INVENTORY PLANNING OF PERISHABLE ITEMS USING REINFORCEMENT LEARNING

by

Ahmet Sualp Say

B.S., Industrial Engineering, Istanbul Technical University, 2020

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude and appreciation to all those who have contributed to the successful completion of my Master's thesis. This journey would not have been possible without the unwavering support and guidance of numerous individuals who have played a significant role in shaping my academic and personal growth. First and foremost, I am immensely grateful to my advisor, Prof. Taner Bilgiç, whose expertise, wisdom, and encouragement have been invaluable throughout this process. His patience, constructive feedback, and tireless dedication to my work have been instrumental in refining my research ideas and pushing me to reach my full potential. I am fortunate to have had the opportunity to learn from such an outstanding mentor. I would also like to extend my sincere thanks to the Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing me with the financial support that has enabled me to focus on my studies and research without undue burden. The scholarship I received from TÜBİTAK has been crucial in affording me the opportunity to pursue my Master's degree and advance my academic career. I am deeply grateful to my company for their continuous support and encouragement to pursue a Master's degree and conduct research. The flexibility and resources provided by the company have been invaluable in allowing me to balance my professional and academic commitments. I would like to thank the management and my colleagues for creating an inspiring and nurturing environment that fosters learning and personal development. My heartfelt appreciation goes to my family and friends for their constant love, support, and encouragement throughout my academic journey. Their unwavering belief in my abilities and their understanding of the sacrifices that come with pursuing higher education have been a source of strength and motivation. This Master's thesis is not only the culmination of my academic efforts, but also a testament to the support, guidance, and encouragement I have received from each and every one of the individuals mentioned above. Thank you for being a part of my journey.

ABSTRACT

INVENTORY PLANNING OF PERISHABLE ITEMS USING REINFORCEMENT LEARNING

Managing perishable inventory effectively is vital in diverse sectors like grocery, pharmaceuticals, composite materials, agriculture, blood. The challenge lies in reducing costs while handling items with limited shelf lives and changing demand. This study delves into the potential of computational techniques, notably the reinforcement learning methods Q-learning and SARSA, to tackle this intricate issue. We turned to reinforcement learning because traditional approaches struggle with increased complexity as the problem grows. We began our exploration with backward dynamic programming for a basic perishable inventory model. This model covered 10 periods and 3 age classes under a deterministic demand. We then expanded this framework to address more unpredictable demand patterns, both stationary and non-stationary, crafting optimal value functions for each. Our study also ventured into the Q-value approach, where transition probabilities were predefined, comparing the results to traditional dynamic programming. We further evaluated Q-learning and SARSA to see how close they converge to optimal. As the problem's complexity rose, especially with advanced demand scenarios like Advance Demand Information and more age classes beyond three, traditional methods fell short. In contrast, reinforcement learning proved nimble, especially in tackling more intricate inventory challenges. Our findings underline that reinforcement learning methods can approximate the near-optimal results achieved by dynamic programming in simpler scenarios. More remarkably, as the problem's intricacy grew, reinforcement learning continued to offer solutions, suggesting its promise in addressing even more complex inventory challenges.

ÖZET

PEKİŞTİRMELİ ÖĞRENME İLE BOZULABİLİR ÜRÜNLERİN ENVANTER PLANLAMASI

Çeşitli sektörlerde, özellikle market, ilaç, tarım ve kimya gibi endüstrilerde, çabuk bozulan ürünlerin envanterinin etkili bir şekilde yönetilmesi hayati öneme sahiptir. Bu ürünlerin sınırlı raf ömrü ve değişken talebi, maliyetlerin düşürülmesini zorlaştırmaktadır. Bu çalışmada, bu karmaşık sorunu ele almak için özellikle Q-öğrenme ve SARSA gibi pekiştirmeli öğrenme yöntemlerinin potansiyeli incelenmiştir. Araştırma, geriye dönük dinamik programlama ile temel bir çabuk bozulan envanter modeli üzerinde başlatılmıştır. Bu model, belirli bir talep altında 10 dönem ve 3 yaş sınıfını kapsamaktadır. Daha sonra, bu çerçeve hem durağan hem de durağan olmayan ve öngörülemeyen talep modellerini ele alacak şekilde genişletilmiş ve her biri için en uygun değer fonksiyonları oluşturulmuştur. Problemin durum uzayı büyüdükçe geleneksel yaklaşımların artan karmaşıklıkla başa çıkma konusunda zorlandığı gözlemlenmiş ve bu nedenle pekiştirmeli öğrenmeye başvurulmuştur. Çalışmada, geçiş olasılıkları önceden tanımlandığında Q-değer yaklaşımına da giriş yapılmış ve sonuçları geleneksel dinamik programlama ile karşılaştırılmıştır. Ayrıca Q-öğrenme ve SARSA'nın ne kadar optimal değerlere yaklaştığı değerlendirilmiştir. Sorunun karmaşıklığı arttıkça, özellikle İleri Talep Bilgisi gibi gelişmiş talep senaryoları ve üçten fazla yaş sınıfı ile, geleneksel yöntemlerin yetersiz kaldığı görülmüştür. Buna karşın, pekiştirmeli öğrenme, özellikle karmaşık envanter problemlerine sonuç üreterek etkili olmuştur. Bulgular, pekiştirmeli öğrenme yöntemlerinin, basit senaryolarda dinamik programlama ile elde edilen yaklaşık en iyi sonuçları tahmin edebileceğini göstermektedir. Daha da önemlisi, sorunun karmaşıklığı arttıkça, pekiştirmeli öğrenmenin çözüm sunmaya devam ettiğini gözlemlenmiştir. Bu, önerilen yöntemlerin daha karmaşık envanter problemlerini çözmedeki potansiyelini göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$C(S_t, x_t)$	Immediate cost of state S taking action x at time t
D_t	Demand occurs between time t and $t+1$
$Q(S_t, x_t)$	Q-value of S taking action x, i.e. approximated value
R_t	Reward at time t
S_s	State space
S_t	State at time t
V	Value function
W_t	Information at time t
X^π	Decision under policy π
X_s	Action space
x_t	Action at time t
y_t	Inventory level at time t
α	Learning rate
γ	Discount rate
ϵ	Exploration parameter
π	Policy function

LIST OF ACRONYMS/ABBREVIATIONS

ADI	Advanced Demand Information
AI	Artificial Intelligence
CFA	Cost Function Approximation
DLA	Direct Lookahead Approximation
DP	Dynamic Programming
DRL	Deep Reinforcement Learning
DQN	Deep Q-Networks
FIFO	First-In, First-Out
LIFO	Last-In, First-Out
MDP	Markov Decision Process
PFA	Policy Function Approximation
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
TD	Temporal Difference
TPM	Transition Probability Matrix
VFA	Value Function Approximation

1. INTRODUCTION

In the contemporary landscape of rapidly evolving supply chains, inventory planning has emerged as an indispensable aspect of ensuring business success and sustainability. The management of perishable items presents a unique set of challenges due to the time-sensitive nature of these products, necessitating a strategic approach to balance customer demands, waste minimization, and the assurance of quality and freshness.

The significance of this topic is further underscored by the staggering statistics associated with food waste and loss, with approximately one-third of all food produced for human consumption being wasted or lost globally [1]. This phenomenon has far-reaching consequences, including 4.4 gigatons of greenhouse gas emissions per year [2] and an economic impact of £12 billion annually in the United Kingdom alone [3]. Moreover, perishable items, such as fresh produce, dairy, and meat, comprise a substantial share of retail sales, with data from the Food Market Institute indicating that in 2019, perishables constituted 51.47% of total supermarket sales in the U.S., amounting to approximately $420 billion [4].

In addition to the food waste in grocery, the pharmaceutical sector is also critically influenced by the challenges of product perishability. The stability and efficacy of drugs are heavily dependent on time, emphasizing the importance of stringent expiration dates in the industry [5, 6]. These dates ensure that, given appropriate storage conditions, the drug's potency and safety remain uncompromised. After this period, there's an obligation to either dispose of the drugs or send them back to their producers, and this disposal can occur at various points in the supply chain. Illustratively, pharmaceutical items, which included medications and vaccines valued over a million U.S. dollars, were discovered to be expired in the Health Department of Chicago's warehouse in 2007 [7].

Given the significant economic and environmental implications of managing perishable items, this thesis aims to study the intricacies of inventory planning for perishable goods, shedding light on the challenges and opportunities in this vital area of supply chain management. The focus on perishable items in this thesis is driven by their unique characteristics, which make inventory planning significantly more complex than for non-perishable goods. Factors such as seasonality, short shelf life, variable demand patterns, and product-specific handling requirements necessitate a specialized approach to inventory planning that takes these complexities into account [8]. Moreover, the global trend toward healthier eating and a growing preference for fresh, high-quality products have placed added emphasis on the need for effective inventory management practices in the perishable goods sector [9].

In recent years, there has been a growing interest in the use of reinforcement learning (RL) techniques to optimize various business processes [10]. In the field of inventory management, these techniques hold particular promise for minimizing the total cost of managing perishable items. Therefore, the research question that this study seeks to answer is: *how can reinforcement learning principles be applied to minimize total inventory cost for perishable items, including lost sales and deterioration cost, in the context of inventory management in different domains?*. By answering this research question, we aim to contribute to the understanding of how reinforcement learning can be used to improve inventory management practices for perishable items. We initially apply dynamic programming to a single-product perishable inventory problem and extend this framework to accommodate various demand scenarios. Subsequently, we implement RL techniques such as Q-learning and SARSA, comparing their performance against classical approaches like dynamic programming in solving complex perishable inventory planning problems. The study's principal contribution is the demonstration of the potential of RL techniques, specifically Q-learning and SARSA, in addressing inventory problems with more than three age classes and non-stationary demand situations. These problems posed a significant computational burden using the dynamic programming approach, highlighting the practical advantages of RL in tackling large-scale, complex inventory management challenges.

In conclusion, this study endeavors to provide an understanding of inventory planning for perishable items through the lens of RL. To achieve this goal, the organization of this thesis is structured as follows: Chapter 2 presents the concepts and related work, encompassing classical approaches and RL methods in inventory planning for perishable items. Chapter 3 delves into the methodology, detailing problem formulation and implementation specifics. In Chapter 4, the results of the computational experiments are presented, followed by a comprehensive discussion and comparison of the findings, which reveal the potential and limitations of the studied approaches. Finally, Chapter 5 concludes the thesis, summarizing the main insights, and offering suggestions.

2. LITERATURE REVIEW

This chapter aims to provide a comprehensive understanding of reinforcement learning (RL) and its potential applications in the domain of perishable inventory management. The structure of the review has been carefully designed to facilitate a coherent and systematic exploration of the subject matter, enabling readers to comprehend the essential concepts and their interrelations effectively.

In Section 2.1, we begin by establishing a solid foundation in the theoretical underpinnings of RL, deviating from the conventional literature review approach. By sequentially discussing core components, the Markov Decision Process, Bellman Equation and Dynamic Programming, and Temporal Difference Learning, the review ensures that readers acquire the requisite understanding of RL techniques before exploring their practical implications. This unique approach, which emphasizes the most recent enhancements first, allows us to better understand how the background of Reinforcement Learning is deeply rooted in older, foundational concepts. In doing so, we aim to demonstrate the strong connection between these core components and the traditional theories that have shaped the development of RL over time.

In Section 2.2, the review begins by examining classical approaches and techniques in inventory management. This discussion offers an insight into the traditional methods and their limitations, thereby establishing the rationale for investigating RL applications in inventory management. By presenting these applications, the review showcases the potential advantages of RL in addressing the distinct challenges associated with perishable items, underscoring the relevance and importance of this research area.

2.1. Reinforcement Learning

RL, a subdomain of machine learning, explores the process by which an agent learns to make informed decisions based on trial-and-error experiences within a dynamic environment. [11]. It presents a versatile and potent mathematical framework for intelligent decision-making, with widespread applicability across diverse fields such as robotics, finance, gaming, and healthcare. [12].

RL, also referred as approximate dynamic or approximate adaptive programming, provides an effective substitute to conventional dynamic programming (DP) techniques by estimating transition probability matrices (TPMs) instead of calculating them precisely. This enables decision-making in extensive and intricate Markov Decision Processes (MDPs), even in situations where the exact transition probabilities are either unknown or computationally prohibitive to determine [13]. In this approach, the learner is not told what to do, but instead learns from the effects of their actions. The literature is quickly growing and has many different methods, making it an important area of study in artificial intelligence (AI) [14].

2.1.1. Core Components

The fundamental compenents of RL can be encapsulated as shown in Figure 2.1, which illustrates the interaction between five primary components: the agent, environment, action, state, and reward. In every iteration t, the agent observes the current state S_t and selects an action x_t from the available action set based on an exploration strategy [15]. A unique challenge in reinforcement learning, not present in other learning approaches, is managing the trade-off between exploration and exploitation. To maximize rewards, an agent must favor actions that have been previously tried and found to yield positive outcomes. However, discovering these rewarding actions requires the agent to explore new, untested actions. Thus, the agent needs to balance exploiting known successful actions to achieve rewards while also exploring new actions to improve future decision-making [10]. Subsequently, the environment reacts to the

chosen action by producing a reward R_t, resulting in a transition to the subsequent state. This ongoing process allows the agent to learn and adapt its decision-making strategy based on the feedback received from the environment, ultimately aiming to maximize or minimize the cumulative reward while navigating through various states and actions.

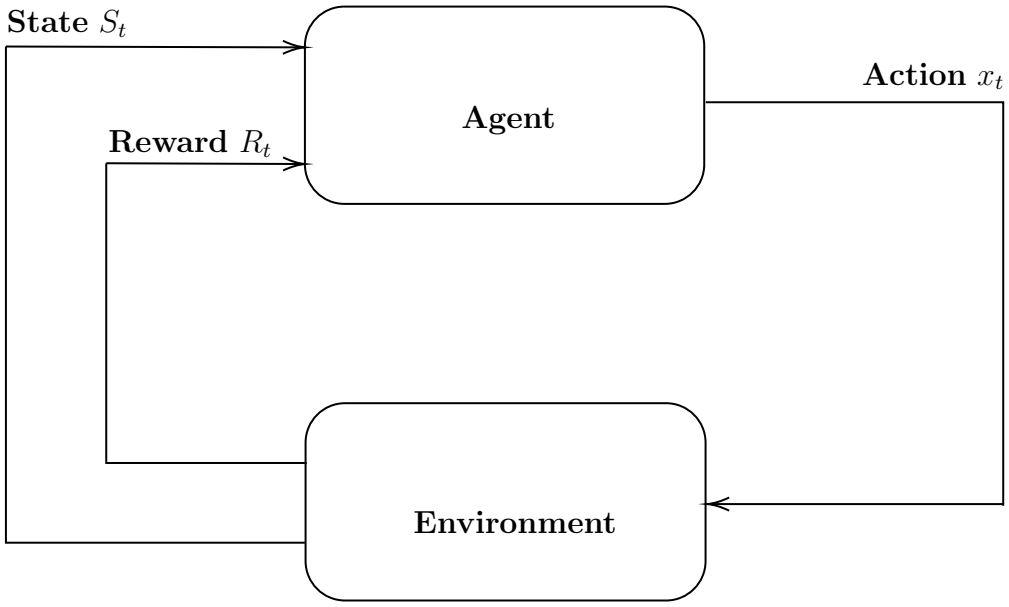

Figure 2.1. Relation diagram of reinforcement learning.

2.1.2. The Markov Decision Process

The MDP serves as a formal framework for modeling decision-making problems in uncertain environments [16]. In an MDP, a system transitions between states using Markov chains, with the probability of transitioning to the subsequent state dependent on the current state. Within specific states, the system selects an action based on a policy, which maps states to actions, and receives an immediate reward upon transitioning. The primary objective of an MDP is to identify the optimal policy that maximizes a performance metric, derived from accumulated rewards over a designated time horizon [13]. Consequently, MDPs constitute the foundation for numerous RL algorithms, as they facilitate modeling sequential decision-making problems [10].

