

SOLVING PERISHABLE INVENTORY CONTROL PROBLEMS IN A
COMPETITIVE ENVIRONMENT USING REINFORCEMENT LEARNING

by

Anıl Turgut

B.S., Industrial Engineering, Özyeğin University, 2022

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

2025

ACKNOWLEDGEMENTS

The dissertation journey was a profound experience, filled with learning, research, and countless late nights and weekends spent immersed in work. Despite the challenges, I found satisfaction and joy in every step of the process.

First, I would like to express my heartfelt gratitude to my supervisor, Prof. Taner Bilgiç, for his invaluable guidance, encouragement, and expertise. His insightful advice and unwavering support have been instrumental in shaping the direction and quality of this work. I am deeply grateful for the patience and belief he has shown in me during this journey.

I am also profoundly grateful to the members of my thesis committee, Prof. Refik Güllü and Assoc. Prof. Enis Kayış, for their constructive feedback. Their expertise and careful evaluation have greatly improved the depth and rigor of this thesis.

I would like to extend my sincere thanks to my family and friends, whose unwavering support and love have been a constant source of strength. They have always been with me, offering encouragement and understanding throughout every step of this journey.

ABSTRACT

SOLVING PERISHABLE INVENTORY CONTROL PROBLEMS IN A COMPETITIVE ENVIRONMENT USING REINFORCEMENT LEARNING

The perishable inventory control problem in a competitive environment is the problem where multiple players compete with other players by managing their own inventory systems. The perishable inventory control problem, which requires a complex sequential decision-making problem with replenishment decisions in each period, cannot be efficiently solved with DP-based equilibrium algorithms due to algorithmic and time complexity. Therefore, algorithms are developed using approximation techniques to solve the problem in this competitive environment.

To the best of our knowledge, this is the first study that solves the perishable inventory control problem in a competitive environment with Reinforcement Learning (RL) algorithms. In this thesis, the perishable inventory control problem in a competitive environment is analyzed in cases against fixed replenishment and dynamic replenishment, and equilibrium solutions are obtained for both cases with the RL algorithm. DP-based equilibrium solutions are approximated with algorithms implemented using Nash-Q Learning, one of the techniques in the Multi-agent RL (MARL) world. An experimental design with different parameters is prepared for both cases in a competitive environment. The Nash Q-Learning algorithm developed for both cases converges to DP-based equilibrium solutions in a considerably shorter time than the runtime of the DP-based algorithm. In addition, it is observed that the agents trained with the Nash Q-Learning algorithm have a higher win-rate in the competitive environment. Learning rate and learning policy can lead to competitive advantage for the agents.

ÖZET

REKABETÇİ ORTAMDA PEKİŞTİRMELİ ÖĞRENMEYİ KULLANARAK BOZULABİLİR ENVANTER KONTROL PROBLEMLERİNİ ÇÖZME

Rekabetçi bir ortamda bozulabilir envanter kontrol problemi, birden fazla oyuncunun kendi envanter sistemlerini yöneterek diğer oyuncularla rekabet ettiği problemdir. Karmaşık bir ardışık karar alma gerektiren bozulabilir envanter kontrol problemi, algoritmik ve zaman karmaşıklığı nedeniyle dinamik programlama (DP) tabanlı denge algoritmalarıyla verimli bir şekilde çözülemez. Bu nedenle, rekabetçi ortamda problemi çözmek için yaklaşıklık teknikleri kullanılarak algoritmalar geliştirilir.

Bildiğimiz kadarıyla bu tez, rekabetçi bir ortamda bozulabilir envanter kontrol problemini Pekiştirmeli Öğrenme (PÖ) algoritmaları ile çözen ilk çalışmadır. Bu tezde, rekabetçi bir ortamda bozulabilir envanter kontrol problemi, sabit ikmal ve dinamik ikmal durumlarına karşı analiz edilmiş ve her iki durum için de PÖ algoritması ile denge çözümleri elde edilmiştir. DP tabanlı denge çözümleri, çoklu aktör dünyasındaki tekniklerden olan Nash-Q Öğrenmesi kullanılarak uygulanan algoritmalarla yaklaşık olarak hesaplanmaktadır. Rekabetçi bir ortamda her iki durum için de farklı parametrelere sahip deneysel bir tasarım hazırlanmaktadır. Testler sonucunda, her iki durum için geliştirilen Nash Q-Öğrenmesi algoritması, DP tabanlı algoritmanın çalışma süresinden önemli ölçüde daha kısa bir sürede DP tabanlı denge çözümlerine yakınsar. Ayrıca, Nash Q-Öğrenmesi algoritmasıyla eğitilen ajanların daha yüksek bir kazanma oranına sahip olduğu gözlemlenmektedir. Öğrenme hızı ve yordamının rekabetçi avantaj sağladığı durumlar gözlenmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
LIST OF SYMBOLS	xi
LIST OF ACRONYMS/ABBREVIATIONS	xii
1. INTRODUCTION	1
2. LITERATURE REVIEW	5
2.1. Perishable Inventory Control	5
2.2. Reinforcement Learning Approach	7
2.3. Multiagent Reinforcement Learning & Game Theory	10
3. PROBLEM FORMULATION	14
3.1. Problem Definition	14
3.2. States and Actions	17
3.3. Demand Structure	19
3.4. State Transition	23
3.5. Cost Structure	24
3.6. Single Agent MDP Formulation	25
3.6.1. Exact Solution Formulation	25
3.6.2. Q-Value Approximation	27
3.7. Multi-Agent Markov Game Formulation	28
3.7.1. Nash Equilibrium	29
3.7.2. Q-Value Approximation	32
3.7.3. Nash Q-Learning Algorithm	34
4. COMPUTATIONAL RESULTS	39
4.1. Experimental Design	40
4.2. Competitive Nash-Q Learning under Fixed Replenishment	43

4.3. Competitive Nash-Q Learning under Dynamic Replenishment 53

4.4. Summary 64

5. CONCLUSION 66

REFERENCES 69

APPENDIX A: DETAILED COMPUTATIONAL RESULTS ON THE
NASH Q-LEARNING UNDER FIXED REPLENISHMENT . 73

APPENDIX B: DETAILED COMPUTATIONAL RESULTS ON THE
NASH Q-LEARNING UNDER DYNAMIC REPLENISHMENT 83



LIST OF FIGURES

Figure 3.1.	Single-agent reinforcement learning diagram.	15
Figure 3.2.	Multi-agent reinforcement learning diagram.	16
Figure 3.3.	Order of events.	18
Figure 3.4.	Algorithm: stage equilibrium and game equilibrium procedure. . .	31
Figure 3.5.	Algorithm: nash q-learning against fixed replenishment.	37
Figure 3.6.	Algorithm: nash q-learning against dynamic replenishment.	38
Figure 4.1.	Exponentially decaying learning rate.	42
Figure 4.2.	Fixed replenishment: cost gap (%) to dynamic programming. . . .	48
Figure 4.3.	Fixed replenishment: demand capture percentage of trained agent.	51
Figure 4.4.	Algorithm performance: epsilon greedy - $\alpha = 0.7$	58
Figure 4.5.	Algorithm performance comparison: greedy - $\alpha = 0.5$ vs. epsilon greedy - $\alpha = 0.7$	59

LIST OF TABLES

Table 4.1.	Test design parameters.	41
Table 4.2.	Fixed replenishment: training runtime (hours).	45
Table 4.3.	Fixed replenishment: cost gap (%) to dp-based equilibrium.	47
Table 4.4.	Fixed replenishment: win-rate (%) under $m = 2$	49
Table 4.5.	Fixed replenishment: win-rate (%) under $m = 3$	52
Table 4.6.	Dynamic replenishment: training runtime (hours).	54
Table 4.7.	Dynamic replenishment ($m = 2, T = 6$): cost gap (%) to equilibrium solution.	56
Table 4.8.	Dynamic replenishment ($m = 2, T = 10$): cost gap (%) to equilibrium solution.	57
Table 4.9.	Dynamic replenishment: win-rate (%) under $m = 2$ and $T = 6$. . .	59
Table 4.10.	Dynamic replenishment: win-rate (%) under $m = 2$ and $T = 10$. . .	61
Table 4.11.	Dynamic replenishment: win-rate (%) under $m = 3$ and $T = 6$. . .	62
Table 4.12.	Dynamic replenishment: win-rate (%) under $m = 3$ and $T = 10$. . .	63
Table A.1.	Fixed replenishment: detailed training runtime (hours).	73

Table A.2.	Fixed replenishment: detailed cost gap (%) to dp-based equilibrium.	74
Table A.3.	Fixed replenishment: detailed win-rate (%) under $m = 2$	78
Table A.4.	Fixed replenishment: detailed win-rate (%) under $m = 3$	81
Table B.1.	Dynamic replenishment: detailed training runtime (hours).	83
Table B.2.	Dynamic replenishment ($m = 2, T = 6$): detailed cost gap (%) to equilibrium solution.	84
Table B.3.	Dynamic replenishment ($m = 2, T = 10$): detailed cost gap (%) to equilibrium solution.	86
Table B.4.	Dynamic replenishment: detailed win-rate (%) under $m = 2$ and $T = 6$	88
Table B.5.	Dynamic replenishment: detailed win-rate (%) under $m = 2$ and $T = 10$	90
Table B.6.	Dynamic replenishment: detailed win-rate (%) under $m = 3$ and $T = 6$	91
Table B.7.	Dynamic replenishment: detailed win-rate (%) under $m = 3$ and $T = 10$	93

LIST OF SYMBOLS

A	Action Space
C_t^j	Cost of player j at the end of period t
D_t	Incoming demand at period t
D_t^j	Allocated demand to the player j at period t
E	Maximum number of epochs
F_j	Fulfillment metric of the player j
m	Product lifetime
Q_t^j	Q-function of player j at period t
S	State Space, $S = (\dots)$
$\bar{S}_t$	Joint inventory state at period t
S_t^j	Inventory state of player j at period t
$S_{t,i}^j$	Quantity of on-hand whose remaining lifetime is i periods for player j at the beginning of period t
T	Planning horizon
V_t^j	Expected cost of player j at period t
V_π^j	Expected cost of player j under policy π
w_n	Weight of normalized inventory difference
w_v	Weight of value gap
X_t^j	Quantity ordered by player j at the beginning of period t
y_t^j	Inventory level of the player j at period t
α	Learning rate of Nash-Q Learning
ϵ	Epsilon value in the Epsilon-Greedy strategy
λ	Mean parameter of Poisson distribution
π^j	Replenishment policy of the player j
σ	Decaying rate

LIST OF ACRONYMS/ABBREVIATIONS

ADP	Adaptive/Approximate Dynamic Programming
BOE	Bellman's Optimality Equation
DP	Dynamic Programming
DRL	Deep Reinforcement Learning
FIFO	First in First out
GT	Game Theory
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
NE	Nash Equilibrium
NID	Normalized Inventory Difference
NN	Neural Network
NQL	Nash Q-Learning
RL	Reinforcement Learning
TD	Temporal Differences

1. INTRODUCTION

Effective inventory management is among the most important building blocks of company profitability and process sustainability. With the importance given to this mechanism, customer demands are satisfied in a timely manner and revenue will be maximized, as both customer satisfaction and any possible lost sales will be prevented. One of the critical operations in inventory management is the replenishment of the goods in the inventory at certain periods. This replenishment policy and how much goods will be stored at the end of the period to meet customer demands in the next period is a critical problem. In addition, the question of which types of goods will be managed in the inventory also increases the complexity of the problem because timing becomes crucial when products can expire. In particular, perishable inventory management causes the curse of modeling/dimensionality because it must take into account that the goods are worthless after a certain period.

The perishable inventory control problem has real-life applications in areas such as healthcare and pharmaceuticals. In poorly managed inventory systems, there is a significant cost associated with products going to waste. Compared to businesses without well-defined and regularly implemented inventory control procedures, companies report having 26% lower inventory levels, which leads to 30% more inventory turns and 15% less shrinkage [1]. Inventory management plays a critical role in terms of both product sustainability and cost, especially considering that most products in the pharmaceutical field are perishable. This problem, which has an important place in the health sector, is especially vital for the management of blood inventory systems and there are pioneering studies on this application in the literature [2, 3].

In the single-agent perishable inventory control problem, a single inventory system in the environment tries to meet the incoming demand by replenishing its inventory by making a replenishment decision at every period. Thus, it minimizes major costs such as the shortage and the expiration due to perishing products.

Researchers often approach this problem with the Dynamic Programming (DP) technique to complete the formulation and obtain optimal replenishment policies [2, 3]. However, since DP involves recursive iterations, the algorithms developed with this technique have memory and computationally burden as the state space grows. Thus, researchers are moving towards more efficient algorithms using approximation techniques [4].

In a competitive environment where multiple perishable inventory systems compete with each other, inventory replenishment decisions are a crucial factor determining competition between inventory systems. Unlike in the single-agent world, each player in a competitive environment not only makes replenishment decisions based on their current inventory state and period, but also considers the inventory state of other players and the replenishment decisions they may make. Thus, the inherently complex problem of perishable inventory control becomes an even more complex problem in a competitive environment. Solving the perishable inventory control problem in the multi-agent world via an N-player Markov Game and obtain equilibrium replenishment decisions might be a possible approach. However, this approach is costly in terms of memory and time due to the algorithmic complexity. Therefore, unlike DP-based approaches, it is necessary to approach the problem with an approximation technique.

In this thesis, our objective is to solve the perishable inventory control problem in a competitive environment using Reinforcement Learning (RL). Since RL provides approximate solutions to complex sequential decision-making problems, we use RL as an approximation of DP-based solutions in this problem. Approximate solutions to the Markov Decision Process (MDP) in the single-agent world are generated using RL algorithms such as Q-Learning and Temporal Differences (TD) [5, 6]. Studies implemented in real-life applications using RL approaches such as TD-Learning show that agents trained with these algorithms make decisions at the level of human logic. For example, in the TD-Gammon study, the agent reaches a level where it can compete with professional backgammon players by self-play [7].

In the multi-agent world, where multiple agents compete with each other, Multi-Agent Reinforcement Learning (MARL) techniques such as Nash Q-Learning are used to solve the N-player Markov game [8]. We aim to formulate the perishable inventory control problem in the competitive environment in the MARL world and solve the problem using the RL algorithm. In addition to solving the problem, we also investigate whether there are dominant policies in the game by running algorithms under different parameters.

We evaluate the perishable inventory control problem in the competitive environment against two cases, fixed replenishment and dynamic replenishment. The main difference between these two cases is the access of the players in the competitive environment to the action space. In the fixed replenishment case, a player can experience different replenishment actions in the action space, while the other players perform a predetermined amount of fixed replenishment in each period. On the other hand, in the dynamic replenishment case, each player in the competitive environment can access the entire action space and experience different replenishment actions. We formulate both cases in a way that is suitable for the MARL world and construct the environment-agent mechanism.

We use these formulations to describe the dynamics of the system such as state and actions, demand allocation, and reward structure. In addition, for both cases, we envision a world where pharmacies in the same competitive environment control their own inventory systems. In other words, considering the importance of the perishable inventory control problem in the pharmaceutical industry, in our thesis, we create an environment where two pharmacies compete against each other and manage their inventory systems in the face of a common demand. We train agents in the competitive environment with the algorithms we developed using the Nash Q-Learning technique in the MARL context for both fixed and dynamic replenishment cases. Trained agents learn to take the best replenishment action in the relevant state and period with self-play as learning progresses. We prepare an experimental design for the performance evaluation of the algorithms.

In this experimental design, agents trained with different parameters are tested in a game scenario and the results are evaluated and reported. In order to analyze the performance of the Nash Q-Learning algorithm in more detail, DP-based algorithms are implemented as benchmarks. In both cases, it is observed whether the Nash Q-Learning algorithm converges to the equilibrium solutions obtained with the developed DP-based algorithms.

As a result of the tests, it is observed that the Nash Q-Learning algorithm implemented in both the fixed replenishment and dynamic replenishment cases converges to the DP-based algorithms that obtain equilibrium solutions in a considerably shorter time than the runtime of the DP-based algorithms. In addition, in the fixed replenishment case, it is reported that the players trained with the Nash Q-Learning algorithm get more win-rates and establish dominance compared to the other players that use fixed replenishment in the competitive environment. In the dynamic replenishment case, although the players trained with the same parameters show similar performances and replenishment policies, it is observed that the performances of the agents trained with different parameter sets vary.

The rest of the thesis proceeds as follows. In Chapter 2, we review the perishable inventory control problem and approaches to its solution in the literature. Chapter 3 presents the definition, notation, and formulations of the perishable inventory control problem in a competitive environment, formulations of a single agent MDP and the N-player Markov game. We also introduce the Nash Q-Learning algorithms developed for fixed and dynamic replenishment cases in Chapter 3. The computational experiments and results of the Nash Q-Learning algorithms are analyzed and reported in Chapter 4. Chapter 5 includes a summary of the thesis work, future work, and the conclusion.

2. LITERATURE REVIEW

The perishable inventory control problem and the algorithms developed to solve this problem have been the subject of many studies in the literature so far. In this chapter of the thesis, the research that contributed or guided the work done in this thesis is discussed. In Section 2.1, previous studies that analyze and formulate a problem of perishable inventory control are reviewed. In Section 2.2 the studies and analysis of Reinforcement Learning and its use in sequential decision-making problems are reviewed. Then in the last section 2.3, papers that examine the relationship between multi-agent reinforcement learning and game theory, algorithms, and implementations developed in this context are examined.

2.1. Perishable Inventory Control

Various studies have been previously carried out in the literature on the management of perishable inventory systems. The perishable inventory control problem is studied by Pierskalla et al. [3] to find the optimal issuing policy considering the blood bank example. When they analyze the various optimal issuing policies in cases with different demand issuance rules, they find the optimal result as the First in First out (FIFO) issuing policy in all cases such as backlogging or lost sales demand. In addition to issuance policies, the transition definitions of the perishable inventory state, which contributed to the formulation of this thesis, are also proposed in this study.

The perishable inventory system consisting of exact two-period lifetime products is analyzed by Nahmias and Pierskalla [9] to find the optimal ordering policy under stochastic incoming demand. They formulate the structure of the two-period n -stage dynamic perishable inventory control problem by proposing dynamic cost and inventory state transition functions. Moreover, Fries [10] studies the classical single-item and multi-period perishable inventory control model to generalize the product lifetime to the fixed l periods.

Fries proposes a dynamic programming (DP) formulation of the perishable inventory control problem to achieve an optimal ordering quantity in each period under independent and identically distributed demand.

