani, M., Abbasi, B., 2018. An age-based lateral-transshipment policy for perishable items. Int. J. Prod. Econ. 198, 93–103.

Dehghani, M., Abbasi, B., Oliveira, F., 2021. Proactive transshipment in the blood supply chain: A stochastic programming approach. Omega 98, 102112.

Firouz, M., Keskin, B.B., Melouk, S.H., 2017. An integrated supplier selection and inventory problem with multi-sourcing and lateral transshipments. Omega 70, 77–93.

Gijsbrechts, J., Boute, R.N., Van Mieghem, J.A., Zhang, D.J., 2022. Can deep reinforcement learning improve inventory management? performance on lost sales, dual-sourcing, and multi-echelon problems. Manuf. Serv. Oper. Manage..

Glazebrook, K., Paterson, C., Rauscher, S., Archibald, T., 2015. Benefits of hybrid lateral transshipments in multi-item inventory systems under periodic replenishment. Prod. Oper. Manage. 24 (2), 311–324.

Gosavi, A., 2009. Reinforcement learning: A tutorial survey and recent advances. INFORMS J. Comput. 21 (2), 178–192.

Guo, T., Mei, Y., Tang, K., Du, W., 2023a. Cooperative co-evolution for large-scale multi-objective air traffic flow management. IEEE Trans. Evol. Comput..

Guo, T., Mei, Y., Tang, K., Du, W., 2023b. A knee-guided evolutionary algorithm for multi-objective air traffic flow management. IEEE Trans. Evol. Comput..

Haarnoja, T., Zhou, A., Abbeel, P., Levine, S., 2018a. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning. PMLR, pp. 1861–1870.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al., 2018b. Soft actor-critic algorithms and applications. arXiv preprint arXiv:1812.05905.

Haijema, R., Minner, S., 2016. Stock-level dependent ordering of perishables: A comparison of hybrid base-stock and constant order policies. Int. J. Prod. Econ. 181, 215–225.

Haijema, R., Minner, S., 2019. Improved ordering of perishables: The value of stock-age information. Int. J. Prod. Econ. 209, 316–324.

Hasselt, H., 2010. Double Q-learning. Adv. Neural Inf. Process. Syst. 23.

He, X., Yang, H., Hu, Z., Lv, C., 2022. Robust lane change decision making for autonomous vehicles: An observation adversarial reinforcement learning approach. IEEE Trans. Intell. Veh. 8 (1), 184–193.