2.1.3. Bellman Equation and Dynamic Programming

The origins of RL can be traced back to the early work on optimal control and dynamic programming by Richard Bellman and others in the 1950s. In the late 1950s, the phrase *optimal control* emerged to define the challenge of creating a controller that reduces a dynamical system's performance metric over a period. This method employs the notions of a system's state and a value function, also known as the *optimal return function*, to establish a functional equation frequently referred to as the Bellman equation [10]. Bellman was responsible for presenting the discrete stochastic variant of the optimal control issue, referred to as MDPs [17]. Additionally, Ronald Howard developed the policy iteration technique specifically for MDPs [18].

Bellman's primary achievement was demonstrating that the computational complexity of an MDP could be significantly decreased using an approach now widely recognized as DP [13]. Dynamic programming has evolved significantly since its inception in the late 1950s, with advancements in areas such as partially observable MDPs [19], various applications [20], asynchronous methods [21], and approximation techniques [22]. The study of these approximations goes by various names, such as approximate dynamic programming, reinforcement learning, and adaptive dynamic programming.

One of the key insights that emerged from this line of research was the potential of learning from experience when the model or transition functions are not known, enabling RL algorithms to refine their policies through trial and error. This approach allowed for the development of more scalable and adaptive solutions, capable of tackling larger and more complex problems [10]. However, it was also recognized early on in the evolution of this field that on large-scale and complex MDPs, these methods break down. The reason for this is the requirement of computing, storing, and manipulating the so-called transition probability matrix (TPM). This matrix is responsible for the breakdown of classical dynamic programming and lead to the curse of modeling and the curse of dimensionality [13].

2.1.4. Temporal Difference Learning

RL possesses the aptitude to adeptly tackle complex and large-scale MDPs, exhibiting superior performance in comparison to traditional DP approaches, facilitated by the convergence of concepts originating from Temporal Difference (TD) learning. In 1988, Sutton developed the TD learning method, integrating the strengths of both the Monte Carlo approach and the DP. Like the Monte Carlo technique, it does not require a model of the environment, and akin to DP, it updates estimates without waiting for the completion of an episode. As a result, TD learning refines the estimation of state-action pair values using immediate rewards and expected rewards from future states [23].

In the literature, two main approaches have been proposed for TD learning. One of these is Q-learning, which is an off-policy algorithm. This means that the learned action-value function, Q, aims to iteratively approximate the optimal action-value function, regardless of the policy being followed [24]. Another approach to TD learning is SARSA, an on-policy algorithm, proposed by Rummery and Niranjan [25] which stands for State-Action-Reward-State-Action. Unlike Q-learning, SARSA operates under the policy when selecting the next action. Both of these TD learning approaches have been extensively studied and applied to various problem domains, demonstrating their flexibility and adaptability in addressing a wide range of sequential decision-making tasks. Furthermore, these methods have laid the groundwork for the development of more advanced RL algorithms, such as Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO), which leverage the power of deep learning to tackle high-dimensional state spaces and complex decision-making environments [26, 27].

Throughout this thesis, the central emphasis is placed on the two previously described algorithms, Q-learning and SARSA, as they form the principal subject matter. In the Chapter 3, a comprehensive examination of the implementation specifics for both techniques are presented, accentuating their practical facets and underscoring the intricacies of their deployment.

2.2. Perishable Inventory Management

The literature on this topic has been vast and has covered a wide range of topics and approaches, including deterministic and stochastic models, optimization techniques, simulation-based approaches, and more recently, reinforcement learning algorithms. Given the broad range of topics and techniques in this field, it can be challenging to identify the most relevant studies and the most effective strategies for managing perishable inventory. However, Goyal and Giri have undertaken an extensive examination of the existing literature, offering a thorough review of the research conducted in the field of perishable inventory management up to 2001 [28]. Additionally, Nahmias has contributed significantly to this area, providing comprehensive reviews that further enrich the understanding of perishable inventory management and its underlying principles [8, 29]. Their work is particularly valuable as it synthesizes a large body of research, highlights key findings, and identifies research gaps. Moreover, their comprehensive analysis serves as a valuable reference point for researchers and practitioners seeking to explore innovative methods and techniques for addressing the unique challenges associated with perishable inventory management.

2.2.1. Classical Approaches and Techniques

Classical approaches on perishable inventory management encompasses a wide array of methodologies, which can be broadly classified into two categories: optimal and heuristic approaches. Optimal solutions strive to achieve the most favorable outcome for a given problem, whereas heuristic approaches generally prioritize computational efficiency and involve the comparison of various inventory policies against optimal benchmarks. Besides, in order to facilitate a comprehensive analysis and comparison of classical approaches, this literature review is concentrated on several key concepts that are prevalent within the field. These concepts, which include inventory policy, product lifetime, demand process, lead time, handling unsatisfied demand, and customer withdrawal policy are elaborated upon in an academic context to facilitate a holistic understanding of perishable inventory management and its challenges.

- Inventory policies dictate the timing and quantity of inventory replenishment. Two primary types of inventory policies are periodic review and continuous review [30–32].
- Product lifetime refers to the time span during which an item maintains its usability, quality, or value before being deemed expired or unsellable. In the context of modeling product lifetime, two distinct approaches have emerged within the literature: deterministic and stochastic [33].
- The demand process encompasses the patterns and fluctuations of customer demand for products over time.
- Lead time refers to the time interval between placing an order for replenishing inventory and the actual receipt of goods. In perishable inventory management, lead time plays a critical role, as it influences the risk of stockouts, excess inventory, and product spoilage.
- Unsatisfied demand arises when customer demand exceeds the available inventory, resulting in stockouts and lost sales opportunities. The literature presents various approaches for addressing unsatisfied demand, such as backordering, substitution, demand rationing and lost sales.
- Customer withdrawal policy refers to the rules and procedures followed when customers retrieve items from the inventory. Two common policies are First-In, First-Out (FIFO) and Last-In, First-Out (LIFO) [34]. FIFO ensures that items with the earliest expiration dates are sold first, while LIFO focuses on selling the most recently received items first.

Although other strategies, such as pricing policies, have been presented in the literature, they are not the primary focus of this review due to their divergence from our chosen methodology [35–37]. This review particularly accentuates research concerning periodic review, FIFO policy, and deterministic product lifetime, as these components are more prevalently implemented in practice and exhibit a higher degree of relevance to our environmental variables. Furthermore, this targeted approach allows for a more in-depth exploration of the nuances and intricacies associated with these specific aspects.

Veinot has been recognized as a pioneer in the domain of perishable inventory management, as highlighted in Pierskalla's review [38]. Veinot's seminal work centered on exploring the optimal order policy and affirming the indispensability of the FIFO principle in the context of deterministic demand and fixed product lifetime assumptions. Building on this groundwork, Van Zyl formulated a periodic review policy inventory model, which aimed to minimize expected costs while taking into account stochastic demand, two-period product lifetimes, zero lead times, and the absence of backordering [39]. In a subsequent development, Nahmias and Pierskalla expanded upon Van Zyl's foundational research by generalizing their investigation to encompass two-period product lifetimes and the formulation of optimal ordering strategies under conditions of stochastic demand [40]. Furthermore, Nahmias and Fries, in their respective studies, investigated the optimal policy for the general lifetime problem with independent and identically distributed demands in distinct settings; Nahmias focused on a backlogging model, while Fries delved into a lost-sales model [41, 42]. Their research findings revealed that the optimal ordering quantity is contingent upon not only the age distribution of the existing inventory but also the time remaining until the end of the planning horizon.

The computational complexity associated with tracking the states of each product in perishable inventory management has posed a significant challenge, particularly as the dimensions of the problem increase. Consequently, while some research has focused on determining optimal solutions, a considerable amount of interest has also been directed towards developing heuristic methods for approximate solutions. Notably, Nahmias proposed solutions for scenarios involving product lifetimes greater than 2 [43]. Nandakumar and Morton extended this line of inquiry to lost sales situations, introducing upper and lower bounds for optimal orders [44]. Moreover, Chao et al. contributed an approximation algorithm that provides worst-case performance guarantees which can deviate up to 100% from the optimal solution, which has served as the foundation for our mathematical modeling part [45]. Similarly, Tekin et al. proposed a strategy and compared it with a classical approach, namely the (Q,r) type policy, demonstrating the value of heuristics in the context of perishable inventory [46].

Building upon the wealth of knowledge present in the extant literature, this thesis concentrates on employing reinforcement learning techniques as a heuristic approach to address perishable inventory control problems that prove to be intractable or unattainable through optimal solutions.

2.2.2. Applications of Reinforcement Learning

In recent years, RL has been successfully applied to a wide range of disciplines, yet its implementation in perishable inventory management remains relatively limited. The unique characteristics of perishable goods, such as their time-sensitive nature and susceptibility to spoilage, present distinct challenges. As a result, this literature review aims to not only focus on perishable inventory management but also encompass the broader field of inventory management. The goal is to understand the potential applications of RL in addressing various challenges and uncertainties that arise in these contexts.

To facilitate a comprehensive analysis of the relevant literature, we have categorized the studies based on specific criteria, such as product type, uncertainties addressed, algorithms employed, and the handling of unsatisfied demand. This structured approach enables us to identify patterns and trends in the use of RL techniques for inventory management, thereby providing a solid foundation for our investigation of its applicability to perishable inventory management problems.

Early applications of RL to inventory management emerged around the early 2000s, as seen in the work of Van Roy et al. [47]. They utilized neurodynamic programming algorithms and TD learning algorithms to compare costs with order-up-to policies for a single product, employing both backorder and lost sale strategies. Similarly, Giannoccaro and Pontrandolfo modeled the inventory problem as MDP and applied RL to determine a near-optimal inventory policy under an average reward criterion for a single product with backordering [15]. Rao et al. used Q-learning to achieve near-optimal solutions for inventory management in warehouse - store relations [48].

Notably, researchers such as Tongeren et al. applied Q-learning to the well-known supply chain problem, the beer game [49], similar to the work of Chaharsooghi et al. [50]. In subsequent years, Sui et al. investigated the application of RL in vendor-managed inventory systems with lost sales policies [51]. The most recent research in this area by Nurkasanah contrasts the Q-learning approach with traditional mathematical methods, especially when faced with multiple challenges. These include having a limited manufacturer and warehouse capacity, restrictions on supplier orders, unpredictable demand, uncertain lead times, and sourcing from multiple suppliers. After conducting experiments the results demonstrated that RL could more efficiently determine optimum order quantities compared to traditional mathematical methods, even under these complex constraints. However, it's noteworthy that the RL approach resulted in higher shortages [52].

In contrast to these general RL applications in inventory management, two studies hold direct relevance to the current research. The first is by Kara and Dogan, who modeled perishable items using an age-based policy [53]. Their experimental results indicate that an ordering policy incorporating age information is viable and that utilizing RL improves outcomes when demand has high variance and products have short lifetimes. However, their study did not compare the results with the optimal solutions that could be found using dynamic programming, focusing instead on comparing the outcomes of SARSA and Q-learning with another heuristic method, while only employing gamma-distributed demand and not non-stationary, correlated demand scenarios. Similarly, Selukar et al. employed Deep Reinforcement Learning (DRL) methods in their research, developing models that consider various limiting factors for retailers' inventory, such as overdue and shortage costs, lead time, and corruption costs for multiple products [54]. Through simulation experiments, they demonstrated that the proposed RL models effectively reduce inventory costs and spoilage rates of goods, particularly when product lead times, life cycles, and demand distribution are known. Nevertheless, our research diverges from the second study, as we do not employ DRL methods in our approach.

In summary, the existing literature demonstrates the potential of RL in addressing inventory management problems, particularly in the context of perishable items. The current research is build upon these previous studies to further explore the application of RL in perishable inventory management and optimize outcomes under various constraints. Consistent with studies by Nahmias, Pierskalla, and Van Zyl, our investigation aims to illustrate the conceptual application of RL without oversimplifying real-world situations, by examining inventory control of a single perishable item characterized by a known, deterministic lifetime, periodic random demand, FIFO policy, and zero lead time [39, 40]. In line with Fries, our approach considers unmet demand as lost [42]. Diverging from studies proposing optimal solutions for lifetimes less than 3 periods, our research focuses on lifetimes greater than 3 periods and employs RL to approximate optimal solutions while assuming the presence of correlated or non-stationary demand. This method contributes an additional layer of complexity to our investigation and expands the purview of the existing literature in the domain.

3. METHODOLOGY

The perishable inventory problem under investigation can be conceptualized as a sequential decision-making problem. This necessitates the delineation of states, transition, objective functions, and other related components as preliminary steps in formulating the problem. In this thesis, we examine the problem of sequential decision-making through the application of reinforcement learning (RL). Specifically, the problem is modeled as an Markov Decision Process (MDP), which serves as a foundational framework for representing decision-making problems in which an action must be selected in each decision state encountered by the system.

In recent years, RL has emerged as a powerful tool for addressing complex sequential decision-making problems. As a form of dynamic (or approximate) adaptive programming (DP), RL offers an efficient alternative to traditional DP methods by approximating transition probability matrix (TPM) rather than computing them precisely. This approach enables decision-making in large and intricate MDPs, even when exact transition probabilities are either unknown or computationally infeasible to calculate.

Our primary objective is to initially solve the fundamental version of this problem using DP. We begin by optimally solving the problem under deterministic demand conditions and subsequently proceed to tackle it under stationary stochastic conditions. Following this, we employ the Q-value approach to establish a connection with RL methods. Upon establishing a solid foundation, we apply RL techniques to the solution, compare the outcomes, and refine our findings accordingly. The RL method is then be implemented to address more intricate problems, in which DP fails, thereby broadening our comprehension of its potential applications in the realm of perishable inventory management and demonstrating the efficacy of RL in solving complex sequential decision-making problems.

3.1. Environment and Model Assumptions

In Section 2.2, we discussed key concepts to classify existing studies. To better understand our contribution to this field, we have defined the problem environment using those concepts:

- Our model operates under a periodic review inventory policy. This involves reviewing the inventory levels at regular, predetermined intervals and placing orders accordingly to replenish the stock.
- The order of events is in a period as follows: the period starts with the observation of items in the stock and their ages; replenishment decision is given; demand is observed; excess or lost sales are calculated; items are discarded when the period ends.
- Our model simulates an environment where a single store sells a solitary product, which is supplied by a single warehouse that has no capacity restriction.
- The product lifetime is predetermined and fixed and lead time is negligible.
- The model is capable of accommodating various demand settings. This aspect is discussed in detail in Section 4.1.
- The model and its associated algorithm assume that customers adhere to FIFO approach. This signifies that products are sold in the order they are received, with older products being sold first.
- The model operates under the assumption that any unmet demand results in a lost sale. This implies that if a customer does not find the product in stock, they will not wait for a restock.

Retail supply chains are complex, involving dynamic processes and requiring coordination between warehouses and stores [55]. Given the intricacies of these operations, it's essential to adopt a simplifying approach. However, it's crucial that these simplifications don't reduce real-world applicability. To ensure relevant insights, we've incorporated simplifications commonly used in the literature, which are outlined below.

- Inventory planning occurs over T periods, indexed by $t = 1, \ldots, T$.
- The product has a fixed lifetime of m periods, and aging commences as soon as goods depart the source.
- Demand is denoted as $D_1, \ldots, D_T$ across the planning horizon.
- In each period t, costs for each unit consist of: ordering cost c^o, holding cost c^h, unmet demand incurs a shortage cost c^s due to lost opportunities and customer goodwill, and unused products with no remaining lives are discarded with a deterioration cost c^d.
- In the last period, T, all stock is discarded with cost of c^d.