Nahmias [2] reviews the optimal issuing and ordering policies existing in the literature in different cases of inventory systems. By analyzing inventory systems with different parameters such as product lifetime type (e.g. fixed), demand uncertainty, and product variety in the inventory, the effectiveness of optimal issuing and ordering policies according to the systems is examined based on studies in the literature. Similarly to Pierskalla's (1972) result, Nahmias shows that in perishable inventory systems with fixed product lifetime and stochastic demand, the optimal issuance policy is FIFO, which is the issuing policy that will be used in this thesis. Relevant work has been done by Parlar et al. [11] to analyze and compare the long-run net average profit of FIFO and LIFO policies in a perishable inventory system with Poisson arrivals and deterministic product lifetimes, considering various costs and revenues to determine optimal input rates and policy performance under different parameters.

Optimal ordering policies in perishables are studied by Tekin et al. [12]. They propose an age-based control policy that considers not only the current inventory level but also the remaining lifetime of products while making replenishment decisions. To be more detailed, their policy is an ordering policy (Q, r, T) that performs an inventory replenishment in quantity Q when the inventory level drops to r or when T units of time have elapsed since the last time the inventory level reached Q . As a result of the tests they performed, the proposed age-based policy outperformed the well-known (Q, r) policy in all different cases. Another relevant work has been done on the management of perishable inventories with substitution by Deniz et al. [13] to create heuristics in different environments. The perishable inventory control problem is analyzed by allowing substitution, and the effects of different issuance rules and replenishment policies on the economic efficiency and predictability of the system are investigated in this extended problem.

The formulation of perishable inventory control models with a DP framework has been the predominant approach so far, providing structured frameworks and optimal solutions for decision-making; however, these methods often struggle with scalability, computational complexity, and adaptation to changing real-world circumstances. As a result, robust approximation algorithms are also investigated. Classic periodic-review stochastic inventory systems are studied by X. Chao et al. [4] to develop approximation algorithms with worst-case performance guarantees for periodic-review perishable inventory systems with general product lifetime. Their aim is to find the near-optimal policy on a perishable inventory system where demand might be non-stationary and correlated over a horizon. They develop two effective near optimal heuristic policies, which are Dual-Balancing (DB) and Proportional-Balancing (PB), to overcome this complex and computationally challenging problem by using nested marginal cost accounting scheme for perishable inventory systems and worst-case performance guarantees the near optimality with both policies. The worst-case performance error bounds are almost 100% but their computational experiments show that for the DB policy are around 0.3% above the optimal cost and the worst-case performance error for the PB policy is around 3% above the optimal and the average performance error does not exceed 1%.

2.2. Reinforcement Learning Approach

As in the perishable inventory control problem, the DP framework and approximation functions are used to formulate and solve the complex sequential decision-making problems in the literature since it offers a methodical and structured approach in multi-period settings. However, recursive functions in DP might be extremely costly in terms of memory and time due to the curse of dimensionality [4]. In order to overcome these limitations, Reinforcement Learning (RL), which uses solution approximation and aims to learn optimal policies through agent-environment interaction, has emerged. Instead of solving Markov Decision Process (MDP) iteratively, it enables the agent to learn optimal policies by its interaction with environment even without the need for model definitions (e.g. transition probabilities) explicitly.

Therefore, complex sequential-decision making problems can be solved by using approximation functions to overcome limitations such as the curse of dimensionality, which can cause computational burden.

Model-Free and Model-Based RL are two broad categories of RL algorithms, differentiated by how the agent-environment interaction is formulated. Model-free RL does not require explicit model to find optimal policies since the agent learns optimal policies through its experiences with the environment, whereas model definitions are strictly required to find the optimal decision set in model-based RL. In addition to the model-free RL, through agent-environment interaction, as learning progresses, the agent learns to take the right action in the relevant state, which leads to the optimal policy path. The well-known model-free RL algorithms are Temporal Difference (TD) learning and Q-Learning which are applied wide-range of areas to solve complex problems. TD-Learning, is developed and generalized by Sutton [14] to create a recency-weighting updating scheme in the context of reinforcement learning. In addition, by integrating observed rewards with bootstrapping projections of future values, TD-Learning gradually changes the value estimates. On the other hand, the Q-Learning algorithm was developed by Watkins et al. [5] using Bellman optimality equations (BOE) to create approximate dynamic solutions to the *NP*-hard problem. Q-learning finds the optimal action-value pairs with Q-value updates through the reward function. Model-free algorithms, in particular Q-Learning, contributed to the solution approach in this thesis in the multi-agent setting.

Recent advances and applications in RL are analyzed by A. Gosavi [15] to cover the mathematical background of this concept and provide the basis knowledge for researchers to use it in their own research. In this study, the connection between RL and MDP, the main idea of this structure, the algorithms developed on it, and the fields and applications where these algorithms are used are mentioned. Moreover, starting from the definition of DP and the transition to the RL is completely given in Bertsekas' RL book [16]. Both model-free and model-based algorithms, their complexities, and applications are included in this book.

The information in this book, which describes and illustrates in detail the formulation of RL algorithms, the DP-ADP relation, and algorithm convergence, has also helped in the progress of this thesis. Sutton and Barto [17] have written a relevant book to explain and introduce RL. It covers a wide range of RL studies, from how RL emerged out of necessity, the deep mathematical procedures behind it, its connections with MDP and Markov Games, the effective RL algorithms developed, and the generic implementations and case studies in which they are applied.

There are many RL implementations in the literature. For example, TD learning was applied by G. Tesauro [7] to create a neural network (NN) that trains itself to be an evaluation function for the backgammon game by playing against itself and learning from the outcome. In other words, by using Temporal Difference, the agent reaches a level where it can compete with professional players in a multi-user environment by providing continuous input and dynamic reward equations. The flow of Tesauro's study is as follows: The learning agent observes the value of the dice and the current state of the backgammon during the game, takes action by making a decision, and an output is produced. With this output, a positive or negative reward returns from the environment. This process continues iteratively, and the agent begins to take actions that will get the highest (depending on the problem) reward in each state. Therefore, by using temporal difference credit assignment, the agent ensures that the decisions made take the most effective action by anticipating the rewards that may come in both the previous and the next states. This study can be described as generalizable competitive environment policy learning rather than just a learning model with defined rules, and similar techniques are used in this thesis to simulate a perishable inventory control problem in a competitive environment. On the other hand, there are also studies in the literature that try to model the inventory control problem with Deep Reinforcement Learning (DRL), which is a combination of NN and RL, and this structure has also produced very successful results. For example, the DRL framework, called Deep Controlled Learning, is proposed by Temizoz et al. [18] to control perishable inventory systems with random lead times, with the result of an optimality gap of less than 0.1%.

In this study, periodic-review single agent perishable inventory control has been analyzed with different environment parameters and contributed to both the environment structure and demand issuance parts of this thesis.

2.3. Multiagent Reinforcement Learning & Game Theory

In the perishable inventory control problem, a learning mechanism can be established by considering the relevant inventory as a single agent (e.g. Tesauro's backgammon study) and preparing an environment that will iteratively interact with this agent. However, the problem becomes more complex when more than one agent is in the same environment and interacts with each other, and this is called Multi-Agent Reinforcement Learning (MARL) in the literature. As each agent in the environment cooperates or competes possibly with each other depending on the game definition, MARL is related to the Game Theory (GT). The Markov game was originally proposed by Littman [19] to generalize MDPs to multiple agents that interact within a common environment and with each other. In addition, a study examining the activity in the MARL world and the algorithms developed was conducted by Shoham et al. [20] for informational analysis purposes. Chang et al. [21] also examined role distinctions in the game by examining the learning structure and converged policies created by the agents in the environment. In this study, they show that agents respond to each other and create an equilibrium point when making decisions.

MARL formulations for solving major problems such as bullwhip effect in the supply chain world were investigated by Tongeren et al. [22]. They designed a system in which each participant in a competitive supply chain (e.g. retailer, wholesaler) is defined as an agent and the sequential individual decisions they make in the joint environment affect the problem objectives (e.g. total profitability). In other words, although the actions taken by agents in each state are independent of each other's actions, the outcome of these individual actions affects a common objective. In this study, which can be considered as a simplified example of the MARL world, they showed that agents learn policies better than humans.

However, they also showed that their study does not guarantee optimal policy convergence, especially since agents ignore each other's actions when taking their own actions. As a successor to this study, after analyzing the simplest form of MARL, the independent multi-agent world, Lanctot et al. [23] showed that this structure is insufficient to explain MARL. The reason for this is that the policies learned by each agent, which are independent of each other, start to overfit and fail to respond to each other. For this reason, they designed a new metric, called joint-policy correlation, to model MARL by creating policy and decision links between agents.

Moreover, Nowe et al. [24] studied the relationship between MARL and GT and showed that multiple agents interacting with each other require more than just modeling with MDP. In addition, unlike the single-agent RL, each agent considers not only its own state when taking actions as in the single-agent, but also the joint state in which other players are involved and the actions that other players can take.

Along with this work strengthening the link between MARL and GT, extensions have also been developed for well-known algorithms in the basic RL world, such as Q-Learning. Hu et al. [25] introduced a novel Q-Learning technique and demonstrated that, under certain strong assumptions, the solution converges to the Nash equilibrium (NE). With this study, a game-theoretic simulation environment is created in which each agent in the same environment evaluates the actions it will take depending on the state and policies of other players. Moreover, convergence concerns of MARL are also studied by Bowling [26] to show that every stage game has an NE and that the algorithm converges provably to that equilibrium. However, there are issues about the agents' selection of NE in case of multiple equilibria. Harsanyi et al. [27] introduce logical standards for determining which uniformly perfect equilibrium point should be used as the outcome of any competitive game.

In extension of their theoretic MARL study [25] and the Q-Learning algorithm [5], Hu et al. [8] introduced the Nash Q-Learning (NQL) algorithm to generalize the NE solution for general-sum stochastic games in a competitive multi-agent context.

A learning agent makes updates based on the assumption of NE behavior over the current Q-values and maintains Q-functions over joint actions. In addition, a period-related NashQ value is also included in the Q-function during Q-table updates, in order for NE to take into account the actions taken by the agents in the environment according to each other's state and policies over time. They experimented with the NQL algorithm in two-player grid games and proved convergence in the NE. In two different systems with deterministic and stochastic movements in a two-player grid world game, they have shown that both players can reach the target point in the grid with the equilibria formed over time using the NQL algorithm they created. This shows that the NQL algorithm can be implemented to handle multi-agent problems in a competitive environment.

Within the scope of the literature review, the studies, formulations, optimal issuing and ordering policies under the context of the perishable inventory control problem in Section 2.1 are used in this thesis in the problem formulation both in the definitions such as state, action space and in the definitions of functions [4] such as state transition, demand issuance [3], and reward. The research papers [15], sample implementations [18] and algorithms [5] in the literature on RL in section 2.2 are also used to design the solution environment and algorithm for the problem formulated in this thesis. In particular, Tesauro's TD-Gammon [6] (In the environment in this thesis, agents take actions simultaneously, whereas in Tesauro's study, although there is a turn-based mechanism) and Tongeren et al. [22] competitive supply chain study played an important role in designing the competitive environment and game dynamics in this thesis. Finally, algorithms in the MARL world, convergence analysis [26], game-theoretic approaches, stage-based equilibrium action interpretations [28], and in particular the NQL algorithm developed by Hu et al. [8] in Section 2.3 are used to implement the perishable inventory control problem in the competitive environment formulated in this thesis. Each of the studies in the literature review and shared in this chapter has helped the thesis either directly in the solution phase or in the post-solution analysis phase.

To the best of our knowledge, no research has been done that looks at solving perishable inventory control problems in a competitive environment using RL techniques. With this respect, combining the perishable inventory management and MARL contexts may be a significant step to find an optimal replenishment policy solution in a competitive environment rather than enumerating all solutions or implementing DP algorithms.



3. PROBLEM FORMULATION

In this chapter of the thesis, we present the formulation and the algorithms implemented to solve the perishable inventory control problem in a competitive environment. The definition of the problem and the motivation is explained in Section 3.1. The state and action notations of the agents in the environment are defined in Section 3.2. The formulations of the mechanisms in the environment that describe the process, such as demand allocation, state transition, and reward structure, are shown in Section 3.3, Section 3.4 and Section 3.5, respectively. In Section 3.6, we show the exact solution formulation of the single agent perishable inventory control problem and the solution approach using the Q value approximation. Finally, in Section 3.7, we present the equilibrium solutions (e.g. Stage and Game equilibrium) for the perishable inventory control problem in the multi-agent world, our formulation with Q-function approximation and the algorithms we implement with this Q-function formulation.

3.1. Problem Definition

The classical world of single-agent Reinforcement Learning (RL) consists of an environment and a single agent, as analyzed in studies [15, 16]. The learning and interaction mechanism in single agent RL is shown in Figure 3.1. At the beginning of the period t , the agent performs a replenishment action X_t (selected from the action space A) based on its current state, and as a result of this decision, the environment returns to the agent at the end of the period t with its immediate reward (e.g. C_t as negative cost as reward) and its state for the next period $t + 1$. This process continues on the planning horizon and for a given number of epochs (iteration).

As the epochs progress, the agent learns which action to take on the basis of the current state and period given as input. Using this communication mechanism of RL, efficient solutions to complex problems have been produced using algorithms such as Q-learning [5] and temporal difference (TD) learning [14].

In particular, in Tesauro’s TD-Gammon study [7], the single agent who has sufficiently completed Q-function updates with self-play has reached a level to compete with world-class players. In the Multi-Agent Reinforcement Learning (MARL) world,

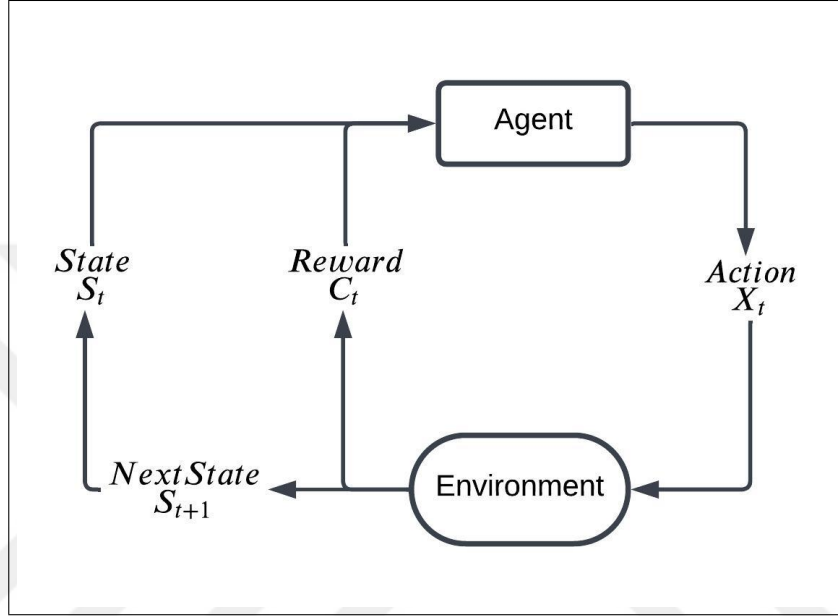


Figure 3.1. Single-agent reinforcement learning diagram.

a single environment consists of multiple agents competing against each other or acting together, as demonstrated in Figure 3.2. In addition to single-agent RL, agents determine the actions they take at the beginning of a period based not only on their current state and the period, but also on the state of other players and the actions they may take in the action space. Similarly to the single-agent mechanism, environment returns a specific reward to each agent and the next state information. In MARL [25], since the decisions taken by players in each period t are based on the joint inventory state $\bar{S}_t$ in that period and take into account the actions that other players may take, the actions of agents X_t^j are not independent of each other [23].

In a competitive environment, since the goal of the game is to win by outperforming other players, the decisions taken in the relevant state should take into account the decisions of other players. Each player learns to respond to each other as the game progresses, and this reveals the game-theoretic side of MARL [24].

The players take actions based on the actions they and other players can take, and the actions taken collectively form the equilibrium solution [29]. The main reason for the equilibrium is that each player has no better action to choose from in the action space given the actions of other players. The two-player Grid-World game [8] is one of the case studies demonstrating the implementation and efficiency of this theory. By combining Nash Equilibrium (NE) and Q-Learning (Nash-Q Learning), the resulting algorithm allows each player to find the optimal policy in a non-cooperative environment [8]. In this thesis, the interaction between MARL and Game Theory

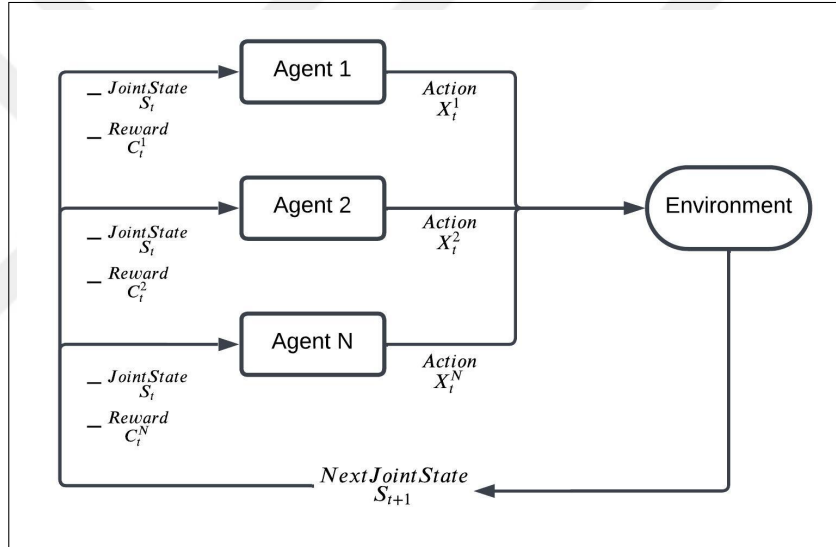


Figure 3.2. Multi-agent reinforcement learning diagram.

(GT) is used to efficiently solve the problem of perishable inventory planning in a competitive environment. In RL planning in the multi-agent world, each player (e.g. pharmacy) controls its own inventory of perishable goods and each pharmacy maintains the specified process within each period t (e.g. day) during the horizon length T . At the beginning of the day, each pharmacy performs its actions (inventory replenishment) after observing the joint state $\bar{S}_t$. With replenishment, inventories are replenished with new products (lifetime with m periods), and inventories are updated immediately before demand arrives. The incoming common demand is assigned to a pharmacy according to a rule, and the selected pharmacy must meet the incoming customer demand with the products in its inventory.