Hong, S., Lee, J., 2022a. Competition in Korean dawn, same-day delivery market intensifies with new players jumping in. https://pulsenews.co.kr/view.php?year=2022&no=234432. (Online; accessed 23-November-2022).
Hong, S., Lee, S., 2022b. Overnight delivery race spills over to big and foreign names in Korea. https://pulsenews.co.kr/view.php?sc=30800028&year=2022&no=537584&mc=. (Online; accessed 23-November-2022).
Huang, S., Ontañón, S., 2020. A closer look at invalid action masking in policy gradient algorithms. arXiv preprint arXiv:2006.14171.
Jiang, Z.-Z., Kong, G., Zhang, Y., 2021. Making the most of your regret: Workers' relocation decisions in on-demand platforms. Manuf. Serv. Oper. Manage. 23 (3), 695–713.
Kaggle, 2018. Store item demand forecasting challenge. https://www.kaggle.com/c/demand-forecasting-kernels-only/overview. (Online; accessed 14-November-2022).
Kanervisto, A., Scheller, C., Hautamäki, V., 2020. Action space shaping in deep reinforcement learning. In: 2020 IEEE Conference on Games (CoG). IEEE, pp. 479–486.
Kang, W., Lee, J., 2022. Korea's overnight fresh food delivery firm oasis seeks kosdaq listing. https://pulsenews.co.kr/view.php?sc=30800028&year=2022&no=381446. (Online; accessed 23-November-2022).
Kara, A., Dogan, I., 2018. Reinforcement learning approaches for specifying ordering policies of perishable inventory systems. Expert Syst. Appl. 91, 150–158.
Lee, T., 2022. Retailers ditch dawn deliveries in favor of more profitable options. https://koreajoongangdaily.joins.com/2022/10/03/business/industry/korea-delivery-grocery/20221003171548624.html. (Online; accessed 23-November-2022).
Li, G., He, X., Zhou, J., Wu, H., 2019. Pricing, replenishment and preservation technology investment decisions for non-instantaneous deteriorating items. Omega 84, 114–126.
Li, Y., Liao, Y., Hu, X., Shen, W., 2020. Lateral transshipment with partial request and random switching. Omega 92, 102134.
Li, X., Sun, L., Gao, J., 2013. Coordinating preventive lateral transshipment between two locations. Comput. Ind. Eng. 66 (4), 933–943.
Li, Q., Yu, P., Du, L., 2021. Separation of perishable inventories in offline retailing through transshipment. Oper. Res..
Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.
Meissner, J., Senicheva, O.V., 2018. Approximate dynamic programming for lateral transshipment problems in multi-location inventory systems. European J. Oper. Res. 265 (1), 49–64.
Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning. In: International Conference on Machine Learning. PMLR, pp. 1928–1937.
Nakandala, D., Lau, H., Shum, P.K., 2017. A lateral transshipment model for perishable inventory management. Int. J. Prod. Res. 55 (18), 5341–5354.
Oroojlooyjadid, A., Nazari, M., Snyder, L.V., Takáč, M., 2022. A deep q-network for the beer game: Deep reinforcement learning for inventory optimization. Manuf. Serv. Oper. Manage. 24 (1), 285–304.
Paterson, C., Kiesmüller, G., Teunter, R., Glazebrook, K., 2011. Inventory models with lateral transshipments: A review. European J. Oper. Res. 210 (2), 125–136.
Roerink, A.-M., 2021. Fresh food jumped 200% in online orders, and shows no signs of dropping. https://www.producebluebook.com/2021/06/10/fresh-food-jumped-200-in-online-orders-and-shows-no-signs-of-dropping/. (Online; accessed 14-July-2023).
Schaul, T., Quan, J., Antonoglou, I., Silver, D., 2015. Prioritized experience replay. arXiv preprint arXiv:1511.05952.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
Sultana, N.N., Meisheri, H., Baniwal, V., Nath, S., Ravindran, B., Khadilkar, H., 2020. Reinforcement learning for multi-product multi-node inventory management in supply chains. arXiv preprint arXiv:2006.04037.
Sun, W., Zou, Y., Zhang, X., Guo, N., Zhang, B., Du, G., 2022. High robustness energy management strategy of hybrid electric vehicle based on improved soft actor-critic deep reinforcement learning. Energy 124806.
Tagaras, G., Vlachos, D., 2002. Effectiveness of stock transshipment under various demand distributions and nonnegligible transshipment times. Prod. Oper. Manage. 11 (2), 183–198.
Van Hasselt, H., Guez, A., Silver, D., 2016. Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30.
Vanvuchelen, N., Gijsbrechts, J., Boute, R., 2020. Use of proximal policy optimization for the joint replenishment problem. Comput. Ind. 119, 103239.
Wang, Z., Dai, Y., Fang, S.-C., Jiang, Z.-Z., Xu, Y., 2020. Inventory transshipment game with limited supply: Trap or treat. Naval Res. Logist. 67 (6), 383–403.
Wang, K.-M., Ma, Z.-J., 2015. Age-based policy for blood transshipment during blood shortage. Transp. Res. Part E: Logist. Transp. Rev. 80, 166–183.
Wei, Y., Yang, M., Chen, J., Liang, L., Ding, T., 2022. Dynamic lateral transshipment policy of perishable foods with replenishment and recycling. Comput. Ind. Eng. 172, 108574.
Wu, Y., Zhou, J., Xia, Y., Zhang, X., Cao, Z., Zhang, J., 2023. Neural airport ground handling. IEEE Trans. Intell. Transp. Syst. 24 (12), 15652–15666.
Zhang, C., Ayer, T., White, C.C., Bodeker, J.N., Roback, J.D., 2023. Inventory sharing for perishable products: Application to platelet inventory management in hospital blood banks. Oper. Res. 71 (5), 1756–1776.
Zhou, J., Wu, Y., Cao, Z., Song, W., Zhang, J., Chen, Z., 2023. Learning large neighborhood search for vehicle routing in airport ground handling. IEEE Trans. Knowl. Data Eng. 35 (9), 9769–9782.
Zhou, Q., Yang, Y., Fu, S., 2022. Deep reinforcement learning approach for solving joint pricing and inventory problem with reference price effects. Expert Syst. Appl. 195, 116564.
Zhu, Z., Lin, K., Zhou, J., 2020. Transfer learning in deep reinforcement learning: A survey. arXiv preprint arXiv:2009.07888.