3.2. Problem Formulation

In light of the current academic landscape, Powell provides a comprehensive theoretical approach to modeling sequential decision-making problems, a domain that presently lacks a standardized methodology. This stands in stark contrast to the well-established framework employed for static, deterministic optimization problems, which has been developed rigorously since the 1950s. The modeling procedure presented in this thesis adheres to the guidelines set forth by Powell in his book, which caters to a technical audience primarily focused on the development and implementation of computer-based models [56].

As Powell suggests, any sequential decision-making problems can be represented by the following sequence:

$$(S_0, x_0, W_1, S_1, x_1, W_2, \ldots, S_t, x_t, W_{t+1}, \ldots, S_T). \tag{3.1}$$

Here, S_t denotes the state variable that encapsulates our knowledge at time t, x_t symbolizes the decision variables that embody the elements under our control, and W_{t+1} represents the exogenous information that emerges after making the decision x_t.

The subsequent sections of this thesis are structured according to Powell's six-step modeling process, which allows for a systematic approach to sequential decision-making problems.

Furthermore, it is crucial to emphasize that the foundation of our mathematical model is deeply rooted in the study conducted by Chao et al., as previously discussed in Section 2.2 [45]. Incorporating Powell's guidelines and building upon the work of Chao et al., this paper aims to model the perishable inventory control system as a MDP, allowing for the application of solution techniques such as DP and RL.

3.2.1. State Variable

Considering the problem necessitates tracking both the inventory level and age distribution, an appropriate state vector for period t is constructed as follows:

$$S_t = (S_{t,1}, \ldots, S_{t,m-1}). \tag{3.2}$$

In this formulation, $S_{t,i}$ represents the quantity of on-hand products with a remaining lifetime of i periods in period t, where i ranges from 1 to $m-1$. To simplify the model, an arbitrary initial state is set to 0, which implies that $S_{1,i} = 0$ for all i within the interval 1 to $m-1$. Consequently, the complete set of S_t constructs the state space known as S.

3.2.2. Decision Variable

The primary objective is to ascertain the optimal order quantities. Consequently, we define x_t as the quantity ordered in period t, and our decisions are based on a policy, $X^\pi(S_t)$, which is expounded upon in subsequent sections. Notably, x_t can only assume integer values and take a finite number of values from predetermined set X_s. Furthermore, let y_t denote the total inventory level upon receiving the order x_t in period t. Consequently, y_t can be expressed as:

$$y_t = \sum_{i=1}^{m-1} S_{t,i} + x_t. \tag{3.3}$$

3.2.3. Exogenous Information

The realized demand for the product between periods t and $t+1$, denoted as D_t, can be regarded as exogenous information, such that $W_t = D_t$. Moreover, at the end of period t, if $y_t - D_t \geq 0$, the surplus inventory incurs a holding cost of $c^h(y_t - D_t)$. Conversely, if $y_t - D_t \leq 0$, the system experiences a lost sales cost of $c^s(D_t - y_t)$. Furthermore, should the inventory with one period remaining lifetime satisfy the condition $S_{t,1} \geq D_t$ the system will incur a deterioration cost of $c^d(S_{t,1} - D_t)$.

3.2.4. Transition Function

The system transits from S_t to S_{t+1} based on FIFO policy, employing the following transition equations:

$$S_{t+1,j} = \max\{0, S_{t,j+1} - \max\{0, D_t - \sum_{i=1}^{j} S_{t,i}\}\}, \quad 1 \leq j \leq m-2. \tag{3.4}$$

At the arrival of demand, it is initially satisfied by $S_{t,1}$, and once depleted, the system proceeds to $S_{t,2}$, continuing in this manner until it reaches the final state, $S_{t,\,m\text{-}1}$. Upon reaching the final state, the system employs the following equation:

$$S_{t+1,m-1} = \max\{0, x_t - \max\{0, D_t - \sum_{i=1}^{m-1} S_{t,i}\}\}. \tag{3.5}$$

In cases where the inventory is insufficient at the final state, unsatisfied demand is lost.

3.2.5. Objective Function

In this study, our primary objective is to minimize the overall inventory costs, encompassing order, lost sales, holding, and deterioration costs, while determining the appropriate order quantities. The cost of single-period, satisfying the demand D_t, can be expressed as follows:

$$C(S_t, x_t, D_t) = c^o x_t + c^h \max\{0, y_t - D_t\} + c^s \max\{0, D_t - y_t\} + c^d \max\{0, S_{t,1} - D_t\}. \tag{3.6}$$

Here, $x_t = X^\pi(S_t)$ for a given policy π. Given a sequence of random demands $D_1, \ldots, D_T$, the cost of a policy π is:

$$\hat{F}^\pi = \sum_{t=0}^{T} C(S_t, X^\pi(S_t), D_t). \tag{3.7}$$

The cost $\hat{F}^\pi$ is random because it depends on a particular sequence of random demands $D_1, \ldots, D_T$. Consequently, taking the expectation over these random demands, yields the final objective function:

$$F^\pi = \mathbb{E}\left[\sum_{t=0}^{T} C(S_t, X^\pi(S_t), D_t)|S_0\right] \tag{3.8}$$

where it is crucial to emphasize that under deterministic conditions, taking the expectation is not required. In contrast, in a stochastic setting, computing the expectation becomes essential. This study aims to address both deterministic and stochastic scenarios; thus, we employ both approaches to ensure a comprehensive analysis.

3.3. Modeling the Uncertain Demand

In the existing literature, numerous approaches have been proposed for modeling uncertainty in exogenous information, including the development of mathematical models, the analysis of historical data, and the observation of real-life outcomes. A common strategy is to model demand based on past data; however, this approach may prove inadequate when the system encounters out-of-stock situations, as the complete demand might not be observed. Consequently, an alternative method for modeling unmet demand could be required. Another possibility is to employ a mathematical model that does not necessitate any supplementary steps. A variety of probability distributions, such as Poisson, Gamma, Erlang, and others, have been advocated in the literature for modeling stochastic demand. Poisson distribution has been employed in many early works in this domain by researchers such as Nahmias, Van Zyl, and Fries [39, 41, 42]. Moreover, Chao et al. proposed different types of demand settings, ranging from autoregressive to Markov modulated demands [45]. By adopting a distribution model, we can generate a set of demands $D = (D_1, D_2, \ldots, D_T)$.

In this research, we explore deterministic, stationary stochastic with independent and identically distributed Poisson distribution, non-stationary and possibly correlated stochastic scenarios to examine the effects of different demand patterns, such as seasonality and other fluctuations in Section 4.1.

3.4. Designing a Policy

In the literature, Powell provides a comprehensive classification of policy design into two overarching categories, each of which is further divided into two distinct classes. The first category is Policy Search, encapsulating policy function approximations (PFAs) and cost function approximations (CFAs). The second category, Lookahead policies, is comprised of value function approximations (VFAs) and direct lookahead approximations (DLAs) [56]. Each of these policy designs, denoted by π, incorporates information about the type of function, F, and any adjustable parameters as specified in Section 3.2.5.

Policy search denotes the systematic exploration and assessment of different decision-making methods or functions. This evaluation is generally conducted via simulations, with the aim of identifying the method demonstrating the highest average efficacy over a predetermined period. Models such as the periodic review reorder quantity model (t, Q), periodic review reorder point-up to level (t, S), continuous order-up to level (s, S), and continuous reorder quantity (s, Q) have garnered considerable attention due to the simplicity of execution. Also, several numeric controls, such as continuous-reorder quantity (s, Q), periodic order up to level (R, S), were employed to articulate inventory policies in the context of susceptibility [57]. In contrast, lookahead policies entail the formulation of proficient strategies by optimizing the aggregate impact or cost of a decision, while simultaneously accounting for an approximated evaluation of future impacts or costs that may emerge from the present decision. Our focus in this thesis is primarily on VFAs, a component of lookahead policies. As our problem at hand is one of minimization, we adapt all notations accordingly. However, these notations can be seamlessly exchanged with those oriented towards maximization.

VFAs, approximate downstream costs resulting from current decisions, which are then considered in conjunction with the initial cost of the current decision. This approach utilizes Bellman's logic and is represented in the following equation for deterministic environments:

$$V_t(S_t) = \min_x(C(S_t, x_t) + \gamma V_{t+1}(S_{t+1})), \quad t = 1, \ldots, T-1. \tag{3.9}$$

In this equation, $C(S_t, x_t)$ denotes the cost associated with a specific action x_t in state S_t. Simultaneously, $\gamma V_{t+1}(S_{t+1})$ signifies the discounted value of the subsequent state [58]. By considering all potential actions within a state and selecting the one with the lowest value, the value of each state can be computed. To adapt this equation for a stochastic environment, Powell proposes the following:

$$V_t(S_t) = \min_x(C(S_t, x) + \mathbb{E}\left[V_{t+1}(S_{t+1})\right]) \quad t = 1, \ldots, T-1. \tag{3.10}$$

In both deterministic and stochastic contexts, the terminal condition is given by:

$$V_T(S_T) = c^d * S_T. \tag{3.11}$$

This condition specifies that the value in last period T is determined by the deterioration cost c^d multiplied by the inventory level S_T. In practical applications, it is generally necessary to replace the value function $V_t(S_{t+1})$ with an approximation, denoted as $\overline{V}_{t+1}(S_{t+1})$ [56]. This is precisely where value function approximation takes place. As a result, the policy to be employed for the problem under investigation can be expressed as:

$$X^\pi(S_t) = \arg\min_x(C(S_t, x) + \mathbb{E}\left[\overline{V}_{t+1}(S_{t+1})|S_t, x\right]), \quad x \in X_s, \tag{3.12}$$

where it demonstrates that in our problem, we aim to select the action that minimizes today's cost while considering expected future costs at each state. The deterministic counterpart of this equation is readily obtained by substituting the expected value with direct costs.

At this stage, the state-of-the-art of this thesis come to the fore. In order to approximate the values of each state, we employ two solution methods: DP and RL. By leveraging the Bellman equation and given that demand probabilities are known, we can calculate optimal order quantities through backward DP using the known TPM.

However, as the product age increases, approximation via simulations becomes necessary. RL offers a wide array of algorithms to address this issue, but our focus is on TD learning algorithms, as previously mentioned in Section 2.1.4. The selected algorithms, Q-learning and SARSA, facilitate the approximation of the value of a state-action pair, driven by the immediate reward and the anticipated reward from the subsequent state. Watkins introduced the following equation to clarify this concept for Q-learning [24]:

$$\hat{Q}(S_t, x_t) \leftarrow Q(S_t, x_t) + \alpha[C(S_t, x_t) + \gamma \min_x Q(S_{t+1}, x) - Q(S_t, x_t)]. \tag{3.13}$$

Here, the equation showcases the iterative process of approximating the value function, so-called Q. Whenever an agent interacts with its environment, the agent's estimated Q values, denoted as $\hat{Q}$, are updated using Equation (3.13). Another representation of the update rule is given below [49]:

$$\hat{Q}(S_t, x_t) \leftarrow (1-\alpha)\underbrace{\hat{Q}(S_t, x_t)}_{\text{old value}} + \alpha \overbrace{\left(\underbrace{C(S_t, x_t)}_{\text{cost}} + \underbrace{\gamma \min \hat{Q}(S_{t+1}, x))}_{\text{approximate future value}}\right)}^{\text{learned value}} \quad x \in X_s, \tag{3.14}$$

where the parameters α and γ play a crucial role in determining the behavior of the value function. The parameter γ serves as the discount factor, controlling the weight assigned to future rewards in the value function. Conversely, α is the learning rate, dictating the trade-off between exploitation and exploration processes; a larger α value results in greater learning from rewards, while a smaller α value promotes the exploitation of current knowledge. The values of α and γ are determined using different methods, to be discussed in Section 4.

Estimated Q-values are stored in a Q-Table. Each time action x_t reaches at the end of S_t, the cost $C(S_t, x_t)$ is calculated and the Q-table is updated with a new $\hat{Q}$ value, as outlined in Equation (3.13). The Q-table format is illustrated in Figure 3.1, with columns correspond to the order quantities from the set X_s, while rows are indexed by the states from the set S. In essence, the Q-table serves as a dynamic repository of accumulated knowledge.However, it's worth noting that the displayed format represents only a single time horizon, t. In our implementation, the Q-table encompasses multiple t dimensions. Adjustments to the update rule and format, in light of this multi-dimensional approach, are elaborated upon in Section 3.5.

	0	1	...	X
1	Q(1, 0)	Q(1, 1)		Q(1, X)
2	Q(2, 0)	Q(2, 1)		Q(2, X)
.				.
.				.
.				.
S	Q(S, 0)	Q(S, 1)	...	Q(S, X)

Figure 3.1. Q-table format.

A crucial divergence between Q-learning and SARSA is discernible in the operational mechanisms of their learning processes. As delineated in Equation (3.13), Q-learning's functionality does not mandate the explicit inclusion of a policy during the update process, thereby characterizing it as an off-policy method. In contrast, SARSA does necessitate the implementation of a policy in its updating protocol, classifying it as an on-policy method. It is imperative to highlight that the conception of *policy* utilized in this context differs from our previous discussion. Here, the term *policy* is more akin to a procedural strategy rather than a conventional policy in the realm of RL. To explicate further, Q-learning, during the update of the Q value, invariably opts for the action associated with the minimal value for the subsequent state. However, the selection of the next action is based on an exploration strategy rather than on the immediate minimization of the action value. On the other hand, the distinguishing characteristic of SARSA in its update rule can be mathematically represented as:

$$Q(S_t, x_t) \leftarrow Q(S_t, x_t) + \alpha[C(S_t, x_t) + \gamma Q(S_{t+1}, x') - Q(S_t, x_t)]. \tag{3.15}$$

In the equation, x' represents the action chosen, not directly by the action yielding the minimum Q-value, but rather in accordance with a specific policy being implemented, such as the ϵ-greedy policy. This is in stark contrast to off-policy algorithms like Q-learning, where the selection of the next action does not necessarily depend on the policy being followed, providing a nuanced perspective on the strategic interplay between action selection and policy implementation in RL.

The distinct update rules for Q-learning and SARSA algorithms lead to substantial differences in their behavior. This difference is well-illustrated by the well-known *Cliff Walking* problem described by Sutton and Barto in their book [10]. Q-learning aims to learn the optimal policy, often following a path dangerously close to the cliff's edge. Due to its ϵ-greedy action selection method, it may occasionally fall off the cliff. On the other hand, SARSA considers the potential future action during learning, which leads it to choose a longer but safer route, away from the edge of the cliff. The upshot is that, Q-learning carries a risk of occasionally selecting an unsafe action, leading to a fall off the cliff. In contrast, SARSA tends to be a safer choice, with more stable learning and more convergence chance but possibly slower progress.

In summary, the algorithm uses a value function to represent the expected long-term reward for each state-action pair. The goal of the algorithm is to select the actions that minimize the overall cost. It does so by updating the value function for each state-action pair iteratively, based on the immediate reward and the estimated future rewards from the next state. This process continues until the algorithm reaches convergence, at which point the value function represents the optimal policy for selecting actions in each state. In Section 3.5, the application of this theoretical foundation to our specific case is expounded, and the impact of the algorithm's parameters on the results are analyzed.

3.5. Implementation Details

In this section, we discuss the implementation of the solution methods, specifically focusing on our application to the problem. As emphasized, we employed three methods: DP, Q-value approach, and RL techniques, namely Q-learning and SARSA. Each offers unique attributes in terms of convergence, computational efficiency, and applicability, presenting a comprehensive toolset to address the challenges presented by our problem. Furthermore, the comparative analysis of these methods not only enhances our understanding of their individual strengths and weaknesses but also provides a roadmap for their synergistic use in complex inventory management scenarios.