After the fulfillment of the demand is complete, the reward values (cost function) for each pharmacy are calculated and assigned before moving to the next day. Then, with the state transition functions, the pharmacy inventory states for the new day are determined and the game proceeds in this way. At the end of the game (when the planning horizon T is complete), the pharmacy with the minimum cost wins the game. Similarly to the two-player grid world game [8], in this simulation, each pharmacy is continuously updating its replenishment policies to achieve a lower cost than the other pharmacies and win the game. Replacement decisions made by pharmacies competing with each other to satisfy common demand and minimize individual costs also converge to the NE. Finally, this thesis analyzes the dominant replenishment policy by developing RL algorithms under the perishable inventory planning problem.

3.2. States and Actions

In this section of the thesis, the mathematical formulation of the problem of perishable inventory control in a competitive environment with a planning horizon of T (possibly infinite) periods, indexed by $t = 1, 2, \dots, T$ is introduced. Perishable inventory control in competitive environment consists of players aiming to satisfy incoming demand by replenishing their inventory. The players are indexed by j where $j = 1, 2, \dots, N$ and each player's inventory consists of a single product with m periods of lifetime, and at the end of its lifetime, the product is discarded. As the lifetime or number of products currently in inventory decreases (e.g. in case of demand fulfillment) or replenishment, the inventory state of each player changes accordingly. These inventory states exist in a state space S that contains all possible inventory states. At the beginning of period t , an inventory matrix [4]

$$S_t^j = (S_{t,1}^j, S_{t,2}^j, \dots, S_{t,m}^j) \quad (3.1)$$

is observed, where $S_{t,i}^j$ is the quantity of on-hand whose remaining lifetime is i periods for player j at the beginning of period t . Thus, S_t^j is the inventory state for the player j at the beginning of the period t . Let S_t^{-j} be the joint inventory state of all players except player j , $S_t^{-j} \equiv (S_{t,i}^1, S_{t,i}^2, \dots, S_{t,i}^{j-1}, S_{t,i}^{j+1}, \dots, S_{t,i}^N)$. Moreover, $\bar{S}_t$ is the joint inventory state of all players $j \in N$ at the beginning of period t .

The inventory states of the players change depending on the replenishment actions after observing $\bar{S}_t$ and before demand is realized in each period. At the end of the period, the players move to a new state $\bar{S}_{t+1} \in S$. The order of events in the competitive environment is demonstrated in Figure 3.3. Each player observes its own state and the

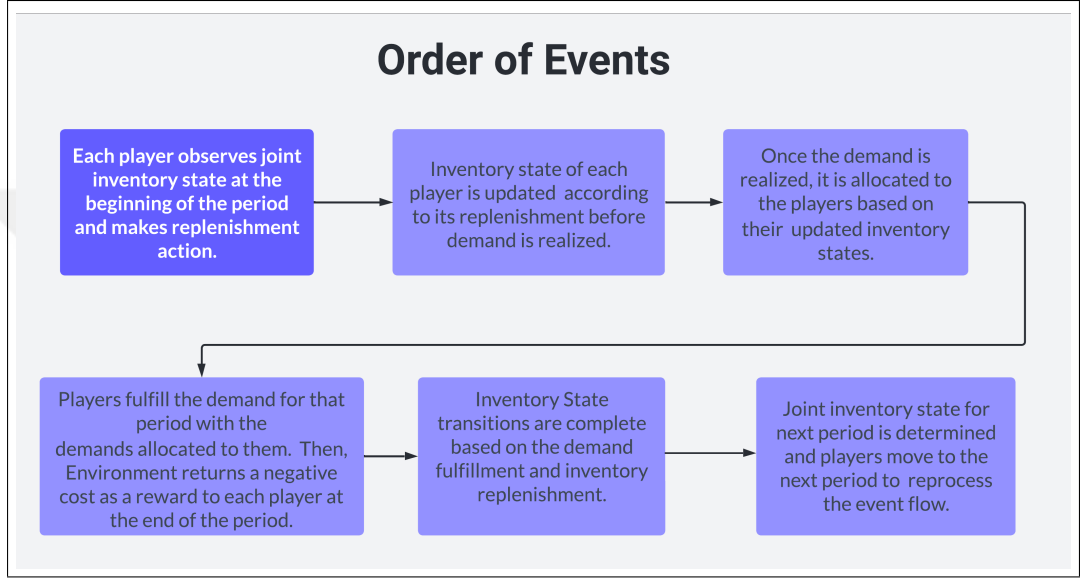


Figure 3.3. Order of events.

states of its competitors, and then takes an action at the beginning of each period to replenish its inventory with new products (m periods of life). $X_t^j \in A$ (where A is the action space) defines the quantity ordered by the player j at the beginning of the period t . In the competitive environment, a player's actions depend on its own state S_t^j and other players' states S_t^{-j} at period t . Thus, the actions of the players X_t^j depend on the joint inventory state $\bar{S}_t$ at the beginning of the period t . Players consider the joint inventory state and potential actions of others when making decisions. Thus, $X_t^j(\bar{S}_t)$ is the quantity ordered by the player j at the beginning of the period t given the joint inventory state $\bar{S}_t$. Let y_t^j be the inventory level of the player j in the period t . When player j orders X_t^j units of item at the beginning of period t , the inventory level equation of each player is defined as

$$y_t^j(y_t^{-j}) = \sum_{i=1}^m S_{t,i}^j + X_t^j(\bar{S}_t), \quad (3.2)$$

where y_t^{-j} is defined analogously for player $-j$.

Inventory level updates are handled at the beginning of each period due to the assumption of lead time being zero. Since the replenishment action taken by the player j depends on the actions of other players $-j$, the inventory level of that player j at the beginning of the period t is a function that depends on the inventory levels of other players $-j$ (3.2).

3.3. Demand Structure

Demand over the planning horizon is a random variable, denoted by $D_1, \dots, D_T$. D_t is the common demand in the competitive environment in the period t and is allocated to each player based on their inventory states. In other words, based on the inventory states of the players at the beginning of the period, a player-based deterministic metric called the fulfillment metric is calculated. The allocation of the demand in that period to the players is determined according to this fulfillment metric. The fulfillment metric, which will be explained in detail later in this section, is a numeric metric that demonstrates the quality of players' inventories based on its quantity and age distribution, and according to this metric, the inventory selection of the incoming demand is determined and is assigned to one of the players. Once the demand allocation is completed, the players fulfill the demand for that period with the demands allocated to them.

It is useful to explain the demand allocation notation with a numerical example. Suppose that there is a two-pharmacy competitive environment and each pharmacy has a single product with lifetime $m = 2$ in their inventory. This implies that each pharmacy has a two-dimensional inventory state matrix S_t^j . The indexes in the inventory state matrix show the remaining lifetime of the product in the relevant inventory, and each index shows how many products with that index lifetime are in the inventory (3.1). Assume that at the beginning of period t , the inventory states of two players (e.g. S_t^1 and S_t^2) are $[2, 0]$ and $[3, 0]$, respectively. This shows that the first pharmacy has two units of a product with a remaining lifetime of a period, while the second pharmacy has three units of a product with a remaining lifetime of a period.

As it is beginning at the period t , neither pharmacy has any new products (e.g, $i = m$) in its inventory. At the beginning of the period, both pharmacies perform their replenishment X_t^1 and X_t^2 , and their inventory states are updated immediately since the lead time being zero. Assume that the inventory replenishment of pharmacies 1 and 2 X_t^1, X_t^2 is 3 and 2, respectively. Then, the corresponding updated inventory states S_t^1, S_t^2 are $[2, 3]$ and $[3, 2]$, respectively. After the inventory replenishment of pharmacies is completed, fulfillment calculation is performed to determine demand allocation. This calculation involves a series of mathematical operations.

First, the inventory level y_t^j of each pharmacy j indicates the total quantity of the product in its inventory at the beginning of the period t (3.2). For this example, the inventory levels of both pharmacies are calculated as 5 ($y_t^1 = 5, y_t^2 = 5$). Second, for each pharmacy, the inventory value v_j is calculated using the inventory states of the pharmacies. This value is a metric calculated based on the age distribution of products in pharmacy inventory and is positively correlated with the quantity of newer products. Inventory value is calculated as

$$v_j = \sum_{i=1}^m i \cdot S_{t,i}^j. \quad (3.3)$$

The output of this inventory value function serves as a metric showing the quality of the pharmacy's inventory in period t and the higher the better. As a result of the function (3.3), the inventory values of pharmacies v_1, v_2 are calculated as 8 and 7 respectively.

In the third stage, a value called the normalized inventory difference (NID_j) is created for each pharmacy using the difference between its own inventory level y_t^j and the maximum inventory level $y_t^{\max}$ in period t . This value is a metric that compares the inventory level of the pharmacies with the inventory levels of other pharmacies in a competitive environment. In other words, we define a metric that measures whether the inventory levels of pharmacies are excessive compared to each other and that we can use in the demand assignee selection calculation.

Calculation of normalized inventory difference is as

$$NID_j = \begin{cases} 0, & \text{if } \max(y_t^1, y_t^2, \dots, y_t^N) = 0, \\ \frac{g(y_t^j - y_t^{\max}) \cdot |y_t^j - y_t^{\max}|}{y_t^{\max}}, & \text{otherwise.} \end{cases} \quad (3.4)$$

where $y_t^{\max} = \max(y_t^1, y_t^2, \dots, y_t^N)$ and auxiliary function

$$g(x) = \begin{cases} -1, & \text{if } x < 0, \\ 0, & \text{if } x = 0, \\ 1, & \text{if } x > 0. \end{cases} \quad (3.5)$$

which is used to control the signs of inventory levels while calculating the normalized inventory difference for each player (3.4). Note that when calculating NID_j , the condition $y_t^j - y_t^{\max} \leq 0$ is inherently satisfied since $y_t^{\max}$ is defined as the maximum value among all y_t^j . Since both pharmacies have the same inventory level, NID_1 and NID_2 are zero in our example.

Subsequently, the fulfillment metric F_j is calculated for each pharmacy with the linear function that calculates it using NID_j and v_j . Since the pharmacy to which the demand will be assigned is determined by this metric, the calculation of the fulfillment metric includes both the total inventory level of the pharmacies and the age distribution of the products in this inventory. This fulfillment metric shows both the inventory level and the quality of the age distribution of the products of pharmacies. Using inventory value v_j (3.3) and NID_j (3.4), we define the fulfillment metric F_j for each player j as

$$F_j = w_n \cdot NID_j + w_v \cdot (v_j - v_{\max}), \quad (3.6)$$

where $v_{\max} = \max(v_1, v_2, \dots, v_N)$, w_n and w_v denote the weights of NID and inventory value gap, respectively. Note that the weights w_n and w_v are common for all pharmacies and are used to prioritize the inventory level or age distribution while calculating the fulfillment metric of each pharmacy. In our thesis, since we assume that the priority of the inventory level and the age distribution are equal, we assign their weights w_n, w_v as one. However, when it is desired to be prioritized in the algorithm, the fulfillment metric setting is done by changing the weights. Furthermore, note that when calculating F_j , the condition $v_j - v_{\max} \leq 0$ is satisfied.

With this function, a single fulfillment metric is calculated for each pharmacy, which includes both inventory level-related (NID_j) and age distribution-related (v_j) values for the pharmacies' inventories. Using the fulfillment metric function (3.6) in the example, F_1 and F_2 are calculated as 0 and -1, respectively.

Finally, a maximization is performed between the fulfillment metrics of the pharmacies in the competitive environment, which determines the pharmacy to which the demand is allocated. Selection of the demand assignee is as

$$k = \arg \max_j F_j. \quad (3.7)$$

In cases where some/all players' fulfillment metrics are equal from equation (3.6), $F_j = F_{-j}$, the player to whom the demand will be allocated is randomly selected. In our example, since $F_1 > F_2$, the incoming demand is fully allocated to pharmacy 1.

At the beginning of each period, after replenishment decisions are made according to player policies, demand arrives and is assigned to a player with the maximum fulfillment metric value (3.7). While all incoming demand is allocated to the selected player k , the remaining players are assigned zero demand and the amount of incoming demand is written as lost sales. Demand allocation is defined as

$$D_t^j = \begin{cases} D_t, & \text{if } j = k, \\ 0, & \text{otherwise.} \end{cases} \quad (3.8)$$

The total quantity of demand is assigned to the selected player k and nothing to the rest of the players proposed in the equation (3.8). Then, the player k might satisfy the partial/full incoming demand, whereas the rest of the players are incurred with the lost sales. Note that D_t^j is a function of the joint state $\bar{S}_t$ due to the allocation rule, and we use the notation $D_t^j(\bar{S}_t)$ to emphasize this dependency where necessary. In our example, since pharmacy 1 is selected as the demand assignee in period t , all incoming demand in that period is allocated to it, while the demand allocated to pharmacy 2 is zero (3.8). Then D_t^1 is D_t , while D_t^2 is 0. If player k cannot satisfy all of the allocated demand, the remaining demand is lost even if there is enough inventory in other players.

3.4. State Transition

We present the transition functions of the players' inventory states when moving to the next period. During period t , inventory states are affected by replenishment actions X_t^j taken by players and demand allocation D_t^j . In addition, the inventory states are updated (3.2) immediately with the replenishment made at the beginning of the period t , and the demand allocation (3.8) is completed according to these updated player inventory states. Then, the players meet the demand assigned to them. Based on the allocation of demand in the current period t and the aging of the products in the inventory, the inventory state is determined for the next period $t + 1$.

The aging of the products in the inventory is discrete and the remaining lifetime of the products decreases by a period for each period transition. During the transition, this update is performed by changing the index of products in the inventory state matrix S_t^j . On the other hand, while the information on the structure and allocation of demand used in this thesis is explained in Section 3.3, this section also explains how the assigned demand is issued. In addition, according to this demand issuing policy, inventory state updates are performed during the period transition. In the literature, many studies have been carried out on the performance of the demand issue rule and its problem-related extensions. Especially in the perishable inventory management framework, First-in-first-out (FIFO) has been the dominant demand issuance rule [3, 13, 30]. We also adopt the FIFO rule and satisfy the demand by using the oldest products (with the least retention lifetime) in the inventory, and continuing with newer products. In cases where the incoming demand cannot be fully met, partial issuance is allowed, and the quantity of unmet demand is considered as lost sales for the selected player.

After completion of the period t , the system proceeds to the subsequent period $t + 1$. By the definition of the inventory state S_t^j , we obtain the following inventory

state transition from $S_{t,i}^j$ to $S_{t+1,i}^j$

$$\begin{aligned} S_{t+1,i}^j &= (S_{t,i+1}^j - (D_t^j(\bar{S}_t) - \sum_{n=1}^i S_{t,n}^j)^+)^+, \quad \text{for } 1 \leq i \leq m-1, \\ S_{t+1,m}^j &= (X_t^j(\bar{S}_t) - (D_t^j(\bar{S}_t) - \sum_{n=1}^{m-1} S_{t,n}^j)^+)^+, \end{aligned} \quad (3.9)$$

where $(x)^+$ is a standard notation for $\max\{x, 0\}$.

The inventory state transitions of the players are performed according to equation (3.9) and current state S_t^j , action X_t^j and demand $D_t^j(\bar{S}_t)$ information are used in these transition functions to calculate the next state S_{t+1}^j .

3.5. Cost Structure

We assume that each player has identical cost parameters, which are also assumed to be constant throughout the planning horizon although our algorithms can work without these assumptions. In each period t , four types of costs may occur: a unit ordering cost c^o , a unit holding cost c^h for leftover inventory, a unit shortage cost c^s for unsatisfied demand and a unit deterioration cost c^d for expired (perishing) products. Denote y_t^j as the total inventory level of the player j after receiving the order at the beginning of period t (3.2). When player j makes a replenishment, the ordering cost $c^o X_t^j$ is incurred. Since the lead time is assumed to be zero, the receiving order at the beginning of the period immediately updates the total inventory level (3.2) of the corresponding player.

At the end of period t , if $y_t^j - D_t^j \geq 0$, then excess inventory incurs a holding cost $c^h(y_t^j - D_t^j)^+$. On the other hand, if $y_t^j - D_t^j \leq 0$ for player $j = k$, then the system incurs a shortage (lost sales) cost $c^s(D_t^j - y_t^j)^+$, and $c^s D_t^j$ for players $-j$ to induce an incentive to all players to attract demand. Lastly, if the inventory with one period remaining life $S_{t,1}^j \geq D_t^j$, then $(S_{t,1}^j - D_t^j)^+$ units perish and incur a deterioration cost $c^d(S_{t,1}^j - D_t^j)^+$. At the end of period t , a reward is determined by the environment for all players.

Reward is the value returned by the total cost function. Single-period cost is a function of player j 's state, action and demand as

$$\begin{aligned} C(S_t^j, X_t^j(\bar{S}_t), D_t^j) = & c^o X_t^j(\bar{S}_t) + c^h (y_t^j(y_t^{-j}) - D_t^j)^+ \\ & + c^s (D_t^j - y_t^j(y_t^{-j}))^+ + c^d (S_{t,1}^j - D_t^j)^+, \end{aligned} \quad (3.10)$$

where cost is a function of state, replenishment action and the allocated demand of the player j at period t . At the end of each period, the environment returns the reward function value (3.10) to the players, which indicates the result of taking action X_t^j in state S_t^j .

3.6. Single Agent MDP Formulation

Up to this point in Chapter 3, formulations and definitions of system dynamics, such as demand allocation, cost (e.g., reward) structure, and notation, are shown. In this section, we show the formulations in the single-agent Markov Decision Process (MDP) world and then move on to the formulation in the multi-agent world. In addition, in this section, the exact solution formulations and the Q-learning approximation are detailed in Sections 3.6.1 and 3.6.2, respectively.

Single-agent RL is a system in which there is an agent and an environment, and a continuous action-reward interaction ensures that the agent takes the best action in the relevant state. Depending on the setup of the problem in the RL, the state transition probabilities can be included during learning, which is called model-based RL. On the other hand, model-free RL techniques are used to solve problems with unknown state transition probabilities.

3.6.1. Exact Solution Formulation

Perishable inventory control is a problem that requires sequential decision-making solutions, and the enumeration approach is not preferred due to the time and memory burden caused by the complexity of the problem. Therefore, researchers often prefer implementing DP to obtain the exact solution to this problem [3, 4].