We implement backward DP to find the optimal results for the three-age problem under deterministic and stationary stochastic demands. The solution was straightforward in this context. It is important to note that we only consider the stochastic versions of the implementations, as the deterministic case represents a scenario in which the probability of demand occurrence is 1. Given that the demand distribution and its parameters are known, we were able to calculate the probability of each demand occurrence. This information allowed us to generate the TPM. Knowing the exact transition probabilities facilitated the calculation of the expected value for each state. This enabled us to optimally solve the problem using a backward approach, effectively dividing the whole problem into smaller subproblems. Additionally, the successful application of the backward DP method in this scenario has laid a strong foundation for the implementation of the Q-value approach and RL techniques. Within the same context, the Q-value approach serves to establish the relationship between DP and RL techniques. Gosavi proposes an algorithm and provides a pseudocode for performing value iteration using Q-values [13]. Utilizing the TPM calculated previously for the DP solution, we iteratively update the Q-values, ultimately obtaining the optimal solution. This approach effectively bridges the gap between DP and RL techniques, demonstrating their interconnected nature within the problem-solving process when the model is known.

Lastly, a simulation pseudocode is introduced to address the problem, which was previously formulated as a sequential decision-making problem using value function approximations. Assuming model uncertainty, the proposed algorithm leverages both the Q-learning and SARSA mechanism, as discussed in Section 3.4. Various implementations of Q-learning and SARSA can be found in the literature; the specific approach implemented in this study is based on Gosavi's work [13]. The fundamental concept of this method involves learning the values of Q-functions through an iterative process. Once the learning process is complete, the approximate ordering policy for each state is determined by selecting the action with the highest Q-value.

Inputs:

algorithm, strategy, decay, c^o, c^h, c^s, c^d, m, T, X_s, α, γ, N, ϵ, S

Initialize:

episode=0, t=1, Q(s, x)=0, $\forall s \in S$ and $\forall x \in X_s$

while episode $\leq$ N **do**

Take action x_t with provided strategy

while t $\leq$ T **do**

Observe cost $C(s_t, x_t)$ and new state s_{t+1}

Update Q-table using the formula:

if algorithm: Q-learning and $x \in X_s$ **then**

$Q_t(s_t, x_t) \leftarrow Q_t(s_t, x_t) + \alpha[C(s_t, x_t) + \gamma \min Q_{t+1}(s_{t+1}, x) - Q_t(s_t, x_t)]$

Select action x_{t+1} with provided strategy

else if algorithm: SARSA and $x \in X_s$ **then**

Select action x_{t+1} with provided strategy

$Q_t(s_t, x_t) \leftarrow Q_t(s_t, x_t) + \alpha[C(s_t, x_t) + \gamma Q_{t+1}(s_{t+1}, x_{t+1}) - Q_t(s_t, x_t)]$

$s_t \leftarrow s_{t+1}; x_t \leftarrow x_{t+1}; t \leftarrow t + 1$

Update ϵ, α if decay is True

t=1

episode=episode+1

Figure 3.2. Proposed simulation pseudocode.

The overall structure of the simulation is delineated in Figure 3.2. It initializes the Q-table with arbitrary values and then enters the learning loop. In each iteration, the algorithm selects a random state from the state space, S_s. Subsequently, it chooses an action from the set of actions, X_s, according to the algorithms' rule and executes the selected action. The algorithm then updates the Q-value of the state-action pair using the Bellman equation and the observed reward. This process is repeated for a predefined number of iterations or until the Q-values converge to a stable solution. Once the learning process is completed, the optimal action for each state is determined by selecting the action with the highest Q-value. As evident in the pseudocode, there's a

nuanced alteration to the conventional Q-table, deviating from classical representation. Given the finite-horizon nature of our problem, we incorporated a time-aware approach, adding a time dimension, t to the Q-values. This ensures that for each time t the value of each state-action pair is updated independently.

c^o, c^h, c^s, c^d are the costs associated with the inventory problem whereas m refers to lifetime of the product, T is planning horizon and X corresponds to maximum quantity that can be ordered. Moreover, the hyperparameters N, α, γ, and ϵ correspond to the number of episodes, learning rate, discount factor, and exploration parameter, respectively. In the context of the defined problem, an episode consists of a series of iterations across the planning horizon $1, \ldots, T$. The discount factor γ governs the emphasis on future rewards, with a value of 0 leading to a myopic agent focused solely on immediate rewards and a value of 1 promoting long-term reward optimization. The learning rate α lies between 0 and 1.

To strike a balance between exploration and exploitation, various methods have been proposed [59]. This study adopts the well-established ϵ-greedy approach for its exploration strategy and also implements pursuit and limiting greedy strategies. The exploration parameter, denoted by ϵ, has been considered in these methods. We utilize not only the standard ϵ-greedy approach but also its modified version, as proposed by Chaharsooghi et al. [50]. The exploration parameter begins at a predefined ratio in the first iteration and decreases to another predefined ratio in the last iteration. This approach ensures that the algorithm effectively explores the state-action space during the initial learning process and leverages the acquired knowledge as the iterations progress. Existing research demonstrates that when the learning rate decays appropriately, the estimates converge to the optimal values for all possible pairs [60]. Consequently, this adaptive methodology is applied not only to the exploration parameter but also to the learning rate, which decays at a predefined ratio to regulate the amount of learning in later episodes.

4. EXPERIMENTS AND RESULTS

We have divided the experiment and results section into six parts. Section 4.1 delves into the various demand settings. It is essential to understand and quantify the demand for perishable items as it has a direct impact on the inventory. We elaborate on how various factors are incorporated to simulate a realistic demand scenario. Next, we explain how Dynamic Programming is used to solve the perishable inventory management problem in Section 4.2. It provides a baseline for comparing the performance of more sophisticated methods that follow. Section 4.3 elaborates on the Q-value approach, implementation of Q-learning algorithm when the model is known, to manage the inventory of perishable items. The subsequent section deals with the implementation of the RL approach. This is the main focus of the research and involves training an RL agent to make optimal decisions regarding inventory management. The section describes the details of the RL setup and the algorithm's nuances in the context of the problem at hand. In Section 4.5, we compare the results obtained from each of the above methods. We evaluate them on various parameters, such as accuracy, efficiency, and scalability, to identify their strengths and weaknesses. Section 4.6 presents an in-depth analysis of the performance of the RL approach in large state spaces. This section assesses the scalability of the RL approach and its ability to handle more complex inventory management scenarios.

In addressing the problem at hand, we found it necessary to make certain assumptions as outlined in Section 3.1. In alignment with established literature, we primarily focused on situations where the product lifetime is predetermined and fixed. The work of Fries provided valuable insight for our study; Fries examined various product lifetimes ranging from 1 to 3+ and proposed optimal policies for products with a lifespan of 1 and 2, also commenting on the policies when product life extends beyond 3 [42]. Given that perishability considerations become significant when the product life is 3 or more, we accordingly concentrated our efforts on exploring this 3-aged problem.

Our work also involves planning inventory over specified periods of time. To ensure we capture both immediate and future effects, we carried out our research over two different time periods: 10 and 25 time units. The results we obtained align with the findings of Nahmias and Fries. They studied the optimal approach for problems involving product lifetime with independently and identically distributed demands. Their studies showed that the best order quantity is affected by both the age of current inventory and the time left until the end of the planning period [41, 42]. Our own findings echo this, showing that decisions change based on both inventory age and the remaining planning period, reinforcing the intricate nature of managing inventory of perishable products effectively.

Furthermore, certain cost parameters needed to be defined to accurately represent the problem at hand. As highlighted in Section 3.1, the total cost comprises four key components: ordering, holding, shortage, and deterioration costs. These parameters embody distinct aspects of inventory management and were set to represent realistic dynamics. We designated values of \$5, \$1, \$8, and \$10 per unit respectively for these parameters. This ensured that our model accurately mirrored the multifaceted nature of real-world inventory management. Similarly, to ensure the problem remains tractable for dynamic programming and to limit the state space, we set an upper boundary for the maximum order quantity. We specifically set this limit to 5 units, meaning the system cannot order more than this amount. The main rationale behind this decision was to reduce the state space, thereby alleviating computational burden. This limitation helps streamline the process and allows us to focus more on the core dynamics of the problem, without being hindered by excessive computational demands.

The coding for this thesis has been primarily conducted using Python. We have adhered to object-oriented programming principles to ensure the code is modular, reusable, and easy to understand. The experiments have been run on a MacBook Pro equipped with the Apple M1 Pro chip and 16GB memory, providing a balance between computational power and energy efficiency.

4.1. Demand Settings

Our research necessitates the accurate depiction of real-world scenarios and the incorporation of uncertainty. Therefore, we implemented two core demand settings: deterministic and stochastic. By doing so, we could systematically gauge the strengths and weaknesses of the various algorithms in different contexts.

The deterministic demand setting was devised to emulate scenarios where demand is known with certainty. This is not a common occurrence in real-world scenarios due to the inherent uncertainty and volatility in demand. However, in this research, we used a constant demand of 3 as a representative case. This deterministic demand setting serves a dual purpose. On one hand, it provides a baseline for examining and validating the efficacy of our proposed algorithms. On the other hand, it serves as a litmus test for intuitive reasoning. Given our assumption that the demand will be exactly 3, the optimal policy in this case - even based on a simple, intuitive standpoint - should dictate the ordering of exactly 3 items when the lead times are zero. Therefore, this deterministic scenario provides us a means to test if our algorithms are functioning sensibly by comparing their outputs against this intuitive benchmark.

For stochastic demand, we distinguished between stationary and non-stationary instances. The stationary stochastic demand setting is intended to mimic realistic demand variations when the demand distribution is known. Here, we employ a Poisson distribution as it is commonly applied in the existing literature and select its λ parameter as 3. With known probabilities of each demand occurrence, we could determine the optimal solution using dynamic programming. In contrast, the non-stationary stochastic demand setting is used to capture situations where demand fluctuates over time, accounting for real-world seasonality or trend. Under this setting, dynamic programming typically yields a myopic solution rather than an optimal one. We stored each period's Poisson distribution mean and variance in an array, enabling the generation of varying demand distributions in response to the changing mean.

Importantly, firms frequently receive Advance Demand Information (ADI) from customers for future periods [61]. Such information must be periodically integrated into future demand estimations by managers. Despite the practical relevance of these models, deriving optimal policies using brute-force dynamic programming is computationally intensive due to the large state spaces involved [62]. Reflecting this challenge, we employed an ADI setting incorporating non-stationary aspects. This setting constructs an array of demands for the next three periods, making use of this information. To allow for comparison with other demands, such as the Poisson distribution, the averages of these array demands were set to 3. We assumed that customers can place orders two periods ahead. Thus, in each period t, a demand array $(D_{t,t}, D_{t,t+1}, D_{t,t+2})$ is received, where $D_{t,s}$ is the order placed in period t for period $s \geq t$. The total demand for period t is $D_t = D_{t-2,t} + D_{t-1,t} + D_{t,t}$. This demand setting mirrors the approach used by Chao et al. in their work, where they developed a heuristic to model perishable inventory systems and determined a worst-case cost scenario [45].

4.2. Dynamic Programming

In order to optimally solve the presented inventory problems, we leveraged the technique of dynamic programming. This approach served as a benchmark against which we could evaluate the approximations derived from the RL algorithms.

Our application of dynamic programming began with a backward computation process. Starting from the final time period, T, we computed the expected cost for each possible state. For each subsequent period, $T-1$, we identified the actions that minimized the cumulative costs, incorporating the cost of the current state into our calculations. This iterative process allowed us to determine the minimum possible cost achievable for each state. As a result, we were able to generate a comprehensive policy guide, detailing the best actions to take depending on the current time period and state. In the grand scheme, this serves to streamline the decision-making process, providing clarity amidst the multitude of potential scenarios.

Table 4.1. Expected cost information look-up table for $t = 9$.

$\mathbf{S}_t$	$\mathbf{x}_t$	**Probability**	$\mathbf{D}_t$	**Total Cost**	$\mathbf{S}_{t-1}$	**Minimum Cost**	$\mathbf{x}_{t-1}$
(0, 1, 2)	0	0.049787	0	0.742018	(1, 2, 0)	11.903821	0
(0, 1, 2)	0	0.149361	1	2.129290	(0, 2, 0)	12.255976	0
(0, 1, 2)	0	0.224042	2	3.952295	(0, 1, 0)	16.640883	0
(0, 1, 2)	0	0.224042	3	4.844202	(0, 0, 0)	21.621868	1
(0, 1, 2)	0	0.168031	4	4.977403	(0, 0, 0)	21.621868	1
(0, 1, 2)	0	0.100819	5	3.792992	(0, 0, 0)	21.621868	1
(0, 1, 2)	0	0.050409	6	2.299771	(0, 0, 0)	21.621868	1
(0, 1, 2)	0	0.021604	7	1.158449	(0, 0, 0)	21.621868	1
(0, 1, 2)	0	0.008102	8	0.499230	(0, 0, 0)	21.621868	1

For illustrative purposes, let's assume we're in time period $t = 9$ and the current state is (0,1,2). At each time instance, we generate a lookup table that includes all feasible actions, the maximum of which is restricted to 5 as decided earlier. Table 4.1 shows an example where the chosen action is 0, arbitrarily picked for this example. Following this, we consider all potential outcomes for demand based on the chosen action. These demand scenarios are shaped by a pre-set Poisson distribution, and to simplify the state space, we restrict the demand scenarios to a maximum of 8 units. Any demand exceeding this is deemed to have negligible probability and is therefore omitted from our calculations. Taking a specific action and the corresponding demands into account, we determine the subsequent state. Once we have established the lookup table for each state action demand triplet for given T, we proceed to calculate the final expected costs for each state-action pair. This process involves using the lookup table values, multiplying them by their respective probabilities, and then aggregating the results. A key part of these results is the *to-go* function, which essentially represents the minimum cost for the subsequent state associated with each state-action pair. These calculated values are then compiled into an *expected cost table*.

Table 4.2 illustrates this concept further. Here, we have compiled the costs for time period $t = 10$, computed using the method previously described. Each entry in this table represents the minimum expected costs for state-action pair, allowing us to readily calculate the minimum cost for each pair, as indicated in the minimum_cost column in Table 4.1.

Table 4.2. Expected cost information table for $t = 10$.

$\mathbf{S}_t$	$\mathbf{x}_t$	**Total Cost**
(0, 1, 2)	0	11.903821
(0, 1, 2)	1	20.565404
(0, 1, 2)	2	32.251552
(0, 1, 2)	3	45.752438
(0, 1, 2)	4	60.160693
(0, 1, 2)	5	74.957821

For further clarification, let's consider the first row of Table 4.1. It demonstrates that when we are at state $S_t = (0,1,2)$ and choose action $x_t = 0$. In this scenario, we encounter demand $D_t = 0$. This specific demand occurs with a probability of 0.049787, and under these conditions, we find ourselves in the same state (0,1,2) at the next time period, t=10. By referring back to Table 4.2, we can see that the minimum cost associated with this state is 11.903821. Hence, the total expected cost, amounting to 0.742018, is calculated as $(0.049787 * 11.903821) + (0.049787 * 3 * 1)$. In this specific case, the second part of the calculation refers to the holding cost incurred, as no ordering, shortage or deterioration costs occurred. This calculation is repeated for each possible state-action-demand triplet, and once all potential states are accounted for, the results are aggregated as shown in Table 4.2. These compiled costs are then used to compute the costs for time period t=8. This step-by-step example demonstrates how the dynamic programming approach allows us to systematically compute and compare potential costs, guiding the decision-making process in selecting the most cost-efficient actions.

Although the method we described can be used for all possible states, we have used (0,1,2) just as an example. Our real focus, though, is on situations where we start with no inventory at t=0. That means we are looking at the initial state of (0,0,0). Thus, we report only the costs for this state. It is important to note that we are focusing on this state because it is a common situation - think about a new business starting up or an existing one that's changing how they manage their inventory.