Using Bellman equations in dynamic programming, decision-making problems are solved in a recursive manner. More specifically, in each period t , under given S_t , the problem is solved by setting the total cost expectation in that period and in successive periods. When we model the perishable inventory control problem with a single agent, the agent makes a replenishment decision X_t in state S_t in period t under stochastic incoming demand D_t , and as a result of this decision, after calculating the cost $C(S_t, X_t, D_t)$ in that period, it switches to the new state S_{t+1} and continues this recursion. Thus, given a cost expectation with all demand possibilities, the agent arrives at the replenishment policy π .

Definition 3.1. *A policy π is a strategy that the agent follows to decide its replenishment actions given the current state S_t . π is a mapping from state to action $\{\pi : S_t \rightarrow X_t\}_t^T$ where X_t is the action taken by agent at period t .*

$X_t(S_t)$ is based on the current state given, defined as $X_\pi(S_t)$. In other words, it represents the replenishment action of the agent at the beginning of the period t under policy π given the current state information of the environment. The exact solution formulation of single-agent MDP is as

$$V_\pi(S_0) = \underset{D_t}{E}_{t=0,1,\dots,T} \left[\sum_{t=1}^T C(S_t, X_\pi(S_t), D_t) \mid S_0 \right], \quad (3.11)$$

where D_t is the incoming stochastic demand at period t , S_t is the inventory state of the agent at period t and X_t is the replenishment action taken by the agent at period t . Equation (3.11) shows the dynamic programming formulation under policy π , using the Bellman equation.

On the other hand, MDP solution using Bellman equations is as

$$V_t^*(S_t) = \min_{X_t} \underset{D_t}{E}_{t=0,1,\dots,T-1} \left[\sum_{t=1}^T C(S_t, X_t, D_t) + V_{t+1}^*(S_{t+1}) \mid S_t \right]. \quad (3.12)$$

With this state-value function (3.12), the optimum replenishment action X_t^* that provides the minimum cost among the total expected costs $V_t(S_t)$ calculated for each period t and state S_t is reached.

In addition to state-value dynamic programming formulation, sequential decision-making problems such as perishable inventory control can also be formulated using action-value functions. In the exact dynamic programming formulation using the Q-function, the state-action pairs are iterated to find the optimal policy. Therefore, this formulation is made with the action-value function $Q_t(S_t, X_t)$. Similarly to the state-value functions, the action-value functions are created using the Bellman equations. Then, single-period action-value exact solution formulation is as

$$Q_t(S_t, X_t) = \underset{D_t}{E}_{t=0,1,\dots,T} [C(S_t, X_t, D_t) + Q_t(S_{t+1}, X_{t+1}) \mid S_t, X_t], \quad (3.13)$$

where $Q_t(S_t, X_t)$ is the Q-value which is the total expected cost when the agent takes replenishment action X_t in inventory state S_t and period t . $Q_t(S_t, X_t)$ contains the immediate cost and the optimal cost-to-go in state S_t for all actions X_t . Then, the optimal policy π^* is calculated by minimizing the Q-values in each period t . Calculation of the optimal replenishment policy is defined as

$$\pi^*(S_t) = \arg \min_{X_t} Q_t(S_t, X_t), \quad t = 1, \dots, T \quad (3.14)$$

where π^* is the optimal replenishment policy, and the best action can be taken in period t given inventory state S_t using that policy. In addition, given an initial inventory state, the optimal replenishment path is created recursively using π^* .

3.6.2. Q-Value Approximation

Although researchers often use dynamic programming to exactly solve the perishable inventory control problem, dynamic programming is not computationally efficient due to the recursive functions involved. In contrast, as the input space grows, the complexity of the problem and the cost of time increase exponentially. This technique, which provides exact solutions for small-size problems, is ineffective for large-size problems. Therefore, the approximation technique is an effective approach to this scaling problem. In other words, instead of making a cost expectation by iteratively computing all possible state-action pairs with all demand probabilities, a cost approximation is made to find the best action that can be taken in the relevant period and state.

The Q-value approximation is seen as an implementation of this mechanism [5,15]. The Q-value approximation is the computation of optimal action values obtained using Q-functions (3.13) and updating the corresponding indexes in the agent's Q-table. The purpose of the Q-function formulation is to approximate the reward returned by the action taken by the agent in the relevant state by updating the Q-value at index (S_t, X_t) in the agent's Q-table. The update mechanism is an approximation based on the observed Q-functions of rewards (e.g., cost) and future estimated costs. This learning framework is called Temporal Difference (TD) Learning [14]. The goal is that the Q values updated using TD converge after sufficient learning and that this convergence point is the optimal Q value. In other words, the approximated point converges to the optimal replenishment policies found in the state-value outputs. TD update is as

$$Q_t(S_t, X_t) = (1 - \alpha)Q_t(S_t, X_t) + \alpha \left[C(S_t, X_t, D_t) + \min_{X_t} Q_{t+1}(S_{t+1}, X_{t+1}) \right], \quad (3.15)$$

where α is the learning rate of the algorithm. In each iteration (e.g, epoch) during learning, the Q-function is updated using the TD equation (3.15). In addition, learning continues according to the maximum number of epochs, and as the epoch progresses, the Q-values converge to the optimal policies [26].

3.7. Multi-Agent Markov Game Formulation

Multi-agent Markov Game is an extension of single-agent MDP where multiple agents interact in the same environment. In single-agent MDP, a single agent performs inventory replenishment with an action it chooses from the action space according to its own state in the relevant period and a cost is generated accordingly. In a Markov game, on the other hand, n agents consider not only their own inventory state, but also the inventory states of all agents in the relevant period and try to determine the action as a response to the actions that other agents can take. Thus, the fact that each agent tries to find the best action relative to the other agents reveals the game-theoretic aspect of the Multi-agent Markov Game. In other words, the solution in the corresponding joint inventory state $\bar{S}$ and period t represents the Nash equilibrium [8].

The formulations used in this thesis are generic and are considered as N -agent, but the analyses and tests performed in Chapter 4 are specifically in the two-agent world. In this section, we show in Section 3.7.1 the formulation of the equilibrium solution of the problem, in Section 3.7.2 the formulation of the approximation of the perishable inventory control problem in the competitive environment, and then in Section 3.7.3 we present the algorithms developed using the approximation functions shown in Section 3.6.

3.7.1. Nash Equilibrium

The Perishable Inventory Control problem in a competitive environment is also set as a Markov game and the solution is reached through equilibrium points. In addition, there are two concepts of equilibrium in this problem. First, the replenishment decision pair for each agent in the relevant state and period is called the Stage Equilibrium, and the Markov game that starts from any initial state and returns the equilibrium replenishment policy till the end of the horizon for each agent is called the Game Equilibrium. Let us collect the state of the game at time t as $\bar{S}_t = (S_t^j, S_t^{-j})$ and define $\bar{S} = (\bar{S}_1, \bar{S}_2, \dots, \bar{S}_T)$.

Definition 3.2. A policy π^j of player j is a collection of functions $\{\pi^j : \bar{S} \rightarrow \Delta_t^j\}_t^T$ where Δ_t^j is a probability space over the action set A_t^j such that $\pi^j(X^j | \bar{S})$ denotes the probability of taking action X^j under (stationary) policy π^j .

The total expected cost $V_\pi^j(\bar{S}, \pi^j, \pi^{-j})$ for player j is given as

$$V_\pi^j(\bar{S}_t, \pi^j, \pi^{-j}) = E_{D_t^j} \left[\sum_{t=0}^T C(\bar{S}_t, \pi^j, D_t^j(\bar{S}_t)) \mid \pi^{-j}, \bar{S}_0 \right]. \quad (3.16)$$

Note that the expectation is over the D_t^j , π^j and π^{-j} .

Definition 3.3. The pair of solutions (π_*^j, π_*^{-j}) that satisfies

$$\begin{aligned} V_\pi^j(\bar{S}_t, \pi_*^{-j}, \pi_*^j) &\leq V_\pi^j(\bar{S}_t, \pi^j, \pi_*^{-j}) \quad \forall \pi^j, \\ V_\pi^{-j}(\bar{S}_t, \pi_*^j, \pi_*^{-j}) &\leq V_\pi^{-j}(\bar{S}_t, \pi^{-j}, \pi_*^j) \quad \forall \pi^{-j} \end{aligned}$$

is called a *Game Equilibrium of the Markov Game*.

The Game Equilibrium of the Markov game shows the equilibrium replenishment policy path of the players in a competitive environment. In other words, it consists of equilibrium solutions that contain the actions $X_t^j(\bar{S}_t)$ to be taken by each player for each period t from the initial state S_0 and the period to the length of the horizon T . Another equilibrium concept, Stage Equilibrium, defines the equilibrium solution in each period t and state $\bar{S}_t$.

Definition 3.4. *At every stage (period) t of the game, Q -function for the player j is defined as*

$$Q_t^j(S, X^j, X^{-j}) = E \left[\sum_{t'=t}^T C(\bar{S}_{t'}, X_{t'}^j, D_{t'}^j(S)) \mid S = \bar{S}, X^j = X_t^j, X^{-j} = X_t^{-j} \right], \quad (3.17)$$

where Q -function for each player depends on the joint state and the actions of all players.

Together with the expectation in the above definition, the Q -values for each player j in the period t are calculated. Then, using these Q -values, a Nash game is generated in the relevant period and state, and the solution of this Nash game is the Stage Equilibrium, which is the equilibrium solution in period t and state $\bar{S}_t$.

The following Algorithm 3.4 describes how to reach Stage and Game equilibrium solutions in the competitive environment. The example used in the algorithm assumes a two-player competitive world, but can also be adapted to the n -agent competitive environment. First, Q -table and state initializations are performed. Then, for each period t within the horizon length T , the Q -values of the players j in each state $\bar{S}$ are calculated, and the Stage equilibrium is obtained with the Nash game solution generated over these Q -values (3.17). This is the equilibrium solution that shows the actions X_t^j to be taken by the players in state $\bar{S}_t$. Then, with the equilibrium actions determined, the state transitions are handled (3.9), and the next state is entered. Stage equilibrium solutions calculated for each period and state are stored. Finally, Game equilibrium solutions π_*^j are obtained using Stage equilibrium solutions in period T .

Algorithm 1 Stage equilibrium and game equilibrium procedure.

1: **Initialization:**

- Set $Q_0^j(\bar{S}, X^j, X^{-j}) = 0$ and $Q_0^{-j}(\bar{S}, X^j, X^{-j}) = 0$ for all S, X^j, X^{-j}
- Initialize state $\bar{S}_0$

2: **for** each stage $t = 0, 1, \dots, T - 1$ **do**

3: **Step 1: Determine Stage Equilibrium Actions**

- Form a two-player Nash game using the current Q-functions

$$Q_t^j(\bar{S}_t, X_t^j, X_t^{-j}), \quad Q_t^{-j}(\bar{S}_t, X_t^j, X_t^{-j})$$

- Compute the Stage Equilibrium actions using Nash game formed by using Q-functions of players.

4: **Step 2: Update State and Q-Functions**

- Determine the next state $\bar{S}_{t+1}$ using the transition function (3.9).
- Update the Q-functions using (3.15).

5: **end for**

6: **Output Game Equilibrium Policies:**

- Derive the Markov Game equilibrium policies π_*^j and π_*^{-j}
-

Figure 3.4. Algorithm: stage equilibrium and game equilibrium procedure.

Using this algorithm, equilibrium solutions to the perishable inventory control problem in a competitive environment are obtained. The replenishment actions that players take for each state and period are obtained through Stage equilibrium, and the replenishment paths starting from any initial state and continuing until the last period are obtained through Game equilibrium. However, solving these two equilibria using expectation is exhaustive for large-scale problems. Therefore, it is necessary to approximate the equilibrium solutions to solve sequential-decision making problems such as perishable inventory control in a competitive environment.

3.7.2. Q-Value Approximation

Similar to the single-agent MDP problem, in the large-scale multi-agent world, finding equilibrium solution(s) becomes even more difficult to achieve due to the problem complexity. Therefore, it seems feasible to find solutions using approximation techniques.

In the multi-agent world, approximation can be done using state-value functions or Q-functions. Q-Learning [5] is a well-known approximation method in the RL world and is implemented in a wide range of applications. The primary factors that contribute to Q-Learning's widespread popularity and its usage as a tool for solving challenging problems are its efficiency, simplicity, and adaptability. The Q-learning equations can be applied, systematically analyzed, and extended (e.g., Nash-Q) according to different problems, which are also features of this method [25]. In the multi-agent world, multiple agents interact with each other in the same environment and take actions accordingly. Although it is possible to formulate the MARL problem using multiple single-agent Q-functions (3.14), it is not sufficient for the solution. The main reason for this is that these equations do not describe the interaction between the agents and the fact that they are in the same environment, which leads to an independent and non-stationary process [23]. Therefore, it becomes difficult to obtain a solution to the problem, and a suitable multi-agent formulation is required.

Although the Q-Learning formulation in MARL is similar to single-agent, the actions that agents might take into account not only their own states but also the states of other agents and the actions they can take. In other words, each agent converges to Nash Equilibrium as it will try to best respond to the actions that other agents can take within its own Q-tables. The Nash-Q Learning algorithm [8], which combines Q-Learning and Game Theory, also effectively reflects the MARL formulation. The players represent agents in a competitive environment, and each player j has its own state S_t^j and action X_t^j in period t . In addition, the joint states of the players are represented as $\bar{S}_t \equiv (S_t^1, S_t^2, \dots, S_t^N)$.

Each player j has its own Q-function, defined as $Q_t^j(\bar{S}_t, X_t^j)$ where $\bar{S}_t$ and X_t^j are the joint inventory state and the replenishment action of the player j in period t , respectively. Note that all agents observe all states before taking an action.

Multi-agent Q-function [8] is defined as

$$Q_t^j(\bar{S}_t, X_t^1, X_t^2, \dots, X_t^N) = C(\bar{S}_t, X_t^j(\bar{S}_t), D_t^j) + \sum_{S'} \min_{X_t^j} Q_t^j(S', X_t^1, X_t^2, \dots, X_t^N), \quad (3.18)$$

where $\bar{S}_t'$ is the next joint state, $C(\bar{S}_t, X_t^j(\bar{S}_t), D_t^j)$ is the cost (3.10) of the player j at the end of period t under joint states and actions.

Using this Q-function (3.18), the indexes in each player's Q-table are updated according to the equilibria obtained in the game. The update procedure is similar to Basic Q-Learning, but additionally takes into account the Nash equilibrium, state, and possible actions of other players. The multi-agent Q-table updating procedure is handled as

$$Q_t^j(\bar{S}_t, X_t^1, \dots, X_t^N) = (1 - \alpha)Q_t^j(\bar{S}_t, X_t^1, \dots, X_t^N) + \alpha \left[C(\bar{S}_t, X_t^j(\bar{S}_t), D_t^j) + \text{Nash}Q_t^j(\bar{S}_t') \right], \quad (3.19)$$

where $\text{Nash}Q_t^j$ values are calculated as

$$\text{Nash}Q_t^j(\bar{S}_t') = \pi^1(\bar{S}_t') \dots \pi^N(\bar{S}_t') \cdot Q_t^j(\bar{S}_t'). \quad (3.20)$$

As a hyper-parameter, the learning rate (α can also be a time-dependent (α_t) parameter. In this thesis we define a learning rate using exponential decay.

Using equation (3.19), the Q tables of each player are updated during the learning period using the $\text{Nash}Q$ values from equation (3.20). Equilibrium actions are determined for each player depending on the specific period t and the joint state $\bar{S}_t$, resulting in a policy function for each player j , defined as $\pi^j(\bar{S}_t)$. Using this policy set and the player's Q table, the $\text{Nash}Q$ value ($\text{Nash}Q_t^j$) of the player j in period t is found. Then $\text{Nash}Q$ values obtained for each player are used during the Q-Table update. Hu and Wellman show that under some strong assumptions, the Nash-Q algorithm converges to the game equilibrium [8].

3.7.3. Nash Q-Learning Algorithm

Multiple agents acting together or against each other in the same environment and trying to take the best action in each period and state are formulated in Section 3.7.1. With this formulation, the equilibria of Stage and Game are found and the solution of the problem is solved in two aspects. In addition, rather than making a cost expectation by considering all possible feasible solutions, the Q-value approximation in the previous section 3.7.2 allows faster and less costly solutions to this problem. In this section, we show the Nash Q-Learning algorithm, which is the implementation of the formulated Q-value approximation technique. Nash Q-Learning [8] is the algorithm that solves complex sequential decision-making problems in the MARL world using the approximation technique.

In this thesis, the Nash-Q Learning algorithm is applied to the perishable inventory control problem in a competitive environment. Thus, at the end of the algorithm, the problem is solved by obtaining replenishment policies based on the Q-functions obtained for each agent. All of the formulations and algorithms in this chapter can be generically applied to the n -agent world. However, in this Section, we propose algorithms in a smaller agent space (e.g., 2-agent). In other words, we show an algorithm that analyzes the best replenishment actions that two inventory agents in the same environment can take against each other.

We consider the perishable inventory control problem in a competitive environment from two different perspectives and develop algorithms accordingly. In the first Algorithm 3.5, one agent can experience different replenishment actions and update its Q-function accordingly, while the other agent can only replenish the average of the demand distribution (Note that we assume $\text{Poisson}(\lambda)$ as demand distribution in this thesis) in each period. Thus, more specifically, the applicability of Nash Q-Learning to the perishable inventory control problem is examined. The algorithm implements the environment of an agent that performs dynamic replenishment based on the experiences it updates with the algorithm against an agent that only replenishes by λ .

In the second Algorithm 3.6, a mechanism is organized where both agents can try different replenishment actions in the bounded action space. Thus, it can be analyzed how both agents respond to each other in a world where both agents can access different replenishment actions.

Both algorithms follow a similar approach, differing in where the agents' replenishment actions are determined and where the Q-functions are updated. In addition, another input parameter that affects the replenishment action is the strategy. In other words, the replenishment action selection strategy is determined according to the exploration strategy defined in the input of the algorithm. While the Greedy strategy tends to emphasize exploitation over experienced equilibria actions, the Epsilon-Greedy strategy tends to try new actions in the action space through exploration.