In the deterministic case, as anticipated, the optimal strategy was to order up to 3 units every period. This strategy resulted in a total cost of \$150 for T=10, with the entirety of this cost being attributed to ordering. Similarly, for T=25, the total cost was \$375, which was again solely a result of ordering. These outcomes confirmed our expectations about the cost and order quantities in a deterministic demand scenario. In the stochastic scenarios, we examined both T=10 and T=25 cases and broke down the total cost into individual components. For T=10, the expected costs of ordering, holding, shortage, and deterioration came out to be \$122.92, \$11.09, \$37.80, and \$1.27 respectively. These summed up to a total expected cost of \$173.10. For T=25, these costs were \$311.30, \$29.75, \$70.62, and \$1.53 respectively, which added up to a total cost of \$413.22. Notably, the optimal solutions demonstrate the feasibility of completing the entire planning horizon with very few items deteriorating. Additionally, in the non-stationary case with T=10, we used a demand pattern of [4, 4, 4, 3, 3, 3, 3, 2, 2, 2] as mean demands. This resulted in a myopic solution total expected cost of \$170.98, with the costs of ordering, holding, shortage, and deterioration being \$124.89, \$10.61, \$33.34, and \$2.14 respectively. For T=25, we used a demand pattern of [5, 5, 5, 5, 5, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2] as mean demands. Notably, the average of these mean demands is 3, aligning it with results from the stationary case for comparative analysis. The total expected myopic solution cost here was \$397.65, with the costs of ordering, holding, shortage, and deterioration being \$296.98, \$24.54, \$24.54, and \$3.19 respectively. It's worth noting that it took nearly 3 hours to reach this solution. This lengthy calculation time illustrates the limitations of DP, as its effectiveness decreases with slight increases in the complexity of the problem due to the expanding state space.

Table 4.3 provides a summary of the results from the DP solution. By examining this table, it becomes clear that costs rise as we move from deterministic to stochastic situations. This points to the presence of a *cost of uncertainty*. A striking observation is the pronounced elevation in the shortage cost in the stochastic instances. This can largely be traced back to the predetermined ordering quantity cap, which is set at 5 as previously mentioned.

Table 4.3. Expected cost of DP solution.

Planning Horizon	Demand Type	Demand Pattern	Total Cost	Ordering Cost	Holding Cost	Shortage Cost	Deterioration Cost
10	Deterministic	Stationary	150.0	150.0	0.0	0.0	0.0
10	Stochastic	Stationary	173.10	122.92	11.09	37.80	1.27
10	Stochastic	Non-Stationary	170.98	124.89	10.61	33.34	1.27
25	Deterministic	Stationary	375.0	375.0	0.0	0.0	0.0
25	Stochastic	Stationary	413.22	311.30	29.75	70.62	1.53
25	Stochastic	Non-Stationary	397.65	296.98	24.54	24.54	3.19

4.3. Q-value Approach

Gosavi proposed the Q-value approach, a method that constructs RL algorithms from their DP counterparts, as outlined in Chapter 2. This technique served as a crucial bridge between conventional DP methods and RL, facilitating our understanding of the RL algorithms' performance when the model is fully known—specifically, when the transition probability matrix (TPM) are provided [13].

We adopted Gosavi's approach in our study, guided by the pseudocode that was presented. The results were in perfect alignment with our anticipations—the output mirrored that of the DP solution. This successful implementation not only served as a validation of the accuracy and reliability of the RL algorithm we designed but also effectively showcased the capability of the RL algorithm to duplicate the results of its DP counterpart when given a complete model. The consistency of our implementation was thereby confirmed. In addition to bolstering our confidence in the RL algorithm, this result also emphasizes the value and effectiveness of the Q-value approach in providing a solid benchmark for RL algorithm performance under fully known model conditions.

4.4. Reinforcement Learning

We implemented two widely recognized RL algorithms, namely Q-Learning and SARSA. As we discussed in Section 3.5, the implementation of these algorithms requires the careful tuning of several hyperparameters, such as the exploration strategy, number of episodes, learning rate, discount factor, among others. To optimize these settings, we engaged in thorough experimental design. We identified key factors to consider in our tuning process. These included the number of episodes, the algorithm employed, the exploration strategy, the exploration decay type, exploration parameters, learning decay type, learning rate, and the discount factor. For the deterministic and stationary stochastic demand cases, we used three factors for episode count: 50000, 100000, and 300000; two factors for the algorithm: SARSA and Q-learning; one factor for the exploration strategy: ϵ-greedy; two factors for the exploration decay type: decayed and static, as elaborated in Section 3.5; two factors for the exploration parameter: 0.825 and 0.325; two factors for the learning decay type; two factors for the learning rate: 0.75 and 0.275; and the discount factor, which remained constant across all experiments. We followed the full factorial design logic to guide our experiment design. This comprehensive approach resulted in a total of 54 unique experiment configurations.

Before analyzing the results, it's essential to recognize that some of the settings did not show signs of convergence when compared to the DP solutions. To address this, we developed a simple methodology to identify and categorize these non-converging configurations, thereby excluding them from the primary analysis. Our methodology relies on two key criteria to confirm if a setup has successfully converged. The first criterion checks if the total cost increases monotonically over time. The second criterion compares the cost from the first period (which contains the total average cost of over all periods) of the RL configuration to the DP solution's cost for the same period. If the cost difference surpasses 5%, we infer that the setup still has potential to converge further. Only the configurations that meet both these criteria are labeled as converged.

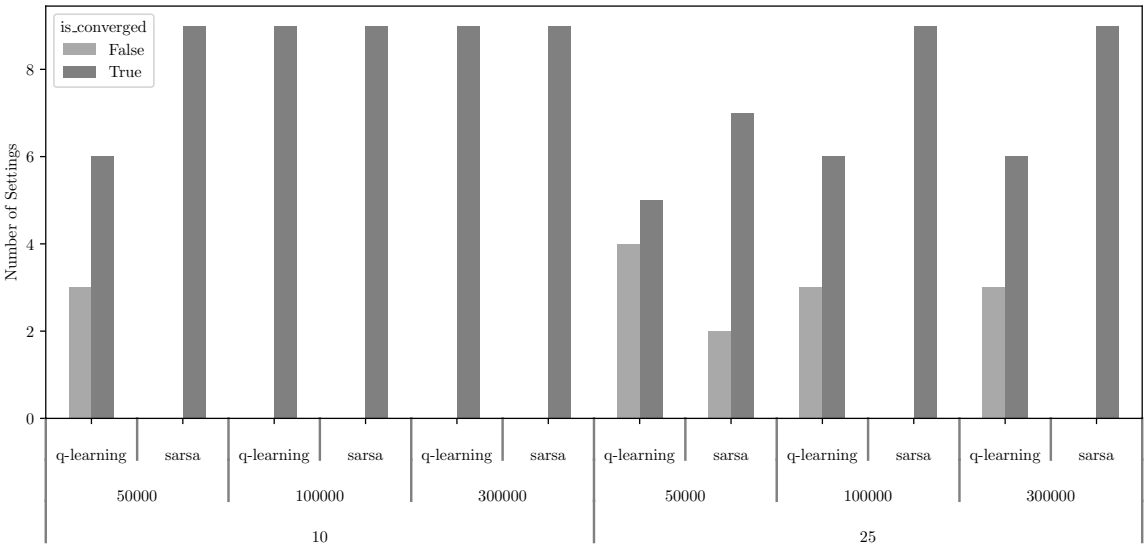

Figure 4.1. Convergence performance of deterministic demand case.

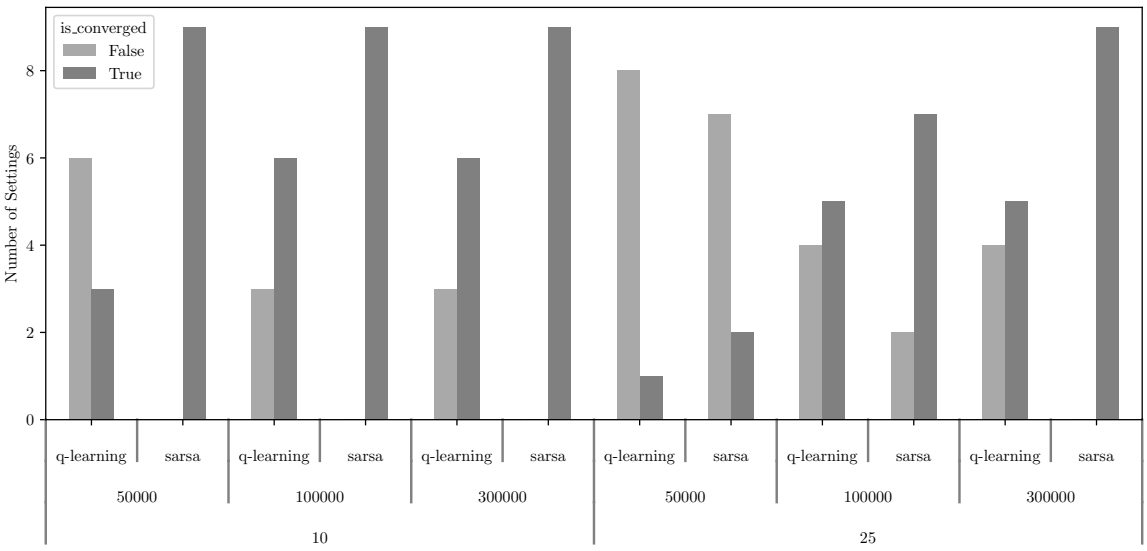

Figure 4.2. Convergence performance of stochastic stationary demand case.

Figures 4.1 and 4.2 depict the convergence performance of each algorithm for planning problems spanning 10 and 25 periods, under various episode counts. These figures offer intriguing insights into the convergence performance of each algorithm. A noticeable trend in both deterministic and stochastic demand settings is that SARSA demonstrates a higher probability of convergence than Q-learning. This observation

aligns with the phenomenon known as the *Cliff Walking* problem, previously discussed in Section 3.4. Additionally, even in the deterministic case, some settings failed to converge. Upon closer examination, it became evident that all non-convergent settings had a static learning rate of 0.85. This implies that, even in the final episodes, the algorithm was attempting to explore the environment. Given that Q-learning aims to find the near-optimal solution in a greedy manner, its failure rate was higher than that of SARSA. This underscores the importance of striking a balance between exploration and exploitation. Moreover, our initial hypothesis suggested that an increased number of episodes would lead to a higher rate of convergence. However, for Q-learning, both graphs show that the number of converged settings remained unchanged between 100.000 and 300.000 episode counts.

To delve deeper into the quantification of the discrepancy from the optimal solution, we computed the squared distance metric. This metric represents the square of the difference between the approximated costs obtained from the RL algorithm for each period, and the corresponding costs found by the DP method. Thus, it indicates the deviation of the RL solution from the DP solution. The rationale for employing this approach was the format of the solution outputs. For instance, in the case of T=10, both DP and RL methods yield cost data for each period. This output can be expressed as an array, allowing a direct comparison between the DP and RL solutions. Figure 4.3 represents the squared distance metrics for the non-converged configurations under the Q-learning method in the stationary stochastic demand scenario. The visual representation shows that even though the number of converged configurations is stable, the squared distances linked to these configurations show a reduction. If we allowed the system to process beyond the 300,000 episodes, we would observe further convergence. In addition, we examined the consistency of the settings to determine if any setting outperformed others with fewer episode counts. In other words, we checked if the same setting with 100.000 episodes could yield better results than with 300.000 episodes. In both deterministic and stochastic demand cases, we did not find any instances where fewer episodes resulted in improved performance. For subsequent analysis, we focus solely on stochastic scenarios due to their greater realism and practical relevance.

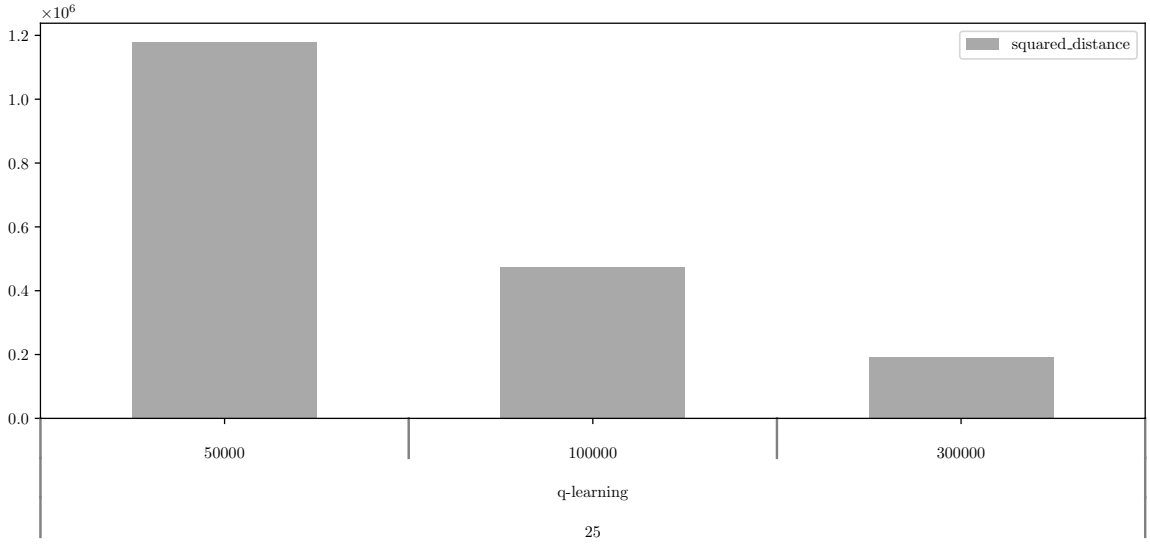

Figure 4.3. Total squared distance for non-convergent settings.

4.4.1. Findings in the Stochastic Stationary Demand

Figure 4.4 and 4.5 provides insight into how the mean squared distance evolves for different algorithms in settings that have achieved convergence under stochastic stationary demand. From this figure, we observe that, when Q-learning converges, it tends to offer solutions that approximate the optimal solution more closely. This behavior, reminiscent of the *Cliff Walking* described in Section 3.4, can be explained by Q-learning's strategy of continuously seeking the minimum cost, whereas SARSA's *play-it-safe* approach may miss nearly optimal solutions.

Furthermore, Figures 4.6 and 4.7 shows the variation in convergence settings across different problem setups. This figure complements our earlier observation that Q-learning, despite its propensity to find solutions nearer to the optimal, has counterparts in SARSA that also exhibit good performance. Additionally, even though we used another metric called cost distance (which is unsquared), the observations remained consistent. We incorporated this metric to better represent real-life relevance, as squaring costs may not always provide the most meaningful insights. Additionally, we introduced a bias metric, to show the cost difference between RL's final period

and the optimal solution. This metric provides insight into the extent of deviation of the overall expected cost from the optimal, and unsurprisingly, the outcomes were consistent with our other findings.

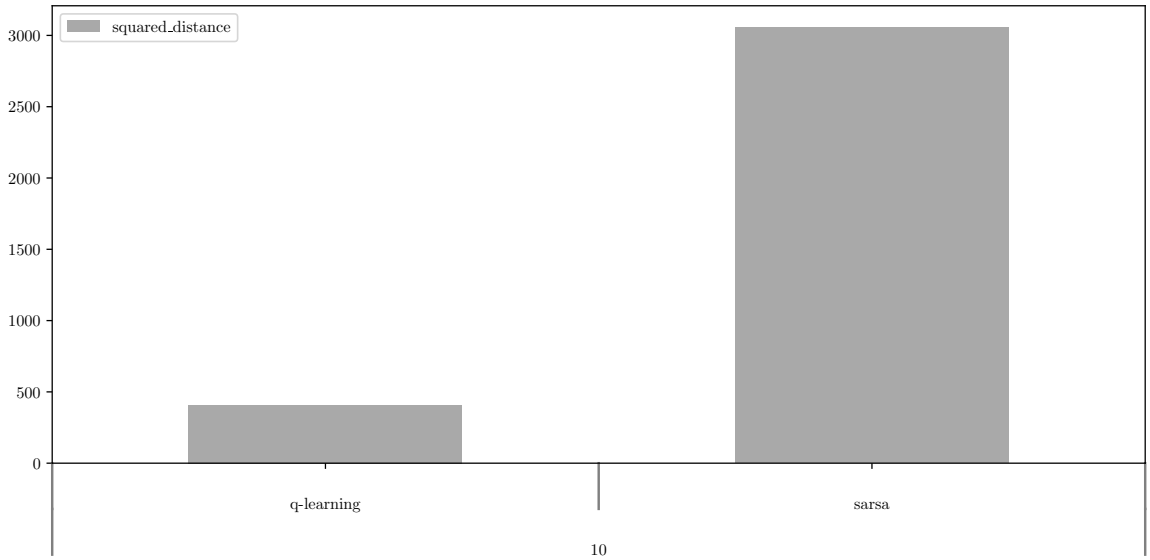

Figure 4.4. Mean squared distance of convergent settings for T=10.

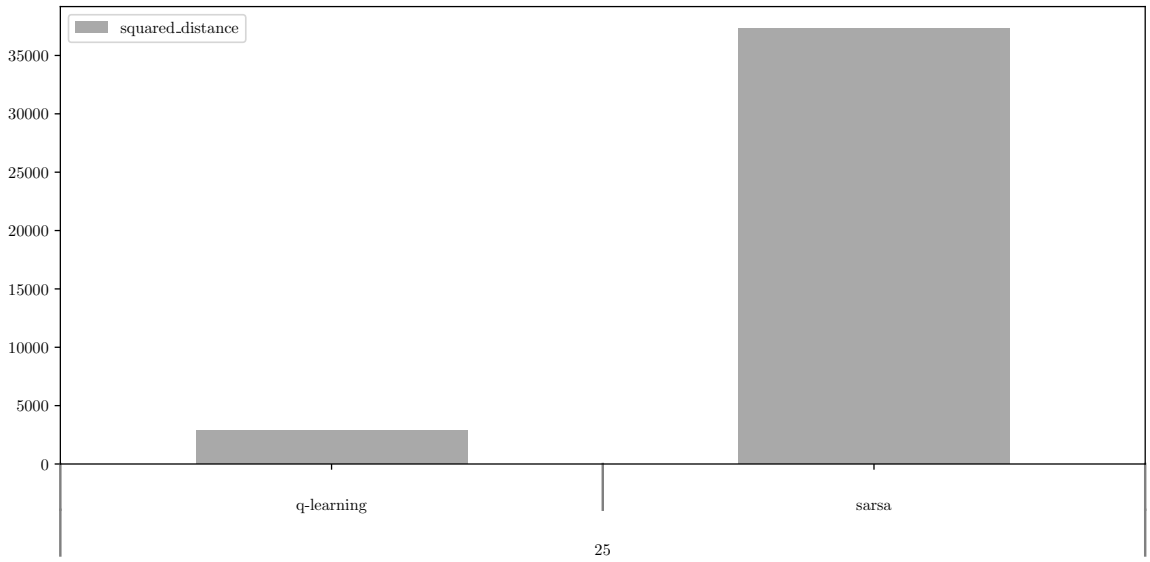

Figure 4.5. Mean squared distance of convergent settings for T=25.

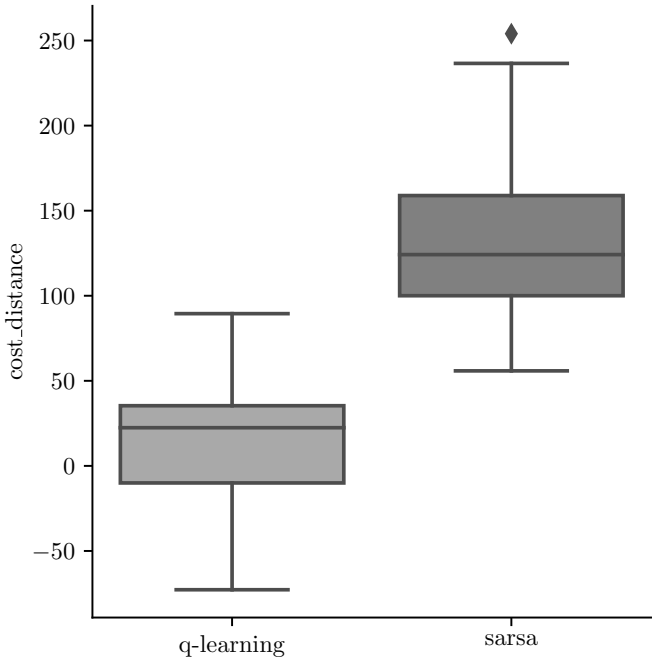

Figure 4.6. Settings' performances for T=10.

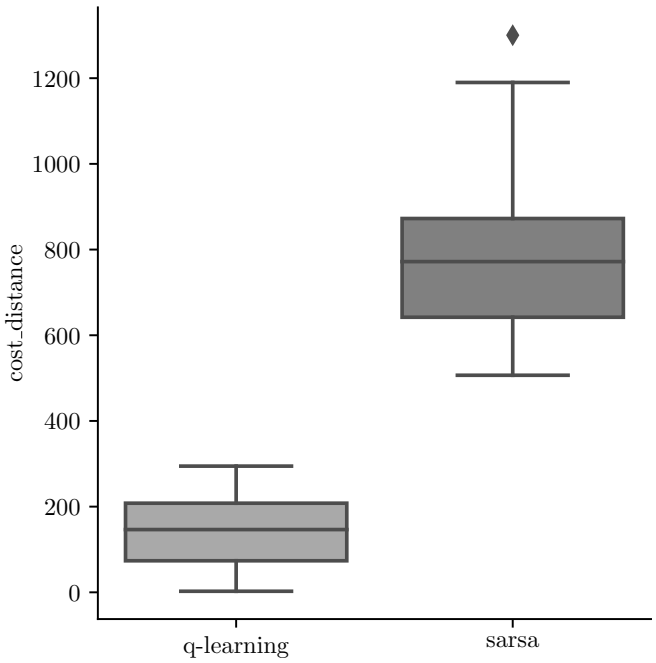

Figure 4.7. Settings' performances for T=25.

In our analysis, we noticed that SARSA exhibited some outlier settings for both planning horizons, primarily due to the application of a high static exploration rate of 0.85. Referring to Figure 4.7, it is evident that as the planning horizon extended, SARSA struggled with exploration and consequently, its solutions diverged significantly from those of Q-learning. From these observations, we distilled the optimal settings for each algorithm, which are outlined in Table 4.4. Analysis of these settings offers insights into the operational differences between the two algorithms. For instance, SARSA, being a more conservative algorithm, requires extensive exploration of the environment. This is evident in its preference for the decayed exploration type, which facilitates greater exploration during initial episodes before exploiting the learnt knowledge. Conversely, Q-learning exhibits a propensity for static exploration, emphasizing its different approach to learning.

Table 4.4. Best settings for stochastic stationary demand.

planning_horizon	algorithm	episode_count	expo_decay_type	expo_parameter	learning_decay_type	learning_rate
10	q-learning	300000	static	0.325	decayed	0.750
10	sarsa	300000	decayed	0.850	decayed	0.750
25	q-learning	300000	static	0.325	static	0.275
25	sarsa	300000	decayed	0.850	decayed	0.750

4.4.2. Findings in the Stochastic Non-Stationary Demand and ADI

Drawing on the convergence data, we chose to increase the episode count for the non-stationary demand scenario to allow for more exploration of the state space, which is slightly larger in this case. We utilized a single episode count factor, set at 500.000, for both problem scenarios. Our observations and conclusions from this experiment were consistent with those obtained in the stationary case. The best settings identified during these experiments are summarized in Table 4.5. Additionally, visual representations emphasizing the results and their implications are included in the Appendix A, Tables A.1, A.2 and Appendix B, Tables B.1, B.2.

Similarly, we incorporated the ADI demand to heighten the complexity, thereby

rendering the problem more representative of real-life scenarios. As with the non-stationary demand situation, we maintained the episode count at 500.000. However, instead of experimenting with varied parameter settings, we solely applied the best settings derived from the non-stationary case for both problem scenarios: static decay and 0.325 exploration for decay learning at a rate of 0.75 for T=10, and static learning with a rate of 0.275 for T=25. Given that the episode count was consistent, the run time remained unchanged.

Table 4.5. Best settings for stochastic non-stationary demand.

planning_horizon	algorithm	episode_count	expo_decay_type	expo_parameter	learning_decay_type	learning_rate
10	q-learning	500000	static	0.325	decayed	0.750
10	sarsa	500000	decayed	0.850	static	0.275
25	q-learning	500000	static	0.325	static	0.275
25	sarsa	500000	decayed	0.850	decayed	0.750

4.5. Comparison of the Results

The true strength of RL algorithms manifests in their ability to approximate costs without full knowledge of the underlying model. Total cost consists of four primary components: ordering, holding, shortage, and deterioration costs. Figures 4.8, 4.9, and 4.10 provide a comparison of these cost components for optimal solutions in deterministic and stationary stochastic cases, and the myopic solution in the non-stationary scenario. In these graphical representations, the darker hues correspond to the cost approximations obtained from the RL algorithms, while the lighter colors signify the costs derived from the DP solutions. Upon examining the visuals, it becomes apparent that Q-learning showcases significant efficiency in approximating the results. Particularly in the deterministic setting for T=10, the ordering costs are almost indistinguishable, while other costs remain negligible. Contrarily, SARSA, in light of our previous observations, appears to be a tad less adept at pinpointing near-optimal solutions, especially in contexts with stochastic demands. This is evidenced by its marginally higher shortage cost, likely due to its heightened exploration tendency.

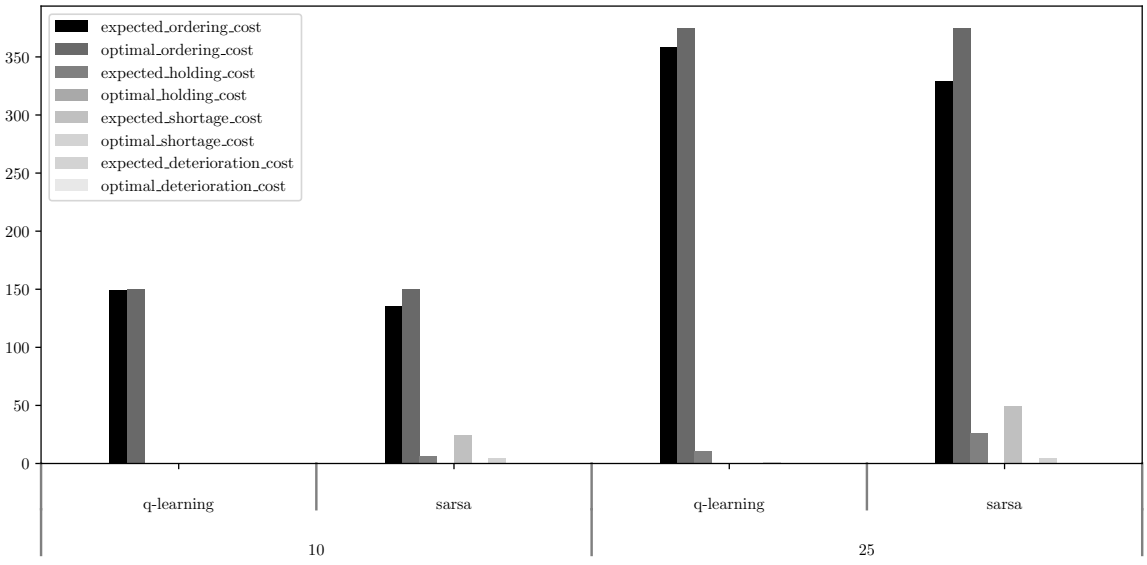

Figure 4.8. Averages of cost components for deterministic demand.

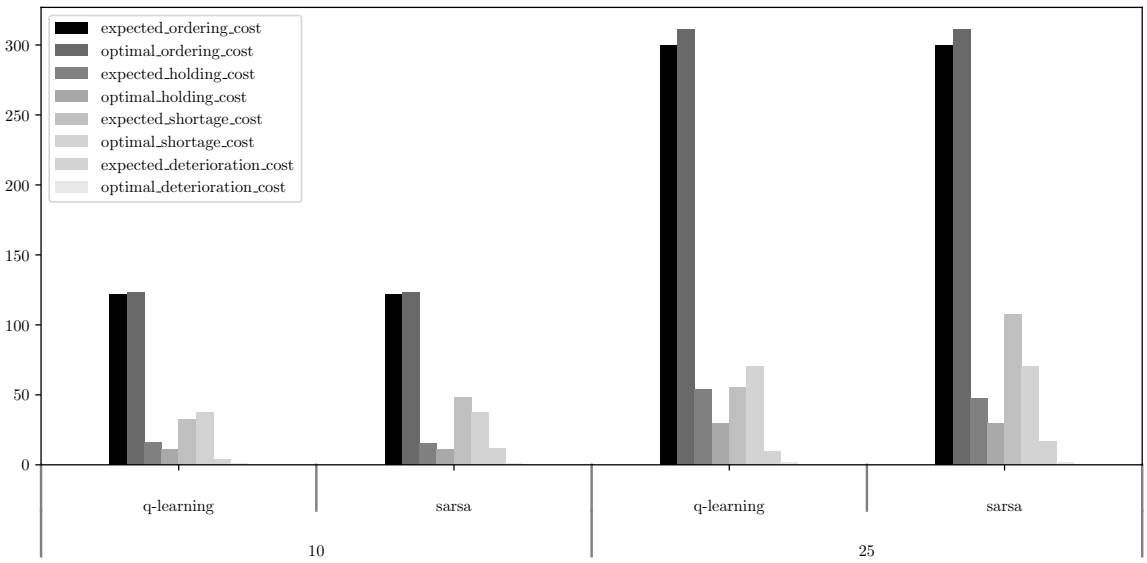

Figure 4.9. Averages of cost components for stochastic stationary demand.

However, it's crucial to bear in mind that the average costs across all converged settings do include some configurations that were not particularly effective, as can be discerned from Figures 4.6 and 4.7. Therefore, to gain a more accurate evaluation of each algorithm's performance, it is advisable to focus on the best configurations for each algorithm.

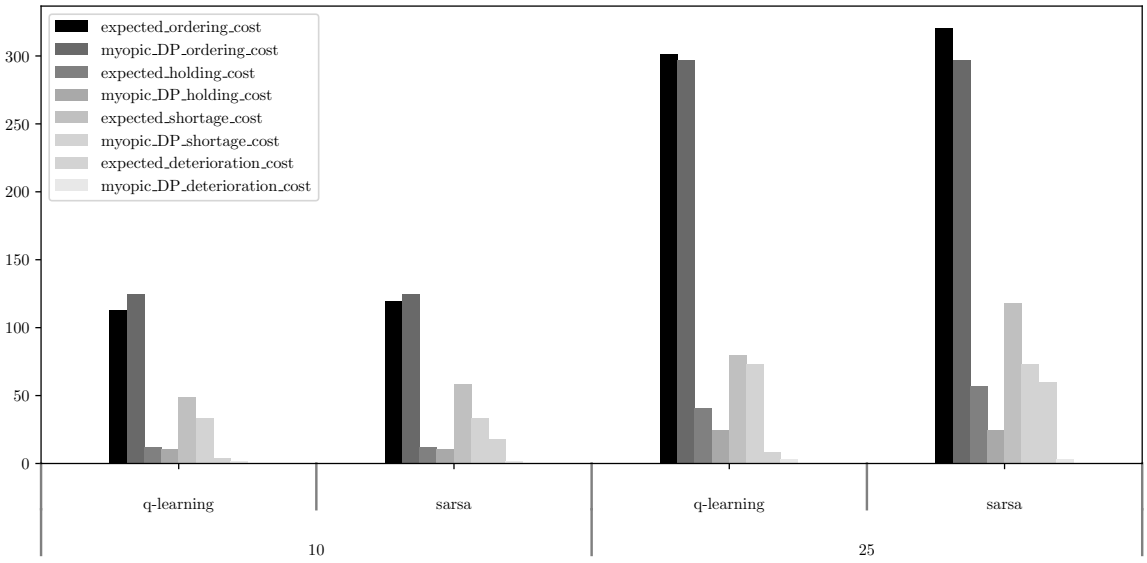

Figure 4.10. Averages of cost components for stochastic non-stationary demand.