The flow for both algorithms is as follows: For each period t (e.g., day) in the horizon length, the joint inventory state $\bar{S}_t$ is observed. Using the observed joint inventory state and the relevant day, the replenishment action X_t^j of the players at the beginning of that day is determined together with the game generated from the Q-table. In other words, actions are determined according to the equilibrium points in this Nash game. After both agents make their replenishment at the beginning of the day, the stochastic demand D_t arrives. A fulfillment metric F_j is determined (3.6) according to the inventory state of the agents after replenishment. Based on this metric, the demand D_t^j to be allocated to the agents is determined (3.8). Afterwards, the environment returns the reward $C(\bar{S}_t, X_t^j(\bar{S}_t), D_t^j)$ and the next state $\bar{S}_{t+1}$ to both agents for the relevant period considering the current state, replenishment, and allocated demand (3.9). With a new Nash game created using the next state information, agents observe their NashQ values $\text{Nash}Q_t^j$ in the future (3.20). Finally, the Q-functions are updated (3.19) for each agent using the reward, Q tables, and NashQ values returned by the environment. This process is performed for each day t in horizon length T and is completed in an epoch when the days are completed. This learning is played with the E parameters provided in the input and the optimal replenishment policies are reached by the algorithm.

In the first Algorithm 3.5 based on fixed lambda replenishment, the flow described above is implemented. In this flow, the first agent selects any action in the action space according to the specified exploration strategy (e.g. epsilon-greedy), while the other agent performs as much as the demand average. In addition, the Q-tables used during the Q-function update are also used for the fixed replenishment of the second agent, since the second agent performs fixed replenishment of that act.

In the second Algorithm 3.6, where both agents can perform dynamic replenishment, the flow is similar and this time the second agent, like the first agent, searches for different replenishment in the action space and updates the Q-function in an unfiltered manner. Thus, both agents learn how to best respond to each other considering the whole action space. In addition, the second algorithm has two different learning rates as input. This allows each agent to start with different initial learning rates and be constructed in the algorithm. On the other hand, the replenishment exploration strategy can be changed parametrically for both players. Using the output of the dynamic replenishment algorithm, optimal Q-functions for both agents can be obtained. In this way, replenishment policies that allow both agents to respond optimally to each other in a competitive environment can also be found.

Algorithm 2 Nash Q-Learning against fixed replenishment

```

1: Input: Initial learning rate  $\alpha_0$ , strategy, epochs  $E$ , epsilon  $\epsilon$ , decay rate  $\sigma$ 
2: Output:  $Q_t^1, Q_t^2$ 
3: for each epoch  $e \in \{1, 2, \dots, E\}$  do
4:   Initialize state  $\bar{S}_t, t$ 
5:   Decay  $\alpha$  with  $\alpha_0 \cdot \sigma^e$ 
6:   while  $t \neq T$  do
7:     if strategy = Epsilon-Greedy then
8:       Generate random number  $r \in [0, 1]$ 
9:       if  $r < \epsilon$  then
10:         $X_t^1$  is selected randomly from action space  $A$ 
11:         $X_t^2 \leftarrow \lambda$ 
12:       else
13:        Compute Nash equilibrium for  $Q_t^1(\bar{S}_t, :, \lambda)$  and  $Q_t^2(\bar{S}_t, :, \lambda)$ 
14:        Select action  $X_t^1$  corresponding to the equilibrium
15:         $X_t^2 \leftarrow \lambda$ 
16:       end if
17:     else
18:       Compute Nash equilibrium for  $Q_t^1(\bar{S}_t, :, \lambda)$  and  $Q_t^2(\bar{S}_t, :, \lambda)$ 
19:       Select action  $X_t^1$  corresponding to the equilibrium
20:        $X_t^2 \leftarrow \lambda$ 
21:     end if
22:     Observe random incoming demand  $D_t$ 
23:     Calculate fulfillment metric  $F_j$  using  $(X_t^1, X_t^2)$  and joint state  $\bar{S}_t$ 
24:     Observe rewards  $C(\bar{S}_t, X_t^j(\bar{S}_t), D_t^j)$  and next state  $\bar{S}_{t+1}$ 
25:     Compute Nash-Q (equilibrium) values  $\text{Nash}Q_t^j(\bar{S}_{t+1})$  for each player  $j$ 
26:     Update Q-function of each player using

$$Q_t^j(\bar{S}_t, X_t^1, X_t^2) = (1 - \alpha)Q_t^j(\bar{S}_t, X_t^1, X_t^2) + \alpha[C(\bar{S}_t, X_t^j(\bar{S}_t), D_t^j) + \text{Nash}Q_t^j(\bar{S}_{t+1})]$$

27:      $\bar{S}_t \leftarrow \bar{S}_{t+1}$ 
28:      $t \leftarrow t + 1$ 
29:   end while(period)
30: end for(epoch)

```

Figure 3.5. Algorithm: nash q-learning against fixed replenishment.

With the optimal Q-functions obtained as a result of the first algorithm, the optimal replenishment policy of the first agent with respect to the other agent, with fixed replenishment in the same competitive environment, can be reached.

Algorithm 3 Nash Q-Learning against dynamic replenishment

```

1: Input: Initial learning rate  $\alpha_0^j$ , strategy, epochs  $E$ , epsilon  $\epsilon$ , decay rate  $\sigma$ 
2: Output:  $Q_t^1, Q_t^2$ 
3: for each epoch  $e \in \{1, 2, \dots, E\}$  do
4:   Initialize state  $\bar{S}_t, t$ 
5:   Decay  $\alpha^j$  with  $\alpha_0^j \cdot \sigma^e$ 
6:   while  $t \neq T$  do
7:     if strategy = Epsilon-Greedy then
8:       Generate random number  $r \in [0, 1]$ 
9:       if  $r < \epsilon$  then
10:         $X_t^1$  is selected randomly from action space  $A$ 
11:         $X_t^2$  is selected randomly from action space  $A$ 
12:       else
13:        Compute Nash equilibrium for  $Q_t^1(\bar{S}_t, :, :)$  and  $Q_t^2(\bar{S}_t, :, :)$ 
14:        Select actions  $X_t^1, X_t^2$  corresponding to the equilibrium
15:       end if
16:     else
17:       Compute Nash equilibrium for  $Q_t^1(\bar{S}_t, :, :)$  and  $Q_t^2(\bar{S}_t, :, :)$ 
18:       Select actions  $X_t^1, X_t^2$  corresponding to the equilibrium
19:     end if
20:     Observe random incoming demand  $D_t$ 
21:     Calculate fulfillment metric  $F_j$  using  $(X_t^1, X_t^2)$  and joint state  $\bar{S}_t$ 
22:     Observe rewards  $C(\bar{S}_t, X_t^j(\bar{S}_t), D_t^j)$  and next state  $\bar{S}_{t+1}$ 
23:     Compute Nash-Q (equilibrium) values  $\text{Nash}Q_t^j(\bar{S}_{t+1})$  for each player  $j$ 
24:     Update Q-function of each player using

$$Q_t^j(\bar{S}_t, X_t^1, X_t^2) = (1 - \alpha^j)Q_t^j(\bar{S}_t, X_t^1, X_t^2) + \alpha^j[C(\bar{S}_t, X_t^j(\bar{S}_t), D_t^j) + \text{Nash}Q_t^j(\bar{S}_{t+1})]$$

25:      $\bar{S}_t \leftarrow \bar{S}_{t+1}$ 
26:      $t \leftarrow t + 1$ 
27:   end while
28: end for

```

Figure 3.6. Algorithm: nash q-learning against dynamic replenishment.

Using the algorithms shown in this section, the perishable inventory control problem is solved in a competitive environment with n agents. In the next chapter 4, we analyze the performance of these algorithms on different input sets.

4. COMPUTATIONAL RESULTS

Computational experiments are conducted using the Nash Q-Learning algorithms explained in Chapter 3 to solve the perishable inventory control problem in a competitive environment. Computational experiments are conducted in a two-player competitive environment. Using cases with different parameters, results are obtained, including the training time of the algorithms, the winning rate of the players in the competitive environment, and comparison to the dynamic programming solution. The final comments and observations are based on these test cases.

All algorithms are developed using Python programming language and all training and test cases are performed on the same computer equipped with Intel Dual-Core i5 2.3 GHz - 16 GB 2133 Mhz memory. With the algorithms developed, two main code concepts are created. Firstly, agents are trained with different hyper-parameters using Nash Q-Learning algorithms, and agents' Q-functions are saved in specified intervals during training. For example, with the specified parameters (e.g. $\alpha = 0.7$, strategy = Epsilon-Greedy), the training process takes about 150000 iterations (epoch) and when an agent is saved every 5K epoch, a total of 30 agents are recorded. The second code concept is to run these trained agents more than once in a game setup to analyze them depending on their epoch level. Thus, the learning and performance of the agents is analyzed and reported as the epoch progresses in the testing process using recorded agents at different epoch levels.

In this chapter, the experimental setup for testing the algorithms is described in Section 4.1. In addition, the parameters over which the test cases are made are also detailed in this section. In Section 4.2, we analyze the performance of the Nash Q-Learning algorithm against a player that uses fixed replenishment as its action. In Section 4.3, we analyze the performance of the Nash Q-Learning algorithm in an environment where all players can dynamically replenish. Finally, Section 4.4 summarizes the results and comments of the analysis.

4.1. Experimental Design

Each test instance in the experimental setup is designed as a competitive perishable inventory control game to analyze the performance of the algorithms. The game proceeds in the order of events as in Figure 3.3 and is run with different parameters defined. The definition of a test instance in our thesis means analyzing the agents trained with the parameters in the context of a game setup. In other words, agents saved at different epoch levels during the training process are considered as a test instance and interpreted accordingly. For each test instance, the algorithms are run 100 times separately with learned Q-functions up to that point, and the statistics such as win rate, cost gap are analyzed. In addition, by fixing the seed during each run, it is ensured that the effect of factors such as incoming demand containing randomness is not misinterpreted in the comparison.

We can analyze the experimental design of algorithms under two headings: agent training and agent testing. We develop two algorithms to solve the perishable inventory control problem using the Nash Q-Learning algorithm. The first Algorithm 3.5 consists of an environment where one player makes a fixed replenishment in each period equal to the average of the random demand (e.g., λ in Poisson distribution), and the other player can give the best replenishment action response to this fixed replenishment player. The goal in the fixed replenishment case is to see whether the agent that trains with Nash Q-Learning defeats the agent that uses fixed replenishment. In the second Algorithm 3.6, each agent in the competitive environment dynamically tries all actions in their action space A to ensure that each agent takes the best replenishment action that it can take in response to other agents in period t and joint state $\bar{S}_t$. In these two algorithms, agents update their Q-values in each epoch until the last epoch E is reached, and as the Q-values are updated, learning progresses since agents use these Q-values in the relevant period, state, and action indexes within the Q-functions and make their replenishment decision accordingly. As learning progresses, replenishment decisions converge to equilibrium solutions.

Agents are recorded at certain intervals during the learning process. In other words, the Q-functions of the agents are recorded at intervals during the training period to analyze how the agent learning process progresses more effectively. In this experimental setup, agents are saved and stored at every 5K or 10K epoch interval during training. Thus, these saved agent files are used in the agent testing part, which is the second stage of the experimental setup. In addition to the training phase, both algorithms work according to the specified input hyper-parameters. These parameters, such as the initial learning rate α and the exploration strategy, which can affect the learning convergence within the algorithm, constitute each test instance in the experimental setup. Thus, we run algorithms with different parameters, test the saved agents, and analyze the results in different test cases. The test design parameters are shown in Table 4.1.

Table 4.1. Test design parameters.

Parameter	Values
Product Lifetime (m)	2, 3
Horizon Length (T)	6, 10
Initial Learning Rate (α)	0.5, 0.7
Learning Strategy	Greedy, Epsilon-Greedy

As can be seen in the parameter table, the two Nash Q-Learning algorithms that we implement, including the fixed replenishment and dynamic replenishment cases, are evaluated within the scope of two different product lifetime m , horizon length T , initial learning rate α and exploration (learning) strategies. The learning rate parameter used in the algorithms is implemented with exponential decay, which is a state-of-the-art learning method, so there is an initial learning rate definition in the parameter table. As the epoch progresses, the learning rate gradually decreases depending on the decay rate, reaching a predetermined minimum of 0.3. The exponentially decaying learning rate curve is shown in Figure 4.1.

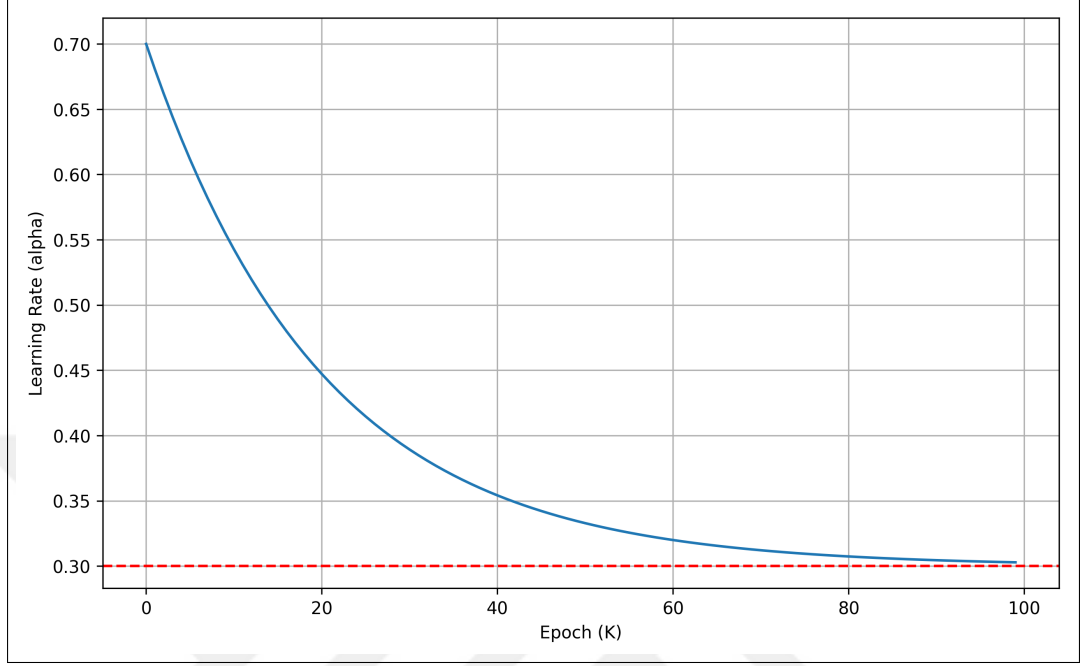


Figure 4.1. Exponentially decaying learning rate.

In addition, the exploration strategy provided as input also affects the algorithm and solution convergence. The exploration strategy determines the agents' action selection strategy within the action space and the action space coverage. In other words, there is an opportunity to observe the exploration-exploitation trade-off according to the chosen strategy. While the Greedy strategy tends to implement actions experienced through exploitation, Epsilon-Greedy is a strategy for exploration.

In the second phase of the experimental design, agent testing, the results are analyzed by playing the saved agents that are trained with different parameters within the scope of a game scenario. The test setup is the same in both the fixed replenishment case and the dynamic replenishment case. The test process is as follows: for each period within the specified horizon length T , the agents take a replenishment action according to the joint inventory state at the beginning of the period. Later, a fulfillment metric is determined according to the inventory states updated according to the replenishment decisions, and demand is allocated based on the joint inventory state.

At the end of the day, the environment returns a reward (negative cost) to each agent. This process repeats for all periods in the horizon length. The agent, with the minimum cost at the end of the horizon length, wins the game. The saved agents are put into iteration at certain intervals, and this game is played 100 times separately for each saved agent pair, and the statistical results of these outputs are analyzed. Using parameters learned by self-play, we test the learned strategy in different games.

In test cases with product lifetime $m = 2$, a comparison mechanism is developed to examine the performance of the Nash Q-Learning algorithm in more detail as the epoch progresses. In problems requiring sequential decision-making, such as perishable inventory control, the optimal solution is reached with DP-based algorithms. Here, the benchmark that we use is the DP-based equilibrium outputs. In other words, the performance of the algorithm is interpreted as directly proportional to how much it converges to the DP-based equilibrium solution. The reason we only compare DP-based equilibrium solution in cases with $m = 2$ is that we do not have a solution in larger sets of problems, such as $m = 3$ due to the exponential time complexity of the dynamic programming algorithm.

For both cases, evaluations are made on similar test metrics. The training times of the algorithms and the durations of the dynamic programming and the equilibrium solution output are reported. The win-rate (percentage) that agents receive in different epochs is shown. Finally, the average cost gaps according to the benchmarks are examined, and where the learning process converges to optimal/equilibrium solutions is analyzed. In the following sections of the chapter, the analyses of both algorithms are performed with different test instances.

4.2. Competitive Nash-Q Learning under Fixed Replenishment

In the fixed replenishment case, a player (e.g., pharmacy) in the competitive environment replenishes at the same rate as the average demand at the beginning of each period as fixed.

Since the Poisson distribution is chosen as the demand distribution during computational experiments, this pharmacy replenishes at the same rate as λ . The other pharmacy in the environment continues its learning process by experiencing different replenishment actions within the action space according to the exploration strategy provided as input. The main objective of examining this case is to analyze the dominance of agent learning with Algorithm 3.5 implemented with the Nash-Q-Learning technique can be compared to the agent performing fixed replenishment in a competitive environment. Thus, the operability and performance of the algorithm can be clearly analyzed.

During the training process, one pharmacy can dynamically select a replenishment action, while the other pharmacy performs fixed replenishment. In other words, while a pharmacy can experience all replenishment within the action space according to the replenishment actions that the other pharmacy can take, the pharmacy performing fixed replenishment updates the λ action-indexed values within the Q-function. In addition to the training process, the algorithms are run for different combinations of parameters in the fixed replenishment case at (product lifetimes) $m=2$ and $m=3$ and for 150K and 125K epochs, respectively, and agent Q-functions are saved at every 5K interval.

In the fixed replenishment case, the dynamic programming (DP) equilibrium solution is used as a benchmark in the algorithm performance analysis. The main reason for this is that it provides an exact solution to the perishable inventory control problem. In other words, in the environment where the second pharmacy gives $X_t = \lambda$, the equilibrium solution is obtained with the cost expectations made considering all periods, states and actions of the first pharmacy. However, this comparison is found in test cases with $m = 2$ because dynamic game solutions cannot be obtained in scenarios where $m \geq 3$ due to the time complexity of the algorithm.

Table 4.2. Fixed replenishment: training runtime (hours).