By analyzing the best performing configurations for each algorithm, we can discern more nuanced insights about their performance and behavior. Visual comparisons of the best solutions provided by the RL methods for both stationary and non-stationary scenarios can be found in Figures 4.11 and 4.12 respectively. Upon examining the average estimated costs for all settings and the best settings graphs pertaining to the T=25 problem, distinct algorithmic behaviors become evident under non-stationary demand. Specifically, SARSA appears to adopt a more aggressive ordering policy, as evidenced by the increased ordering frequency. On the other hand, Q-learning's ordering policy displays a degree of moderation, with orders either slightly higher or lower across all settings. These variations in ordering policies highlight the differences in the learning strategies adopted by the two algorithms. An intriguing point of comparison arises when contrasting the myopic solution of the DP in a non-stationary context with the RL approximation. Although visuals indicate that the total expected costs are somewhat akin, RL tends to register marginally elevated costs, marked by increased shortage but reduced ordering expenses for both algorithms.

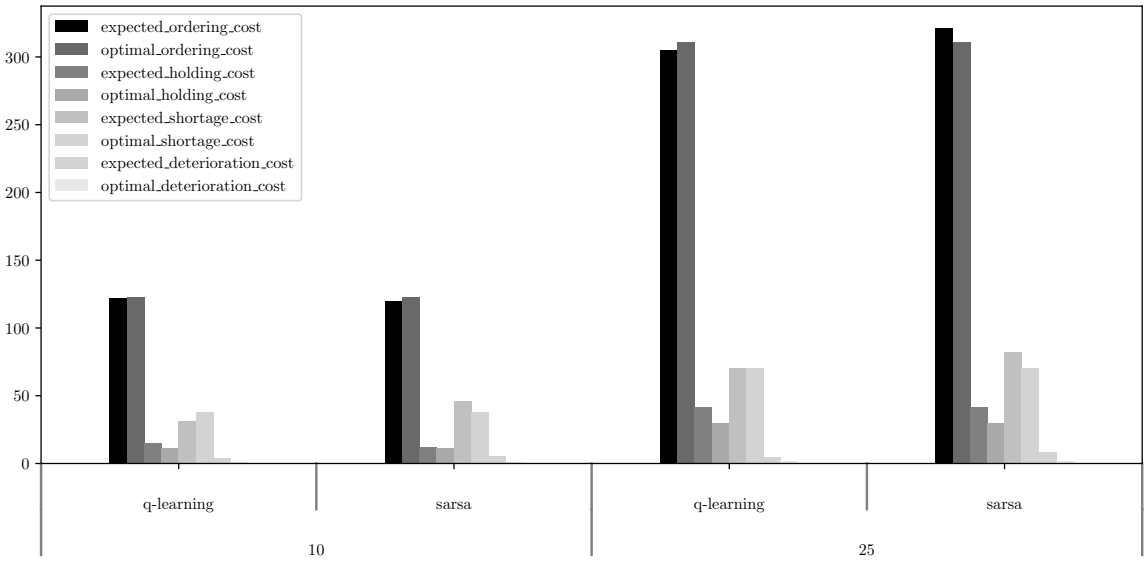

Figure 4.11. Best settings' cost components for stochastic stationary demand.

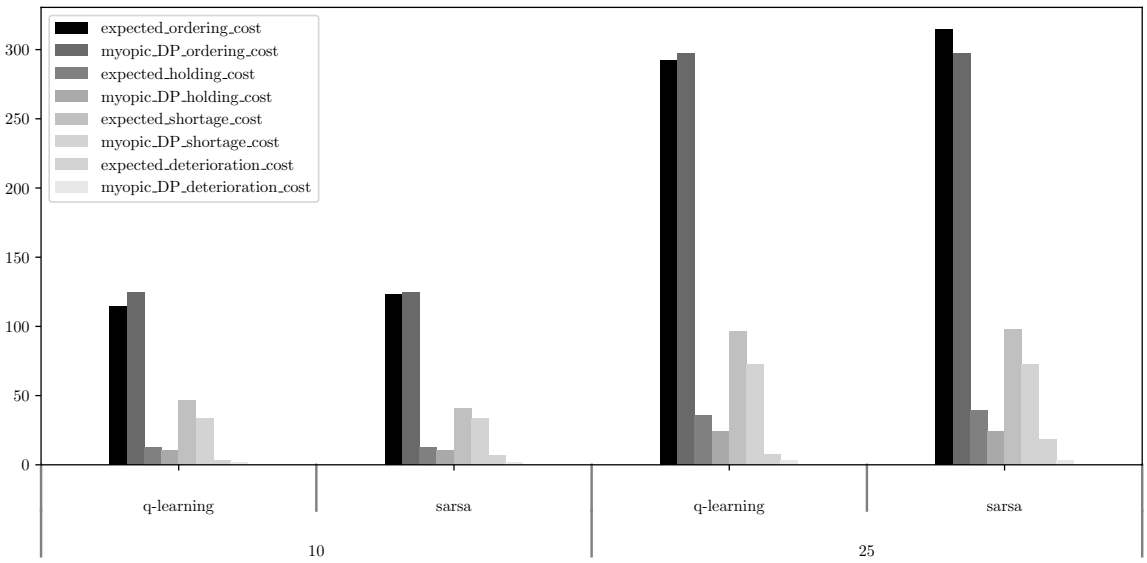

Figure 4.12. Best settings' cost components for stochastic non-stationary demand.

Drawing on the insights previously discussed, we endeavored to approximate costs for the ADI demand scenarios. In the creation of these demands, an empirical approach was adopted. For both stationary and non-stationary stochastic cases, we primarily focused on selecting the mean of demands for each t. This process essentially involved formulating an array embodying the lambda parameter for the Poisson distribution

from which a demand originates. In the context of the ADI demand scenario, we designed an array capturing the demand specifics for each iteration. While our intent was to maintain an overall average demand close to 3, the intrinsic variability of the ADI method naturally led to greater demand fluctuations. However, in the absence of a benchmark solution for these settings, a direct cost comparison is not meaningful. Nevertheless, relying on the proven reliability and consistency of the results demonstrated in the previous scenarios, we can confidently assert that these approximations hold credibility. Even more importantly, they offer a robust framework for tackling real-life demand settings. Not only do these findings offer valuable insights into the performance of our chosen RL algorithms, they also suggest promising avenues for deploying these methods in complex and dynamic real-life demand scenarios, thus broadening applicability and practical significance.

Table 4.6 presents a comprehensive breakdown of the expected costs generated by the best RL solutions under varied demand types and patterns. The costs, spanning deterministic stationary to stochastic non-stationary with ADI, are detailed for two distinct planning horizons: 10 and 25. In the T=10, the deterministic stationary demand exhibits the most favorable total cost outcome at 149.33, solely attributed to ordering costs. When we transition to the stochastic non-stationary demand pattern, there's a notable uptick in the total cost. This increase can be primarily attributed to the emergence of both shortage and deterioration costs. Intriguingly, the ADI pattern for non-stationary demand paints a slightly different picture. The total costs are reduced in comparison to the standard non-stationary scenario, hinting at the potential efficiencies achieved through advanced demand information. For the extended planning horizon of 25 periods, we observe a similar trend with the deterministic stationary scenario leading in terms of minimal total costs. However, as we venture into the stochastic patterns, the costs rise substantially. Of particular note is the significant surge in deterioration costs across all stochastic patterns, emphasizing the inherent challenges of managing perishable items over more extended periods. The elevated shortage cost observed in the stochastic non-stationary scenario echoes the constraints highlighted in the DP solution for stochastic stationary cases, where it is already capped at 5 units.

Table 4.6. Expected cost of best RL solutions.

Planning Horizon	Demand Type	Demand Pattern	Total Cost	Ordering Cost	Holding Cost	Shortage Cost	Deterioration Cost
10	Deterministic	Stationary	149.33	149.33	0.0	0.0	0.0
10	Stochastic	Stationary	178.78	126.7	16.38	31.90	3.8
10	Stochastic	Non-Stationary	176.86	126.0	14.18	33.63	3.05
10	Stochastic	Non-Stationary ADI	162.79	97.04	12.81	48.19	4.75
25	Deterministic	Stationary	370.53	370.53	0.0	0.0	0.0
25	Stochastic	Stationary	417.81	303.81	42.77	66.1	5.13
25	Stochastic	Non-Stationary	429.7	281.08	37.35	103.45	7.83
25	Stochastic	Non-Stationary ADI	374.63	259.45	50.95	50.84	13.4

4.6. Performance of Reinforcement Learning in Large State Spaces

Beyond the insights that RL algorithms can offer in demand scenarios that prove unsolvable via DP, RL methods also provide solutions for scenarios with product lifetimes greater than 3. This challenge is highlighted in Chapter 2, where it is discussed in nearly all the papers that the complexity of such problems increases exponentially, making DP an infeasible solution method. To illustrate this, we attempted to solve a scenario with a product life of 5 and a planning horizon of 16, using both DP and our proposed RL algorithms. We applied the best settings identified for the stochastic stationary case with a T=10. The RL approach completed this task in approximately 3 hours. It found the near-optimal solution for the initial state [0,0,0,0,0] to be \$256.06, which was further broken down into its components: \$185.64 for ordering, \$51.26 for holding, \$17.51 for shortage, and \$1.65 for deterioration costs. However, the DP method continued running for two days without reaching a solution, at which point we decided to halt the process, demonstrating the vast computational inefficiency of DP for such complex problems. This observation underscores the advantages of RL algorithms, particularly in dealing with problem scenarios of higher complexity and longer product lifetimes. RL's ability to provide feasible solutions where traditional methods fail further highlights its practicality and potential for broad applicability in the field of inventory management.

5. CONCLUSION AND FUTURE WORK

Inventory management, especially for perishable goods, remains a pivotal challenge in the realm of supply chain and operations management. Perishable inventory management comes with its own set of intricacies, such as short product lifespans, the risk of spoilage, demand uncertainties, and the need for efficient replenishment strategies. Addressing these challenges not only ensures the smooth operation of the supply chain but also leads to optimal resource utilization and minimizes wastage. Moreover, modeling such inventory systems, given their multifaceted nature, requires us to account for every state of each product. As we scale up the system, the computational intricacy of tracking these states magnifies, presenting significant challenges, especially when using traditional methods. One common approach has been to represent perishable inventory problem as a Markov Decision Process (MDP). However, with this complexity, traditional tools like Dynamic Programming (DP) often stumble, especially as the problem dimensions burgeon. Recognizing these complexities, this research delved into the potential of Reinforcement Learning (RL) algorithms, specifically Q-learning and SARSA, in addressing such challenges. In our analysis, RL methods were compared against traditional DP solutions. In deterministic settings with smaller time horizons, both RL algorithms closely mirrored the results from DP, indicating their reliability and accuracy in simpler scenarios. Particularly, Q-learning showcased an impressive capacity to approximate costs, making it a standout in deterministic contexts.

However, the true strengths of RL became more apparent in stochastic demand scenarios, especially when faced with non-stationary demands. This prowess was further highlighted under the Advanced Demand Information (ADI) condition, a situation where the underlying model is not known, expectations cannot be calculated and more alike situation to real life. Both Q-learning and SARSA adapted effectively, offering robust solutions. But subtle differences in their approaches emerged, with SARSA showing heightened exploration tendencies, leading to slight discrepancies in outcomes.

Beyond just comparing results, visual representation of cost components provided clearer insights into each algorithm's behavior. These visual aids were pivotal in discerning the subtle differences in performance, ordering tendencies, and cost accrual across various scenarios. Moreover, one of the main highlights of the study was the scalability of RL methods. When the product life was extended beyond traditionally manageable limits for DP, RL algorithms continued to provide feasible solutions, emphasizing their computational efficiency. Such findings resonate with the primary challenges addressed in the literature review, wherein the scalability of traditional methods posed significant challenges.

Looking forward, there are several potential avenues for further exploration. Our study predominantly contrasted the RL algorithms with DP solutions. However, a multitude of heuristic methods for solving multi-period perishable inventory problems exist in the literature. These methods, often tailored for specific problem types and constraints, provide an interesting comparison point. Future research could therefore involve implementing and benchmarking RL algorithms against these heuristics, enhancing our understanding of the comparative advantages and limitations of each approach. Moreover, while we observed promising results with RL algorithms, our study did not provide any worst-case performance guarantees. Gosavi proposes such a guarantee in the literature, offering a safety net in terms of the quality of solutions obtained through RL [13]. Implementing this guarantee and assessing the performance of RL algorithms under such a criterion would constitute a valuable future study. This effort would further our understanding of the robustness of RL solutions, giving us insights into their performance, not just on average, but also in the worst-case scenarios.

Furthermore, another interesting extension of this work could be to introduce non-stationary costs in place of deterministic costs. This modification could better model real-life scenarios, such as the impact of inflation rates, seasonal cost, currency rate changes or supply and demand dynamics. Such an enhancement would add a further layer of complexity and realism to the models, opening up additional avenues for exploration and understanding and can be tackled by RL.

In summary, this study illuminated the practical and computational benefits of RL algorithms for solving multi-period perishable inventory problems. However, further research is needed to deepen our understanding of the application and performance of these algorithms, especially in comparison to other heuristic methods and under worst-case performance guarantees. Through such research, we can continue to refine and enhance the efficacy of RL approaches in the realm of inventory management.

REFERENCES

1. "Global Food Losses and Food Waste – Extent, Causes and Prevention", FAO, Rome, https://www.fao.org/3/i2697e/i2697e.pdf, accessed on April 22, 2023.

2. "Reducing Food Loss and Waste", WRI, 2013, https://www.wri.org/research/reducing-food-loss-and-waste, accessed on April 22, 2023.

3. "Food Surplus and Waste in the UK", WRAP, 2020, https://wrap.org.uk/sites/default/files/2020-11/Food-surplus-and-waste-in-the-UK-key-facts-Jan-2020.pdf, accessed on April 22, 2023.

4. "Consumer Expenditures Study: 2019 Data (2021) Progressive Grocer", 2021, https://progressivegrocer.com/consumer-expenditures-study-2019-data, accessed on April 22, 2023.

5. Masoumi, A. H., M. Yu and A. Nagurney, "A Supply Chain Generalized Network Oligopoly Model for Pharmaceuticals Under Brand Differentiation and Perishability", *Transportation Research Part E: Logistics and Transportation Review*, Vol. 48, No. 4, pp. 762–780, 2012.

6. Vila-Parrish, A. R. and J. S. Ivy, *In Handbook of Healthcare Operations Management*, Springer, New York, 2013.

7. Chung, S. H. and C. Kwon, "Integrated Supply Chain Management for Perishable Products: Dynamics and Oligopolistic Competition Perspectives with Application to Pharmaceuticals", *International Journal of Production Economics*, Vol. 179, pp. 117–129, 2016.

8. Nahmias, S., "Perishable Inventory Theory: A Review", *Operations Research*, Vol. 30, No. 4, p. 680–708, 1982.

9. Broekmeulen, R. A. C. M. and K. H. van Donselaar, "A Heuristic to Manage Perishable Inventory with Batch Ordering, Positive Lead-times, and Time-varying Demand", *Computers Operations Research*, Vol. 36, No. 11, pp. 3013–3018, 2009.