E	m = 2		m = 3	
	T = 6	T = 10	T = 6	T = 10
5K	0.086	0.1237	0.3958	0.6401
15K	0.246	0.3707	2.7155	3.5178
35K	0.562	0.8657	8.7459	9.2312
45K	0.721	1.1137	9.4930	12.3688
55K	0.881	1.3634	11.7843	15.5065
65K	1.039	1.6141	14.1540	18.6442
75K	1.199	1.8648	16.5236	21.7819
95K	1.519	2.3661	21.2630	28.0572
105K	1.680	2.6168	23.6326	31.1949
115K	1.840	2.8674	26.0023	34.3326
125K	2.000	3.1181	28.3719	37.4703
150K	2.401	3.7448	-	-

Table 4.2 summarizes the training runtime (hours) of the algorithm in the fixed replenishment case. In other words, the table illustrates the runtime of the agents in hours, which learning epochs have been completed and saved. Detailed training runtime information is shown in Table A.1. As the horizon length T and product lifetime m in the algorithm input increase, the algorithm training runtime also increases. The main reason for this increase in the training times of agents, especially in product lifetime transitions while keeping T and epoch the same, is the exponential growth of the state space $(|A|^N)^m$, where A is the action space, N is the number of players, and m is the product lifetime). However, the training-time differences in horizon length transitions are similar under the same product lifetime.

In addition to the Nash Q-Learning training runtime, it is also necessary to consider the DP-based equilibrium solution runtime used as a benchmark for comparison. Due to the algorithm runtime complexity, the DP-based equilibrium algorithms are

run for cases with $m = 2$ and the results are obtained. Under product lifetime $m = 2$, the algorithms took 2.55 and 4.25 hours to run for the horizon length $T = 6$ and $T = 10$, respectively. The Nash Q-Learning algorithm is generally completed in a shorter time than the DP-based equilibrium algorithm, but another important point here is at what epoch level the Nash Q-Learning algorithm converges in terms of performance. Since it would be more accurate to make a comparison according to this convergence, the training times in Table 4.2 are mentioned in the following sections.

In the cases where product lifetime $m = 2$, the performance of the algorithm is analyzed by making a comparison with the DP-based equilibrium solutions. In the game scenario, agents trained with different parameters compete with another pharmacy during the test and the results are obtained. In order to compare the performances of these agents according to equilibrium solution, an environment is prepared in which a pharmacy that takes action according to the DP-based equilibrium and a pharmacy that performs fixed replenishment are prepared in a cloned environment (same demand and fixed seed). Thus, the performance of Nash Q-Learning agents saved at different epochs and the agent that performs replenishment according to DP-based equilibrium can be compared against a pharmacy that performs fixed replenishment in a fair environment. In addition, the cost structure and parameters related to the environment within the system, the reward mechanism, and structures such as transition functions are also completely common.

Table 4.3. Fixed replenishment: cost gap (%) to dp-based equilibrium.

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
5K	Greedy	53.7	56.2	54.2	35.9
	Epsilon-Greedy	39.8	49.7	34.3	48.7
25K	Greedy	38.2	12.2	40.1	32.5
	Epsilon-Greedy	25.3	29.9	34.6	36.2
45K	Greedy	19.2	7.5	29.3	25.1
	Epsilon-Greedy	47.5	19.9	29.7	24.3
65K	Greedy	14.9	46.6	32.7	24.8
	Epsilon-Greedy	43.1	9.6	29.6	16.0
90K	Greedy	14.0	14.3	26.0	17.2
	Epsilon-Greedy	6.9	6.1	18.3	13.6
105K	Greedy	6.2	18.3	15.8	10.6
	Epsilon-Greedy	10.2	4.8	17.7	3.6
125K	Greedy	4.6	9.0	11.2	7.8
	Epsilon-Greedy	5.3	1.9	12.1	6.1
145K	Greedy	2.8	4.9	7.0	5.4
	Epsilon-Greedy	6.0	0.7	4.6	3.0
150K	Greedy	3.6	2.6	5.2	4.3
	Epsilon-Greedy	9.2	0.6	9.6	4.5

Table 4.3 shows the average cost gap percentages of epoch-based agents trained with different parameters, compared to the agent that takes replenishment actions according to the DP-based equilibrium in the cloned environment. In other words, it summarizes how much the average cost of the Nash Q-Learning agent obtained as a result of each test instance is different in percentage terms compared to DP-based equilibrium. Detailed cost gap information in the fixed replenishment case is illustrated in Table A.2. As learning progresses, a decrease in the percentages of cost gaps is observed in all parameters.

In particular, with the Epsilon-Greedy exploration strategy, convergence to the DP-based equilibrium solution is faster at $\alpha = 0.7$ learning rate. It progresses with a 5% gap with the agents saved at 100K and shows almost the same cost performance as the DP-based equilibrium at 125K epoch levels. When we look at the general average, the Epsilon-Greedy and $\alpha = 0.7$ parameter pairs are in the test instances that show the closest performance to DP-based equilibrium solution.

The performance of the agent trained with the Epsilon-Greedy exploration strategy and the set of parameters containing $\alpha = 0.7$ learning rate is shown in Figure 4.2. When the average cost gap percentages are examined, the cost behaviors of the

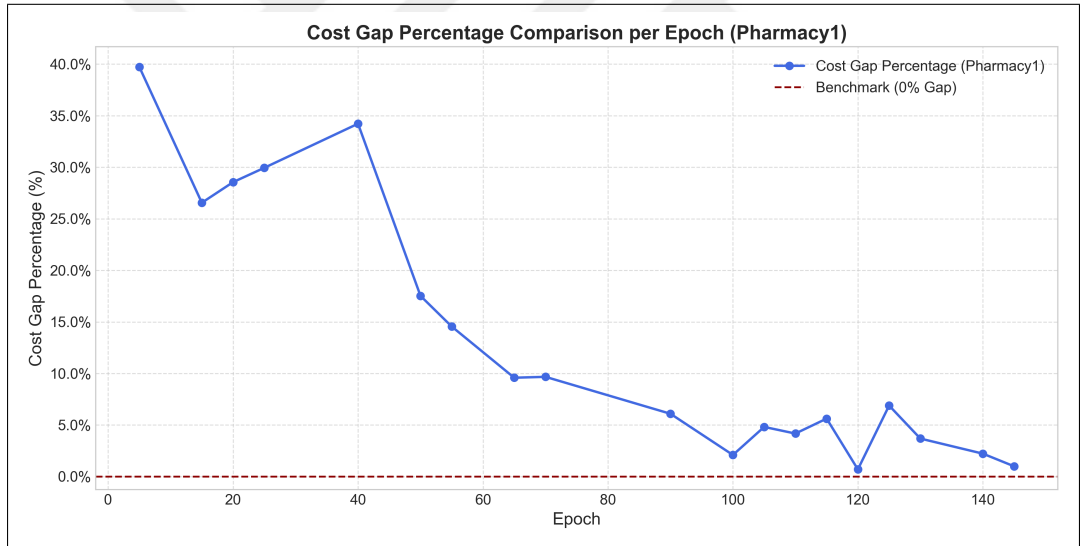


Figure 4.2. Fixed replenishment: cost gap (%) to dynamic programming.

agents saved at 100K and later are very close to each other with DP-based equilibrium solution. This shows that the Nash Q-Learning algorithm gives equilibria results from this epoch level and converges to the DP-based equilibrium solution.

Another metric that we can use to examine the performance of the Nash Q-Learning algorithm is the win-rate. The dominance of the pharmacy that learns with the Nash Q-Learning algorithm and makes replenishment decisions based on these learned Q-functions over the pharmacy that performs fixed replenishment can be mea-

sured with this win-rate metric. In other words, training agent learns the policy from the Q-function and the equilibrium. In particular, considering the information of the epoch-based agents, it also provides insight into which epoch the agents are now more dominant than the pharmacy that performs fixed replenishment. Since the DP-based equilibrium algorithm gives equilibria results, it is the winner of all games. For this reason, a comparison of the Nash Q-Learning algorithm is interpreted as reaching a 100% win-rate like the DP-based algorithm.

Table 4.4. Fixed replenishment: win-rate (%) under $m = 2$.

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
5K	Greedy	40	35	0	0
	Epsilon-Greedy	46	46	100	34
25K	Greedy	60	58	66	46
	Epsilon-Greedy	36	54	58	0
45K	Greedy	78	74	100	74
	Epsilon-Greedy	54	45	20	100
65K	Greedy	32	64	100	80
	Epsilon-Greedy	50	52	100	100
85K	Greedy	40	100	100	100
	Epsilon-Greedy	100	74	94	28
105K	Greedy	44	84	100	56
	Epsilon-Greedy	100	44	84	68
125K	Greedy	86	44	100	20
	Epsilon-Greedy	60	100	100	100
145K	Greedy	95	62	100	36
	Epsilon-Greedy	92	100	80	100
150K	Greedy	94	78	60	88
	Epsilon-Greedy	86	100	92	100
Average	Greedy	68.13	73.67	88.13	67.73
	Epsilon-Greedy	67.80	75.00	63.13	70.20

Table 4.4 illustrates the win rates (%) of Nash Q-Learning agents trained with different parameters in the game with product lifetime $m = 2$ and detailed win-rate information is shown in Table A.3. The table demonstrates that the Greedy strategy achieves consistently high win-rates as training epochs increase, particularly at higher training counts. In contrast, the Epsilon-Greedy strategy, which balances exploration and exploitation, displays more variability, occasionally outperforming Greedy at certain iterations but lacking consistency. At $T = 6$, the Greedy strategy generally performs better, while Epsilon-Greedy exhibits strong but inconsistent results. At $T = 10$, both strategies achieve high win-rates, although Greedy shows a slight advantage in average performance. The choice of α affects the results, with no universally optimal setting. Overall, the Greedy strategy is more reliable and achieves higher average win-rates, while Epsilon-Greedy demonstrates competitive but less predictable performance. When we consider the win-rate averages, we can say that the strategy-learning rate pair that dominates the win-rate metric the most is Greedy - 0.5, compared to the pharmacy that takes fixed replenishment actions.

As the dynamic replenishment agent in a competitive environment converges to the DP-based equilibrium policy, an increase in the win-rate is observed. When converging policies are analyzed, the objective is to be the assignee to meet the incoming demand. During the horizon length, the trained agent tries to keep its fulfillment metric high and minimize its own cost by exposing the fixed replenishment player to both lost sales and deterioration cost. When the agent trained with Epsilon Greedy - $\alpha = 0.7$ parameter set is analyzed in the set of games with $m = 2$ and $T = 6$, it is seen in Figure 4.3 that as the epoch progresses, the rate of receiving the demand on average of 100 games increases with the learned policies. When the cost distribution of the agents is analyzed, the holding and ordering costs make up about 54% of the costs of the trained agent in the game. On the other hand, deterioration and shortage costs account for approximately 75% of the total cost of the fixed replenishment agent. It is observed that the trained agent wins the game by taking the deterioration cost into account and making higher replenishment than the fixed replenishment agent, exposing this agent to high deterioration and shortage costs.

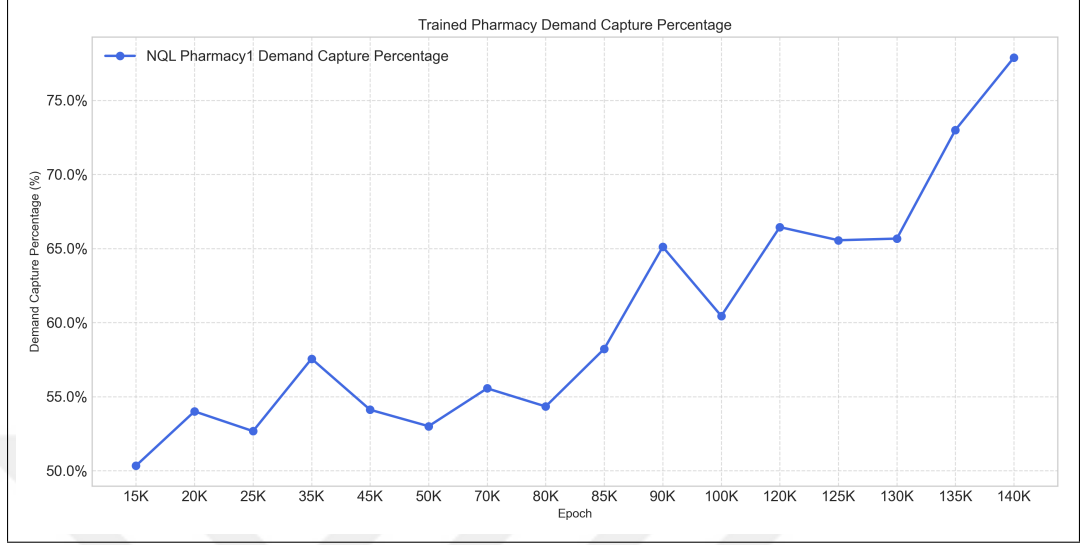


Figure 4.3. Fixed replenishment: demand capture percentage of trained agent.

In the first and last periods of the horizon length, the agent makes a low replenishment. However, since it does not meet the demand and results in lost sales, it reacts with high replenishment in the next period and continues in a similar way for the next few periods. When the training agent decides to replenish, it is observed that it replenishes more than the fixed agent replenishes. In the game set with $m = 2$, $\lambda = 3$ and $T = 6$, it is observed that the agent trained with Epsilon Greedy - $\alpha = 0.7$ follows a replenishment policy $[2, 5, 4, 5, 4, 0]$ under the same seed. Each number in the array indicates the amount of replenishment in the corresponding period. Against an agent that replenishes λ in each period, trained agent wins games by taking all the demand after the first period, exposing the fixed agent to both lost sales and deterioration cost.

Table 4.5 also illustrates the win-rates of agents trained in the environment with product lifetime $m = 3$ and detailed win-rate information is shown in Table A.4.

Table 4.5. Fixed replenishment: win-rate (%) under $m = 3$.

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
5K	Greedy	48	64	15	24
	Epsilon-Greedy	45	30	45	56
25K	Greedy	72	68	45	16
	Epsilon-Greedy	56	45	60	100
45K	Greedy	96	65	38	28
	Epsilon-Greedy	100	60	56	56
65K	Greedy	100	68	28	76
	Epsilon-Greedy	52	60	67	28
85K	Greedy	80	24	70	20
	Epsilon-Greedy	64	65	78	60
105K	Greedy	100	70	84	84
	Epsilon-Greedy	56	100	88	52
125K	Greedy	56	68	96	83
	Epsilon-Greedy	100	100	95	100
Average	Greedy	76.56	64.04	59.92	52.68
	Epsilon-Greedy	77.70	64.04	61.72	64.40

The Greedy strategy improves consistently with more epochs, achieving its best win-rate (100%) at 105K iterations with $\alpha = 0.5$. However, it also exhibits weaknesses in some scenarios, such as 85K iterations with $\alpha = 0.7$, where the win-rate drops to 24%. In contrast, the Epsilon-Greedy strategy demonstrates more adaptability, often outperforming Greedy at lower training counts and excelling at higher iterations, particularly at 125K, where it achieves perfect or near-perfect win-rates in several configurations. The learning rate (α) also plays a significant role, with $\alpha = 0.5$ generally favoring Greedy, while $\alpha = 0.7$ provides an edge to Epsilon-Greedy under certain conditions. On average, Epsilon-Greedy achieves slightly higher win-rates in all configurations, with 77.70% at $\alpha = 0.5$ compared to 76.56% for Greedy.

Overall, the analysis highlights the strengths of Greedy in stable, well-learned environments and the robustness of Epsilon-Greedy in scenarios requiring exploration, making the latter a more versatile strategy across a wider range of conditions.

4.3. Competitive Nash-Q Learning under Dynamic Replenishment

In the dynamic replenishment case, each pharmacy in the competitive environment experiences all replenishment actions in the action space A and tries to respond in the best way to the actions that other pharmacies might take with self-play learning. In Section 4.2, one pharmacy performs fixed replenishment λ in each period, while the other pharmacy performs dynamic replenishment and learns the best replenishment action in each state and period. In this case, all pharmacies are given free access to the entire action space. Thus, the actions taken by pharmacies against each other in any state and period are analyzed and reported.

In algorithmic design, the dynamic replenishment case is similar to fixed replenishment. The difference is that in each period, instead of the second pharmacy performing a fixed replenishment, it replenishes according to the equilibrium actions obtained from the Q-functions as the other pharmacy. The training process is carried out using Algorithm 3.6 and pharmacies complete their learning through self-play. Different test instances are created with the experimental design parameters in Table 4.1. Each test instance is run for 125K epochs, and Q-functions of both pharmacies are saved every 5K epochs. Thus, the testing process is continued with Q-values recorded for each test instance.

Each Q-function pair recorded during training is tested in the test mechanism described in the experimental design. The epoch-wise saved pairs are played 100 times, and the average results are analyzed and reported. This testing environment is set up with the same parameters for both the Nash Q-Learning algorithm and the environment in which the equilibrium solutions are analyzed. Unlike the fixed replenishment case, the test design parameters are evaluated for each pharmacy in this case.

In the fixed replenishment case, the learning rate and the exploration strategy are defined as single parameters, whereas in this case each pharmacy has a separate learning rate and exploration strategy. Thus, the performance or dominance of pharmacies in the same competitive environment can be analyzed in different parameters.

Similarly to the performance analysis in the fixed replenishment case, we use the equilibrium solutions of the Markov Game proposed in Section 3.7.1 to evaluate the performance of the Nash Q-Learning algorithm in the dynamic replenishment case. The equilibrium solutions represent the equilibrium actions that each player takes in each state and period in the two-player competitive world. We also analyze whether the Nash Q-Learning algorithm converges to this equilibrium solution. This benchmark was performed on input sets with a product life $m = 2$ due to algorithm complexity. The average cost gap analysis of the pharmacies trained with the Nash Q-Learning algorithm in the test results is calculated and reported according to the equilibrium solutions. Another test metric, win-rate, shows the percentage of games won by pharmacies trained with different parameters in the game.

Table 4.6. Dynamic replenishment: training runtime (hours).

N	$m = 2$		$m = 3$	
	T = 6	T = 10	T = 6	T = 10
5K	0.45	0.64	2.05	3.31
15K	0.96	1.45	10.59	13.72
25K	1.33	2.03	12.13	20.03
35K	1.88	2.90	29.29	30.92
45K	2.38	3.68	31.38	40.89
55K	2.71	4.20	36.33	47.80
75K	3.38	5.25	46.56	61.38
85K	3.71	5.77	51.54	67.98
105K	4.37	6.80	61.46	81.12
125K	5.89	9.18	83.55	110.35

Table 4.6 illustrates the training runtime (hours) of the algorithm in the dynamic replenishment case. The detailed training runtime of the dynamic replenishment case is illustrated in Table B.1. Under the same product lifetime m , the training times at the same epochs increase as the horizon length T increases. Similarly to training times in the fixed replenishment case, since the state space grows, the times increase more when the product lifetime m increases. In addition, the training times in the dynamic replenishment case are relatively longer than in the fixed replenishment case. This is due to the increased complexity of all pharmacies in the competitive world as they have access to the entire action space.

In addition to the training times of the Nash Q-Learning algorithm, it is also necessary to examine the times of the algorithm developed to obtain Markov game equilibrium solutions. For product lifetime $m = 2$, this algorithm was run at two different horizon lengths and equilibrium solutions were obtained. For $T = 6$ and $T = 10$, the algorithm takes 9.22 and 13.24 hours, respectively. Later in the section, it is useful to compare these training times during cost gap analysis and algorithm convergence. However, as can be seen from the table, the Nash Q-Learning algorithm is faster than the DP-Based equilibrium algorithm at the product lifetime $m = 2$. Due to algorithm and time complexity, equilibrium solutions cannot be obtained for product lifetime $m \geq 3$. In the dynamic replenishment case, agent pairs recorded through Nash Q-Learning algorithms with different parameters are played 100 times in the test environment to obtain different test metrics. In order to measure the performance of the algorithm, equilibrium solutions are taken as benchmarks, especially in test cases with product lifetime $m = 2$. Thus, it can be observed if the outputs of the algorithm converge to equilibrium solutions.

As in the fixed replenishment case, the DP-based equilibrium algorithm with Nash Q-Learning is run 100 times in the same way in two cloned environments and a comparison is made based on the average of the results. The cost gap percentage metric is useful to observe whether pharmacies' replenishment decisions and policies converge to equilibrium solutions as the epoch progresses. This is because under the

same environment and demand distribution, the comparison between the policies of two pharmacies according to the Nash Q-Learning algorithm and the policies according to the equilibrium solutions is based on the average cost gap.

Table 4.7. Dynamic replenishment ($m = 2, T = 6$): cost gap (%) to equilibrium solution.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	30.94	33.14	30.98	44.03	39.36	32.44
	Greedy - E. Greedy	37.36	44.09	29.42	28.02	29.36	28.09
	Epsilon-Greedy - E. Greedy	46.09	53.17	49.36	38.09	41.53	42.35
25K	Greedy - Greedy	17.17	10.41	13.18	26.67	15.06	16.06
	Greedy - E. Greedy	20.05	28.22	20.05	18.22	20.05	18.22
	Epsilon-Greedy - E. Greedy	34.12	38.79	40.63	28.22	19.83	13.34
45K	Greedy - Greedy	5.39	9.55	4.25	14.95	12.45	12.58
	Greedy - E. Greedy	14.20	22.33	14.10	12.58	14.20	6.33
	Epsilon-Greedy - E. Greedy	28.74	25.23	34.20	12.33	8.25	9.60
65K	Greedy - Greedy	7.32	2.79	3.27	10.72	8.50	10.74
	Greedy - E. Greedy	8.92	15.48	8.05	2.13	18.92	2.48
	Epsilon-Greedy - E. Greedy	16.80	25.69	28.92	5.48	4.14	5.86
85K	Greedy - Greedy	1.90	2.18	4.76	15.83	5.97	7.45
	Greedy - E. Greedy	5.26	13.68	5.78	3.18	5.26	3.68
	Epsilon-Greedy - E. Greedy	22.50	24.05	20.26	3.68	3.73	5.55
105K	Greedy - Greedy	0.76	2.32	0.18	4.60	9.78	7.42
	Greedy - E. Greedy	1.35	6.73	0.35	0.73	7.35	0.73
	Epsilon-Greedy - E. Greedy	5.12	13.83	17.47	1.41	4.47	0.83
110K	Greedy - Greedy	0.63	0.53	0.82	4.65	9.54	4.03
	Greedy - E. Greedy	1.19	4.28	0.19	0.28	3.19	0.28
	Epsilon-Greedy - E. Greedy	8.11	9.61	11.59	2.27	0.85	0.29
120K	Greedy - Greedy	0.44	0.33	0.92	2.93	6.56	6.26
	Greedy - E. Greedy	0.55	4.37	0.17	0.26	5.55	0.37
	Epsilon-Greedy - E. Greedy	9.25	7.12	5.55	0.37	0.22	0.50
125K	Greedy - Greedy	0.71	0.53	0.61	2.45	7.08	6.07
	Greedy - E. Greedy	0.80	2.16	0.40	0.24	5.80	0.16
	Epsilon-Greedy - E. Greedy	6.51	9.39	5.80	0.16	0.47	0.20

Table 4.7 shows the percentages of the cost gaps of the pharmacies trained and saved with different parameters at different epoch levels to the equilibrium solutions after 100 games. Detailed cost gap information of pharmacies under $m = 2$ and $T = 6$ is shown in Table B.2. The performance of pharmacies with different learning rates and exploration strategies in the same competitive environment is analyzed. When analyzing the average cost gaps of both pharmacies, it is seen that both pharmacies reached the lowest values in $\alpha = 0.5$ and Greedy exploration strategies. On the other hand, the lowest average cost gap was observed in the Epsilon-Greedy case and $\alpha = 0.7$ as an individual pharmacy. The strongest parameter fits are Greedy - $\alpha = 0.5$ and Epsilon Greedy - $\alpha = 0.7$.

When the table is analyzed in general, in all test cases, as the epoch progresses, the cost gap of the pharmacies in the case to the equilibrium solutions decreases and the learning continues successfully. In particular, in the Greedy - $\alpha = 0.5$ Epsilon Greedy - $\alpha = 0.7$ case of Pharmacy 1 and 2 respectively, both pharmacies perform their learning quickly and converge to equilibrium solutions. After 100K epochs, both pharmacies' equilibrium policies converge by making replenishment at almost the same costs as equilibrium solutions.

Table 4.8. Dynamic replenishment ($m = 2, T = 10$): cost gap (%) to equilibrium solution.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	32.45	35.13	32.01	45.78	40.12	33.49
	Greedy - E. Greedy	38.69	45.00	30.41	28.85	29.91	29.79
	Epsilon-Greedy - E. Greedy	47.37	54.86	51.01	39.70	43.06	43.77
25K	Greedy - Greedy	18.92	12.12	14.09	28.00	16.20	17.06
	Greedy - E. Greedy	21.86	28.78	21.08	19.85	21.77	18.97
	Epsilon-Greedy - E. Greedy	34.72	40.79	41.14	30.04	21.10	14.54
45K	Greedy - Greedy	7.12	10.97	5.84	16.74	13.23	13.12
	Greedy - E. Greedy	16.19	23.34	15.86	13.57	15.75	7.29
	Epsilon-Greedy - E. Greedy	30.30	26.80	34.90	13.41	8.95	10.45
65K	Greedy - Greedy	8.51	3.70	4.96	11.34	9.97	11.73
	Greedy - E. Greedy	9.96	17.27	9.62	4.02	20.33	3.84
	Epsilon-Greedy - E. Greedy	18.42	26.89	30.84	6.68	5.95	7.76
85K	Greedy - Greedy	3.75	4.16	5.30	16.69	7.82	9.43
	Greedy - E. Greedy	6.86	15.29	5.87	5.67	6.90	4.24
	Epsilon-Greedy - E. Greedy	24.45	24.68	21.14	4.37	4.64	7.33
95K	Greedy - Greedy	4.07	3.27	3.09	4.32	6.82	12.22
	Greedy - E. Greedy	3.92	8.86	2.10	4.63	7.73	1.82
	Epsilon-Greedy - E. Greedy	16.44	20.49	17.03	3.73	7.61	3.51
105K	Greedy - Greedy	1.62	3.86	1.43	5.47	11.29	9.31
	Greedy - E. Greedy	1.89	7.53	1.05	2.23	8.27	2.48
	Epsilon-Greedy - E. Greedy	6.84	14.38	18.48	3.22	17.85	2.35
120K	Greedy - Greedy	1.56	1.87	1.44	4.59	8.20	8.20
	Greedy - E. Greedy	1.76	5.96	2.27	2.36	6.81	1.13
	Epsilon-Greedy - E. Greedy	11.00	8.95	7.05	1.62	1.80	1.45
125K	Greedy - Greedy	2.43	2.35	1.66	3.59	7.83	7.45
	Greedy - E. Greedy	1.98	3.35	2.33	0.80	6.67	0.74
	Epsilon-Greedy - E. Greedy	7.17	9.97	7.37	1.10	1.09	2.07

Table 4.8 similarly shows the cost gap percentages found with horizon length $T = 10$ and detailed cost gap information for pharmacies is illustrated in Table B.3. According to the average cost gaps of individual pharmacies, the lowest value is observed with the parameters Epsilon Greedy - $\alpha = 0.7$, while the parameter pair where both pharmacies in the same environment reach the lowest average cost gap is Greedy- $\alpha = 0.5$.

As the epoch progresses, the cost gaps of the pharmacies decrease as expected, but Epsilon Greedy - $\alpha = 0.5$ is the most successful parameter pair. While this exploration strategy achieves very good results with $\alpha = 0.7$, it does not achieve satisfactory results with a learning rate of 0.5. Figure 4.4 shows the cost gaps of trained and tested

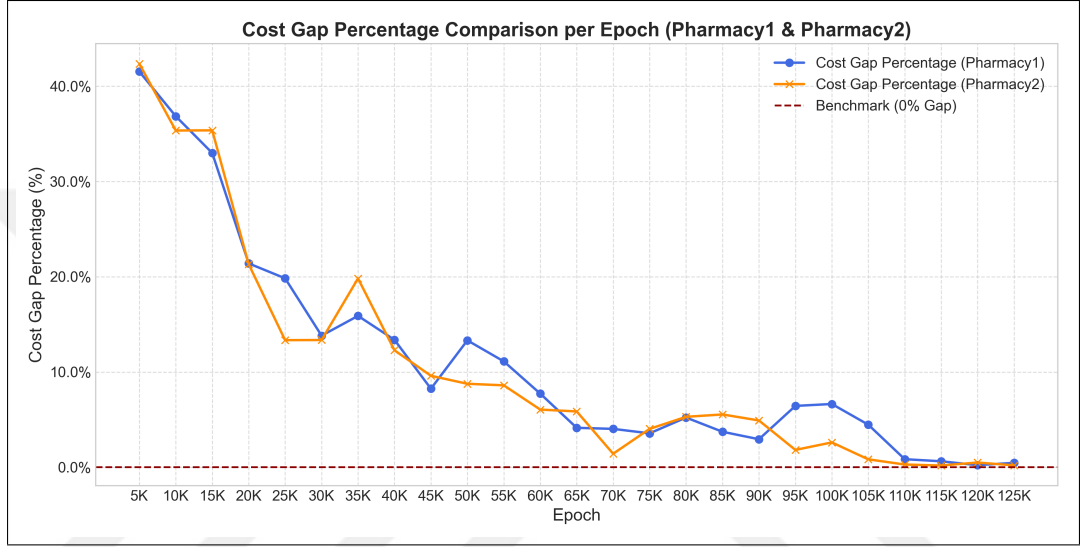


Figure 4.4. Algorithm performance: epsilon greedy - $\alpha = 0.7$.

pharmacies with Epsilon Greedy - $\alpha = 0.7$ parameters to equilibrium solutions. After 70K epochs, the convergence of equilibrium solutions occurred in both pharmacies. Especially after 100K epochs, pharmacies recorded with Nash Q-Learning algorithm make replenishment with almost the same cost as equilibrium solutions. Thus, the Nash-Q Learning algorithm converges twice as fast in the environment at the same product lifetime and horizon length than algorithms that return equilibrium solutions.

Figure 4.5 shows the cost gaps of Pharmacies 1 and 2 trained with Greedy - $\alpha = 0.5$ and Epsilon Greedy - $\alpha = 0.7$ parameters, respectively. The Epsilon Greedy - $\alpha = 0.7$ parameter pair exhibits a more dominant performance and converges to equilibrium solutions with generally lower cost gap percentages.

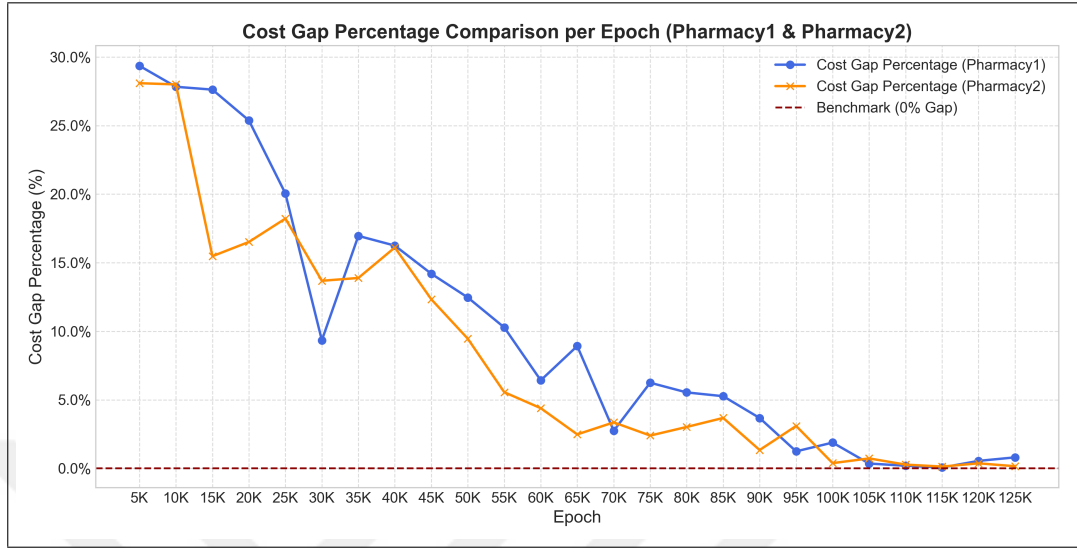


Figure 4.5. Algorithm performance comparison: greedy - $\alpha = 0.5$ vs. epsilon greedy - $\alpha = 0.7$.

Table 4.9. Dynamic replenishment: win-rate (%) under $m = 2$ and $T = 6$.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	40.00	60.00	27.00	73.00	48.00	52.00
	Greedy - E. Greedy	54.00	46.00	35.00	65.00	54.00	46.00
	Epsilon-Greedy - E. Greedy	40.00	50.00	53.00	47.00	75.00	25.00
25K	Greedy - Greedy	76.00	24.00	14.00	86.00	56.00	44.00
	Greedy - E. Greedy	38.00	62.00	15.00	85.00	44.00	56.00
	Epsilon-Greedy - E. Greedy	88.00	12.00	57.00	43.00	52.00	48.00
45K	Greedy - Greedy	52.00	48.00	54.00	46.00	56.00	44.00
	Greedy - E. Greedy	56.00	44.00	58.00	42.00	85.00	15.00
	Epsilon-Greedy - E. Greedy	84.00	16.00	45.00	55.00	55.00	45.00
65K	Greedy - Greedy	24.00	76.00	58.00	18.00	44.00	56.00
	Greedy - E. Greedy	65.00	35.00	42.00	58.00	48.00	52.00
	Epsilon-Greedy - E. Greedy	80.00	20.00	45.00	55.00	42.00	58.00
85K	Greedy - Greedy	54.00	46.00	82.00	18.00	68.00	32.00
	Greedy - E. Greedy	72.00	28.00	55.00	45.00	45.00	55.00
	Epsilon-Greedy - E. Greedy	88.00	12.00	40.00	60.00	60.00	40.00
105K	Greedy - Greedy	54.00	46.00	92.00	8.00	60.00	40.00
	Greedy - E. Greedy	74.00	26.00	56.00	44.00	20.00	80.00
	Epsilon-Greedy - E. Greedy	26.00	74.00	10.00	90.00	30.00	70.00
115K	Greedy - Greedy	60.00	40.00	78.00	22.00	46.00	54.00
	Greedy - E. Greedy	60.00	38.00	47.00	51.00	0.00	95.00
	Epsilon-Greedy - E. Greedy	40.00	60.00	10.00	88.00	32.00	68.00
120K	Greedy - Greedy	55.00	45.00	82.00	18.00	36.00	64.00
	Greedy - E. Greedy	85.00	15.00	48.00	52.00	8.00	92.00
	Epsilon-Greedy - E. Greedy	26.00	74.00	0.00	100.00	24.00	76.00
125K	Greedy - Greedy	50.00	50.00	80.00	20.00	70.00	30.00
	Greedy - E. Greedy	100.00	0.00	38.00	62.00	10.00	90.00
	Epsilon-Greedy - E. Greedy	72.00	28.00	0.00	100.00	72.00	28.00
Avg	Greedy - Greedy	50.92	49.00	59.08	39.80	50.40	49.60
	Greedy - E. Greedy	64.48	35.44	47.16	52.76	43.08	56.72
	Epsilon-Greedy - E. Greedy	49.84	49.44	38.68	61.24	49.64	50.28

Table 4.9 shows the win rates of pharmacies that have completed training with different exploration strategies and α in a competitive environment with the product lifetime $m = 2$ and the horizon length $T = 6$. Detailed win-rate information for pharmacies is shown in Table B.4. The average win rates of the pharmacy pairs saved at different epochs are analyzed by running the game 100 times. For games run with the same parameter sets (e.g., Greedy-Greedy & $\alpha = 0.5$ - $\alpha = 0.5$), the percentages are close to each other. The main reason for this is that the best response of the two pharmacies to each other is similar. In other words, pharmacies with the same parameters cannot outperform each other because they converge to similar policy paths and equilibrium solutions. In addition, the variation of the exploration strategies under the same learning rate also shows a variation in the win rate.

Dynamic agents that train on the same parameters converge to similar policies. At the beginning of the horizon length, they perform high replenishment together, while towards the end they decrease their replenishment amounts. The main reason for this is that they are exposed to high deterioration cost due to discarded products. Afterwards, they see lost sales with fewer orders placed, and they do high replenishment again for 1-2 periods and 0 replenishment in the last periods. In the game set with $m = 2$, $\lambda = 3$ and $T = 6$, it is observed that both agents trained with Epsilon Greedy - $\alpha = 0.7$ follow a replenishment policy $[4, 5, 0, 0, 4, 0]$ under the same seed. When the cost distributions of the agents trained with the same parameter are analyzed, it is observed that deterioration and shortage costs account for nearly 60% of the total cost.

When the competitive environment with agents trained with different parameter sets is analyzed, it is observed that the agents that converge faster than the other agent try to take the demand by replenishing more than the other agent by taking the deterioration cost over the entire horizon. In the game set with $m = 2$, $\lambda = 3$ and $T = 6$, it is observed that agent trained with Epsilon Greedy - $\alpha = 0.7$ follows a replenishment policy $[5, 5, 3, 5, 4, 0]$ whereas agent trained with Epsilon Greedy - $\alpha = 0.5$ follows a replenishment policy $[3, 4, 0, 4, 2, 0]$.