10. Sutton, R. S. and A. G. Barto, *Reinforcement Learning - An Introduction*, The MIT Press, Cambridge, MA, 2018.

11. Kaelbling, L. P., M. L. Littman and A. W. Moore, "Reinforcement Learning: A Survey", *Journal of Artificial Intelligence Research*, Vol. 4, p. 237–285, 1996.

12. Abdulhameed, S. A. and S. Lupenko, "Potentials of Reinforcement Learning in Contemporary Scenarios", *Scientific Journal of the Ternopil National Technical University*, Vol. 2, No. 106, p. 92–100, 2022.

13. Gosavi, A., "Reinforcement Learning: A Tutorial Survey and Recent Advances", *INFORMS Journal on Computing*, Vol. 21, No. 2, pp. 178–192, 2009.

14. Ravichandiran, S., *Introduction to Reinforcement Learning in Hands-on Reinforcement Learning with Python: Master Reinforcement and Deep Reinforcement Learning using Openai Gym and Tensorflow*, Packt Publishing, Birmingham, UK, 2018.

15. Giannoccaro, I. and P. Pontrandolfo, "Inventory Management in Supply Chains: A Reinforcement Learning Approach", *International Journal of Production Economics*, Vol. 78, No. 2, pp. 153–161, 2002.

16. Puterman, M. L., *Markov Decision Processes Discrete Stochastic Dynamic Programming*, John Wiley Sons, New York, 2005.

17. Bellman, R. E., "A Markov Decision Process", *Journal of Mathematical Mechanics*, Vol. 6, p. 679–684, 1957.

18. Howard, R., *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, MA, 1960.

19. Lovejoy, W. S., "A Survey of Algorithmic Methods for Partially Observed Markov Decision Processes", *Annals of Operations Research*, Vol. 28, pp. 47–66, 1991.

20. White, D., "Real Applications of Markov Decision Processes", *Interfaces*, Vol. 15, pp. 73–83, 1985.

21. Bertsekas, D. P., "Distributed Dynamic Programming", *IEEE Transactions on Automatic Control*, Vol. 27, pp. 610–616, 1982.

22. Rust, J., *Handbook of Computational Economics*, Elsevier, Amsterdam, 1996.

23. Sutton, R., "Learning to Predict by the Methods of Temporal Differences", *Machine Learning*, Vol. 3, No. 1, pp. 9–44, 1988.

24. Watkins, C. J. C. H., *Learning from Delayed Rewards*, M.S. Thesis, King's College, 1989.

25. Rummery, G. A. and M. Niranjan, *On-line Q-learning using Connectionist Systems*, Technical Report CUED/F-INFENG/TR, Cambridge University, Cambridge, UK, 1994.

26. Mnih, V., K. Kavukcuoglu and D. Silver, "Human-level Control through Deep Reinforcement Learning", *Nature*, Vol. 518, No. 7540, p. 529–533, 2015.

27. Schulman, J., F. Wolski, P. Dhariwal and A. Radford, "Proximal Policy Optimization Algorithms", ArXiv:1707.06347 [cs.LG], 2017.

28. Goyal, S. K. and B. C. Giri, "Recent Trends in Modeling of Deteriorating Inventory", *European Journal of Operational Research*, Vol. 134, No. 1, pp. 1–16, 2001.

29. Nahmias, S., "Continuous Review Perishable Inventory Models", *International Series in Operations Research and Management Science*, Vol. 160, p. 25–31, 2011.

30. Weiss, H. J., "Optimal Ordering Policies for Continuous Review Perishable Inventory", *Operations Research*, Vol. 28, No. 2, p. 365–374, 1980.

31. Goh, C. H., B. S. Greenberg and H. Matsuo, "Two-stage Perishable Inventory Models", *Management Science*, Vol. 39, No. 5, p. 633–649, 1993.

32. Liu, L. and Z. Lian, "(S, s) Continuous Review Models for Products with Fixed Lifetimes", *Operations Research*, Vol. 47, No. 1, p. 150–158, 1999.

33. Friedman, Y. and Y. Hoch, "A Dynamic Lot-size Model with Inventory Deterioration", *INFOR: Information Systems and Operational Research*, Vol. 16, No. 2, p. 183–188, 1978.

34. Parlar, M., D. Perry and W. Stadje, "FIFO versus LIFO Issuing Policies for Stochastic Perishable Inventory Systems", *Methodology and Computing in Applied Probability*, Vol. 13, No. 2, p. 405–417, 2010.

35. Chen, L. M. and A. Sapra, "Joint Inventory and Pricing Decisions for Perishable Products with Two-period Lifetime", *Naval Research Logistics (NRL)*, Vol. 60, No. 5, p. 343–366, 2013.

36. Chen, X., Z. Pang and L. Pan, "Coordinating Inventory Control and Pricing Strategies for Perishable Products", *Operations Research*, Vol. 62, No. 2, p. 284–300, 2014.

37. Li, Q., P. Yu and X. Wu, "Managing Perishable Inventories in Retailing: Replenishment, Clearance sales, and Segregation", *Operations Research*, Vol. 64, No. 6, p. 1270–1284, 2016.

38. Pierskalla, W. P. and C. D. Roach, "Optimal Issuing Policies for Perishable Inventory", *Management Science*, Vol. 18, No. 11, p. 603–614, 1972.

39. Van Zyl, G. J. J., *Inventory Control for Perishable Commodities*, M.S. Thesis, University of North Carolina, 1964.

40. Nahmias, S. and W. P. Pierskalla, "Optimal Ordering Policies for a Product that Perishes in Two Periods subject to Stochastic Demand", *Naval Research Logistics Quarterly*, Vol. 20, No. 2, p. 207–229, 1973.

41. Nahmias, S., "Optimal Ordering Policies for Perishable Inventory—II", *Operations Research*, Vol. 23, No. 4, p. 735–749, 1975.

42. Fries, B., "Optimal Ordering Policy for a Perishable Commodity with Fixed Lifetime", *Operations Research*, Vol. 23, No. 1, p. 46–61, 1975.

43. Nahmias, S., "Myopic Approximations for the Perishable Inventory Problem", *Management Science*, Vol. 22, No. 9, pp. 1002–1008, 1976.

44. Nandakumar, P. and T. E. Morton, "Near Myopic Heuristics for the Fixed-life Perishable Inventory Problem", *Management Science*, Vol. 39, pp. 1490–1498, 1993.

45. Chao, X., X. Gong, C. Shi and H. Zhang, "Approximation Algorithms for Perishable Inventory Systems", *Operations Research*, Vol. 63, No. 3, p. 585–601, 2015.

46. Tekin, E., U. Gurler and E. Berk, "Age-based vs. Stock Level Control Policies for a Perishable Inventory System", *European Journal of Operational Research*, Vol. 134, No. 2, p. 309–329, 2001.

47. Van Roy, B., D. P. Bertsekas, Y. Lee and J. N. Tsitsiklis, "A Neuro-dynamic Programming Approach to Retailer Inventory Management", *36th IEEE Conference on Decision and Control*, San Diego, California, USA, 1997.

48. Rao, J. J., K. K. Ravulapati and T. K. Das, "A Simulation-based Approach to Study Stochastic Inventory-planning Games", *International Journal of Systems Science*, Vol. 34, No. 12-13, p. 717–730, 1993.

49. Tongeren, T., U. Kaymak, D. Naso and V. A. E., "Q-learning in a Competitive Supply Chain", *IEEE International Conference on Systems, Man and Cybernetics*, p. 1211–1216, 2007.

50. Chaharsooghi, S., J. Heydari and S. Zegordi, "A Reinforcement Learning Model for Supply Chain Ordering Management: An Application to the Beer Game", *Decision Support Systems*, Vol. 45, pp. 949–959, 2008.

51. Sui, Z., A. Gosavi and L. Lin, "A Reinforcement Learning Approach for Inventory Replenishment in Vendor-Managed Inventory Systems With Consignment Inventory", *Engineering Management Journal*, Vol. 22, pp. 44–53, 2010.

52. Nurkasanah, I., "Reinforcement Learning Approach for Efficient Inventory Policy in Multi-Echelon Supply Chain Under Various Assumptions and Constraints", *Journal of Information Systems Engineering and Business Intelligence*, Vol. 7, p. 138, 2021.

53. Kara, A. and I. Dogan, "Reinforcement Learning Approaches for Specifying Ordering Policies of Perishable Inventory Systems", *Expert Systems with Applications*, Vol. 91, p. 150–158, 2018.

54. Selukar, M., P. Jain and T. Kumar, "Inventory Control of Multiple Perishable Goods using Deep Reinforcement Learning for Sustainable Environment", *Sustainable Energy Technologies and Assessments*, Vol. 52, 2022.

55. Fernie, J. and L. Sparks, *Logistics and Retail Management: Emerging Issues and New Challenges in the Retail Supply Chain*, Kogan Page, London, 2018.

56. Powell, W., *Sequential Decision Analytics and Modeling*, Foundations and Trends in Technology, Information and Operations Management, Boston, 2022.

57. Silver, E. A., D. F. Pyke and D. J. Thomas, *Inventory and Production Management in Supply Chains*, Taylor Francis Group, Boca Raton, 2016.

58. Bellman, R., *Dynamic Programming*, Princeton University Press, USA, 1957.

59. Tijsma, A. D., M. M. Drugan and M. A. Wiering, "Comparing Exploration Strategies for Q-learning in Random Stochastic Mazes", .

60. Jaakkola, T., M. I. Jordan and S. P. Singh, "On the Convergence of Stochastic Iterative Dynamic Programming Algorithms", *Neural Computation*, Vol. 6, No. 6, 1994.

61. Gallego, G. and O. Ozer, "Integrating Replenishment Decisions with Advance Demand Information", *Management Science*, Vol. 47, No. 10, p. 1344–1360, 2001.

62. Lu, X., J. S. Song and R. A., "Inventory Planning with Forecast Updates: Approximate Solutions and Cost Error Bounds", *Operations Research*, Vol. 54, No. 6, p. 1079–1097, 2006.

63. Veinott, A. F., *Optimal Ordering, Issuing and Disposal of Inventory with Known Demand*, M.S. Thesis, Columbia University, 1960.

APPENDIX A: MEAN SQUARED DISTANCE OF CONVERGENT SETTINGS

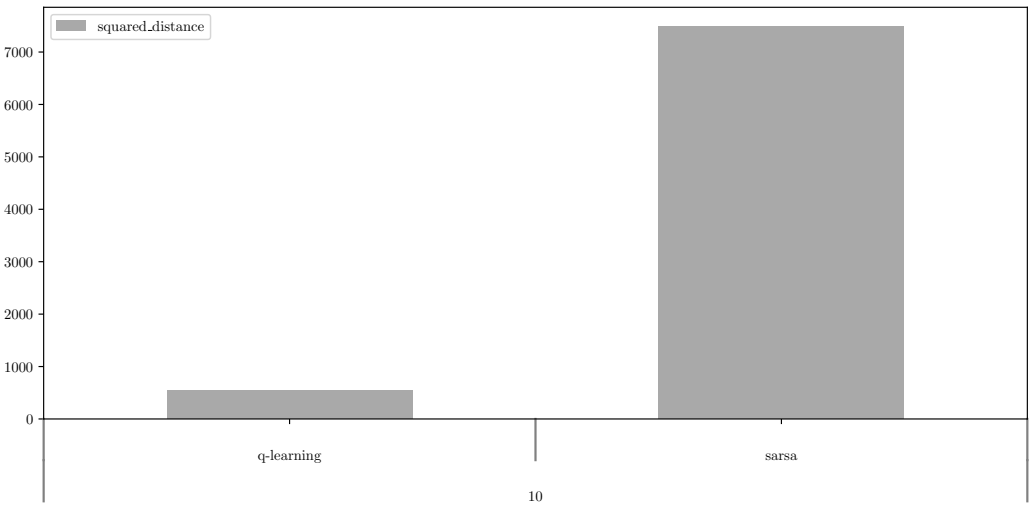

Figure A.1. Convergent Settings for T=10.

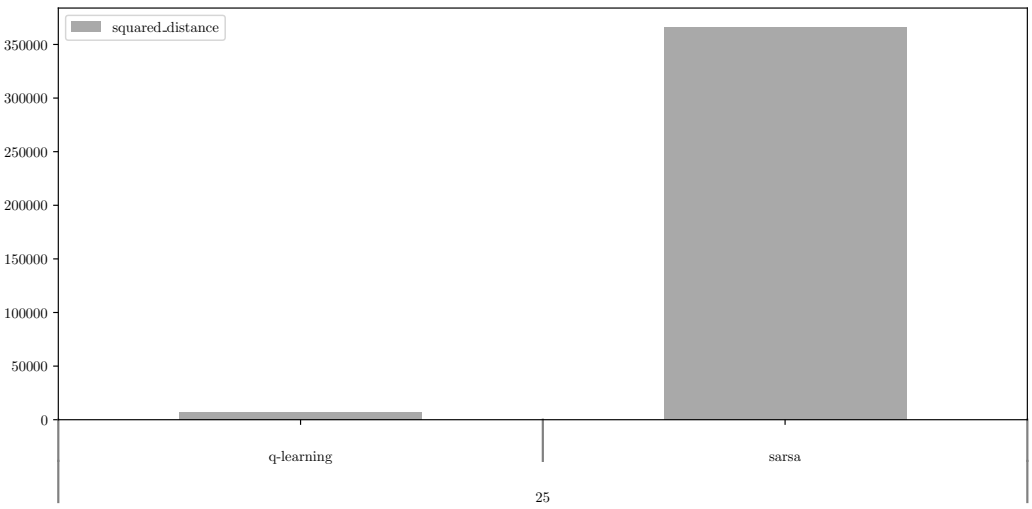

Figure A.2. Convergent Settings for T=25.

APPENDIX B: SETTINGS' PERFORMANCE

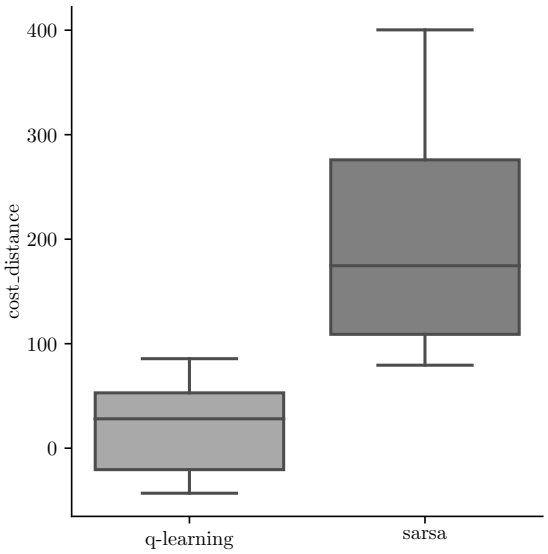

Figure B.1. Performance for T=10.

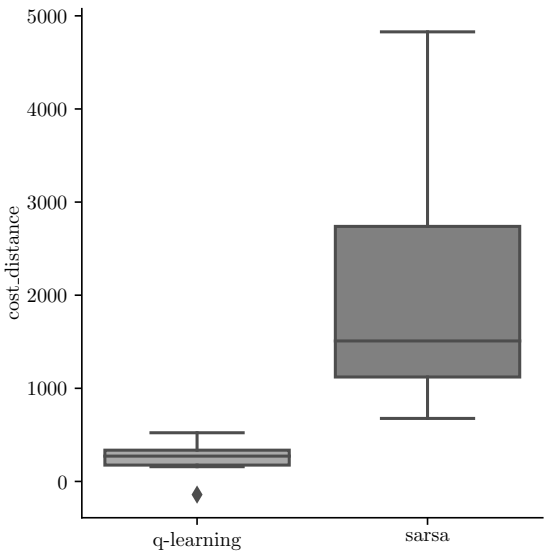

Figure B.2. Performance for T=25.