Therefore, agent that converge faster than the other agent in a competitive environment tries to defeat the other player by throwing away its replenishment and incurring the lost sales cost.

In the case where $\alpha = 0.7$ for both pharmacies, the pharmacy with the Epsilon-Greedy exploration strategy achieves a win-rate close to 57% on average. In the case where $\alpha = 0.5$ for both pharmacies, the pharmacy with the Greedy strategy wins 65% of the games on average. In particular, the pharmacy trained with the Epsilon Greedy - $\alpha = 0.7$ parameter pair achieved a better win rate than all other test parameters.

Table 4.10. Dynamic replenishment: win-rate (%) under $m = 2$ and $T = 10$.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	35.00	65.00	35.00	65.00	44.00	56.00
	Greedy - E. Greedy	47.00	53.00	43.00	57.00	58.00	42.00
	Epsilon-Greedy - E. Greedy	47.00	43.00	58.00	42.00	27.00	73.00
25K	Greedy - Greedy	54.00	46.00	18.00	82.00	52.00	48.00
	Greedy - E. Greedy	46.00	54.00	36.00	64.00	60.00	40.00
	Epsilon-Greedy - E. Greedy	24.00	76.00	58.00	42.00	75.00	25.00
45K	Greedy - Greedy	44.00	56.00	58.00	42.00	27.00	73.00
	Greedy - E. Greedy	52.00	48.00	73.00	27.00	67.00	33.00
	Epsilon-Greedy - E. Greedy	68.00	32.00	50.00	50.00	41.00	59.00
65K	Greedy - Greedy	54.00	46.00	60.00	16.00	88.00	12.00
	Greedy - E. Greedy	68.00	32.00	48.00	52.00	42.00	58.00
	Epsilon-Greedy - E. Greedy	58.00	42.00	52.00	48.00	58.00	42.00
85K	Greedy - Greedy	22.00	78.00	65.00	35.00	35.00	65.00
	Greedy - E. Greedy	86.00	14.00	65.00	35.00	52.00	48.00
	Epsilon-Greedy - E. Greedy	53.00	47.00	36.00	64.00	24.00	76.00
95K	Greedy - Greedy	64.00	36.00	80.00	20.00	68.00	32.00
	Greedy - E. Greedy	90.00	10.00	46.00	54.00	35.00	65.00
	Epsilon-Greedy - E. Greedy	42.00	58.00	28.00	72.00	48.00	52.00
105K	Greedy - Greedy	44.00	56.00	100.00	0.00	60.00	40.00
	Greedy - E. Greedy	77.00	23.00	54.00	46.00	28.00	72.00
	Epsilon-Greedy - E. Greedy	88.00	12.00	15.00	85.00	38.00	62.00
120K	Greedy - Greedy	32.00	68.00	98.00	2.00	55.00	45.00
	Greedy - E. Greedy	75.00	25.00	46.00	54.00	10.00	90.00
	Epsilon-Greedy - E. Greedy	54.00	46.00	8.00	92.00	38.00	62.00
125K	Greedy - Greedy	75.00	25.00	100.00	0.00	74.00	26.00
	Greedy - E. Greedy	92.00	8.00	36.00	64.00	12.00	88.00
	Epsilon-Greedy - E. Greedy	52.00	48.00	0.00	100.00	36.00	64.00
Avg	Greedy - Greedy	49.36	50.56	63.32	35.56	51.44	48.56
	Greedy - E. Greedy	65.04	34.88	46.80	53.12	42.04	57.76
	Epsilon-Greedy - E. Greedy	50.40	48.88	40.56	59.36	49.36	50.56

Table 4.10 shows the win rates of pharmacies trained with different exploration strategies and α in a competitive environment with $m = 2$ and $T = 10$.

Detailed win-rate information for pharmacies is shown in Table B.5 In this case, similar patterns are observed as in the case of outputs at horizon length $T = 6$. Pharmacies under the same parameter pairs perform similarly and reach equilibrium solutions in similar epochs. Under the same learning rate, different exploration strategies affect dominance in the environment. The parameter pairs $\alpha = 0.5$ with the Greedy strategy and $\alpha = 0.7$ with the Epsilon-Greedy strategy achieve the highest win-rate on average. In addition, in cases involving the Epsilon Greedy - $\alpha = 0.7$ parameter pair, the pharmacy wins a large percentage of games after converging to equilibrium solutions.

Table 4.11. Dynamic replenishment: win-rate (%) under $m = 3$ and $T = 6$.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	42.00	58.00	28.00	72.00	55.00	45.00
	Greedy - E. Greedy	48.00	52.00	56.00	44.00	51.00	49.00
	Epsilon-Greedy - E. Greedy	35.00	55.00	52.00	48.00	35.00	65.00
25K	Greedy - Greedy	83.00	17.00	14.00	86.00	60.00	40.00
	Greedy - E. Greedy	56.00	44.00	48.00	52.00	40.00	60.00
	Epsilon-Greedy - E. Greedy	67.00	33.00	57.00	43.00	58.00	42.00
45K	Greedy - Greedy	66.00	34.00	59.00	41.00	56.00	44.00
	Greedy - E. Greedy	56.00	44.00	62.00	38.00	86.00	14.00
	Epsilon-Greedy - E. Greedy	54.00	46.00	43.00	57.00	35.00	65.00
65K	Greedy - Greedy	56.00	44.00	63.00	13.00	46.00	54.00
	Greedy - E. Greedy	47.00	53.00	55.00	45.00	44.00	56.00
	Epsilon-Greedy - E. Greedy	56.00	44.00	43.00	57.00	45.00	55.00
85K	Greedy - Greedy	54.00	46.00	90.00	10.00	57.00	43.00
	Greedy - E. Greedy	72.00	28.00	38.00	62.00	41.00	59.00
	Epsilon-Greedy - E. Greedy	33.00	67.00	37.00	63.00	58.00	42.00
105K	Greedy - Greedy	48.00	52.00	90.00	10.00	52.00	48.00
	Greedy - E. Greedy	82.00	18.00	42.00	58.00	28.00	72.00
	Epsilon-Greedy - E. Greedy	26.00	74.00	4.00	96.00	45.00	55.00
115K	Greedy - Greedy	53.00	47.00	86.00	14.00	40.00	60.00
	Greedy - E. Greedy	80.00	18.00	22.00	76.00	13.00	82.00
	Epsilon-Greedy - E. Greedy	69.00	31.00	15.00	83.00	68.00	32.00
120K	Greedy - Greedy	60.00	40.00	90.00	10.00	46.00	54.00
	Greedy - E. Greedy	82.00	18.00	45.00	55.00	15.00	85.00
	Epsilon-Greedy - E. Greedy	82.00	18.00	8.00	92.00	28.00	72.00
125K	Greedy - Greedy	44.00	56.00	88.00	12.00	43.00	57.00
	Greedy - E. Greedy	88.00	12.00	35.00	65.00	2.00	98.00
	Epsilon-Greedy - E. Greedy	70.00	30.00	5.00	95.00	55.00	45.00
Avg	Greedy - Greedy	49.60	50.32	64.04	34.84	48.32	51.68
	Greedy - E. Greedy	61.96	37.96	48.24	51.68	42.08	57.72
	Epsilon-Greedy - E. Greedy	51.28	48.00	38.20	61.72	50.40	49.52

Table 4.11 shows the win rates obtained over the games played by pharmacies in the product lifetime $m = 3$ and horizon length $T = 6$ environment. When $\alpha = 0.5$, the pharmacy with Greedy strategy is observed to be more dominant than the pharmacy with Epsilon-Greedy, while at $\alpha = 0.7$, the dominance of Epsilon-Greedy is observed.

However, when the two cases are analyzed, with the same learning rate α , pharmacies with the Greedy strategy do not exhibit more dominant performance. On the other hand, pharmacies with the Greedy strategy achieve higher winning rates in earlier epochs. Pharmacies with the Epsilon-Greedy strategy, on the other hand, achieve high winning rates in slightly later epochs and exhibit a more dominant performance.

Table 4.12. Dynamic replenishment: win-rate (%) under $m = 3$ and $T = 10$.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	65.00	35.00	27.00	73.00	56.00	44.00
	Greedy - E. Greedy	38.00	62.00	27.00	73.00	56.00	44.00
	Epsilon-Greedy - E. Greedy	75.00	15.00	73.00	27.00	75.00	25.00
25K	Greedy - Greedy	32.00	68.00	52.00	48.00	60.00	40.00
	Greedy - E. Greedy	52.00	48.00	55.00	45.00	58.00	42.00
	Epsilon-Greedy - E. Greedy	58.00	42.00	56.00	44.00	59.00	41.00
45K	Greedy - Greedy	47.00	53.00	37.00	63.00	68.00	32.00
	Greedy - E. Greedy	60.00	40.00	44.00	56.00	66.00	34.00
	Epsilon-Greedy - E. Greedy	17.00	83.00	40.00	60.00	50.00	50.00
65K	Greedy - Greedy	50.00	50.00	65.00	11.00	18.00	82.00
	Greedy - E. Greedy	55.00	45.00	37.00	63.00	38.00	62.00
	Epsilon-Greedy - E. Greedy	60.00	40.00	48.00	52.00	49.00	51.00
85K	Greedy - Greedy	57.00	43.00	72.00	28.00	35.00	65.00
	Greedy - E. Greedy	82.00	18.00	27.00	73.00	49.00	51.00
	Epsilon-Greedy - E. Greedy	50.00	50.00	5.00	95.00	34.00	66.00
95K	Greedy - Greedy	52.00	48.00	65.00	35.00	32.00	68.00
	Greedy - E. Greedy	70.00	30.00	55.00	45.00	30.00	70.00
	Epsilon-Greedy - E. Greedy	43.00	57.00	16.00	84.00	55.00	45.00
105K	Greedy - Greedy	67.00	33.00	82.00	18.00	40.00	60.00
	Greedy - E. Greedy	88.00	12.00	62.00	38.00	22.00	78.00
	Epsilon-Greedy - E. Greedy	50.00	50.00	22.00	78.00	38.00	62.00
115K	Greedy - Greedy	70.00	30.00	92.00	8.00	20.00	80.00
	Greedy - E. Greedy	54.00	44.00	32.00	66.00	5.00	90.00
	Epsilon-Greedy - E. Greedy	28.00	72.00	18.00	80.00	27.00	73.00
120K	Greedy - Greedy	32.00	68.00	90.00	10.00	24.00	76.00
	Greedy - E. Greedy	80.00	20.00	30.00	70.00	15.00	85.00
	Epsilon-Greedy - E. Greedy	85.00	15.00	8.00	92.00	58.00	42.00
125K	Greedy - Greedy	18.00	82.00	88.00	12.00	38.00	62.00
	Greedy - E. Greedy	85.00	15.00	18.00	82.00	20.00	80.00
	Epsilon-Greedy - E. Greedy	57.00	43.00	0.00	100.00	50.00	50.00
Avg	Greedy - Greedy	48.72	51.20	63.36	35.52	47.64	52.36
	Greedy - E. Greedy	61.20	38.72	46.92	53.00	40.08	59.72
	Epsilon-Greedy - E. Greedy	49.56	49.72	35.28	64.64	49.12	50.80

Table 4.12 shows the win rates obtained in the games played by pharmacies in the product lifetime $m = 3$ and the horizon length $T = 10$ environment and the detailed win-rates are illustrated in Table B.7. When test cases are analyzed, strategy or α selection does not determine dominance, while in the $m = 3$ world, pairs affect the win rate. For example, in cases where both pharmacies have learning rate $\alpha = 0.5$, pharmacies that choose the Greedy strategy achieve an average win rate of almost 62%.

A similar situation is observed with $\alpha = 0.7$ and the Epsilon Greedy strategy. In the previous product lifetime and horizon length cases, Epsilon Greedy - $\alpha = 0.7$ shows much more dominant results, while in $m = 3$ and $T = 6$, Greedy - $\alpha = 0.5$ observes close win rates since the state space is larger in the $m = 3$ case, there are more policies to explore, and the Greedy exploration strategy is a better parameter for this need.

4.4. Summary

In this chapter, we analyze the computational outputs of two Nash Q-Learning algorithms developed to solve the perishable inventory control problem in a competitive environment with different test parameters. So far in the literature, researchers have tried to obtain exact solutions to problems that require sequential decision-making, such as the perishable inventory control problem, by approaching them with the Dynamic Programming (DP) technique [3, 30]. In this context, we evaluate the performance of our algorithm by comparing it with DP-based equilibrium solutions in two different cases, fixed replenishment and dynamic replenishment. In addition, algorithm scaling is also tested by running the algorithms with large input sets.

In the fixed replenishment case, tests are performed using the Algorithm 3.5 developed in Section 3. In order to evaluate the solution quality of the algorithm, it is compared with the DP-Based equilibrium solutions obtained at product lifetime $m = 2$. As shown in Table 4.3 and Figure 4.2, the Nash Q-Learning algorithm converges to equilibrium solutions after 100K epochs.

The training times of the corresponding epoch level in Table 4.2 are also considerably shorter than the DP-based equilibrium solutions. Tables 4.4 and Table 4.5 show the winning rates obtained by the replenishing pharmacy with different parameters according to the Nash Q-Learning algorithm and the superiority of the pharmacy increases in the following epochs. In particular, the dominance of pharmacies trained with Greedy - $\alpha = 0.5$ and Epsilon Greedy - $\alpha = 0.7$ parameter pairs is stronger and more consistent.

In the dynamic replenishment case, the performance of the Nash Q-Learning algorithm is evaluated together with the test cases using the Algorithm 3.6 developed in Section 3. In order to analyze and report the performance and convergence of the developed algorithm, at product lifetime $m = 2$, comparisons are made with the equilibrium solutions of the Markov Game in Section 3.7. Tables 4.7 and Table 4.8 show the differences of the Nash Q-learning algorithm with the equilibrium solutions, and it can be seen that both pharmacies in the competitive environment at two different horizon lengths converge to equilibrium solutions after 100K epochs. In addition, we observe that pharmacies with the Epsilon Greedy - $\alpha = 0.7$ parameter pair converge faster. As can be seen in Figure 4.4 and Figure 4.5, the cost gap graphs of both pharmacies converge to equilibrium solutions after 100K epochs and using the training times in Table 4.6, it can be said that the solution convergence speed of the Nash Q-Learning algorithm is almost two times faster than the Markov Game solving algorithm for a small problem (e.g., $m = 2$).

When we examine the win-rate tables in the dynamic replenishment case, we do not observe any dominance of pharmacies trained with the same parameters against each other in 100-game test environments; pharmacies achieve similar win rates. However, different win rate values are observed with different exploration strategies under the same learning rate. Pharmacies trained with Greedy - $\alpha = 0.5$ and Epsilon Greedy - $\alpha = 0.7$ parameter pairs dominate the other pharmacies in their environment and achieve a higher win rate. In addition, when these two pairs of parameters are tested in the same game, the pharmacy with the parameter Epsilon-Greedy - $\alpha = 0.7$ partially achieves a higher win-rate. When the Nash Q-Learning algorithms developed for both cases are analyzed with different test parameters, it is observed that they converge to DP-based equilibrium solutions in a shorter time than the runtime of the algorithm that obtains these equilibrium solutions. In addition, with other analyses, it is observed that in the fixed replenishment case, the replenishment policy set by the pharmacy trained with Nash Q-Learning dominates the other pharmacies in the environment. It is observed that parameter selection affects the convergence and dominance of the algorithm in the game.

5. CONCLUSION

In this thesis, our objective is to solve the perishable inventory control in a competitive environment using the Reinforcement Learning (RL) technique. For this purpose, we study the problem of perishable inventory control in a competitive environment under two concepts. In the fixed replenishment case, we consider a setting where a player in the competitive environment experiences different replenishment actions in the action space and the other players make a fixed replenishment equal to the average of the random demand. The dynamic replenishment case is analyzed by allowing all players in the competitive environment to access all replenishment actions in the action space.

In the single-agent perishable inventory control problem, a single agent learns to take the best action in a state period through a self-play with the environment via an action-reward mechanism. In Multi-agent Reinforcement Learning (MARL), each agent in a competitive environment learns to make the best response to other agents given the joint inventory state. In this context, the first step to formulate the problem is to define the mathematical notation of the competitive environment and the players interacting with it. We propose joint inventory states in the environment, how demand is allocated to players according to the fulfillment metric in the relevant period, the reward mechanism, and the state transition functions while moving to the next period.

For the fixed replenishment case, we design an environment in which one pharmacy can replenish dynamically, while the other pharmacy uses fixed replenishment. With the Nash-Q Learning algorithm that we develop, we train the pharmacy that performs dynamic replenishment and observe whether it has superiority with the replenishment policies it determines against the other pharmacy in the same environment that performs fixed replenishment. We evaluate the algorithm performance tests on solution convergence and the win-rate of the pharmacy.

As a benchmark, we design an algorithm that obtains DP-based equilibrium solutions. We observe whether the solutions of the pharmacy trained with the Nash Q-Learning algorithm converge to DP-based equilibrium solutions. After 100K epochs, we observe that the solutions of the Nash Q-Learning algorithm converge to DP-based equilibrium solutions and this convergence time is considerably shorter than the runtime of the DP-based equilibrium algorithm. In addition, it is observed that pharmacies trained with Greedy - α and Epsilon Greedy - $\alpha = 0.7$ Exploration strategy - Learning rate parameter pairs dominate by achieving a much higher win-rate than the other pharmacy in the competitive environment.

In the dynamic replenishment case, an environment is prepared where all pharmacies in the competitive environment are allowed to experience different replenishment actions and compete with each other. All pharmacies are put through a training process using the developed Nash-Q Learning algorithm. For algorithm tests, an algorithm that solves the Markov Game and obtains equilibrium solutions is implemented as a benchmark. As a result of the tests, the Nash Q-Learning algorithm converges to the Markov Game solving algorithm almost twice as fast as the runtime of this algorithm. It is observed that pharmacies that have trained under the same parameters cannot establish dominance against each other and make decisions according to similar replenishment policies, while the win-rate received by pharmacies that train with different exploration strategies under the same learning rate is different.

We obtain replenishment policies for pharmacies in competitive environments using the Nash Q-Learning algorithms that we implemented for both cases. Considering the convergence of the obtained solutions to the equilibrium solutions, it is also observed that the algorithms are computationally efficient. Using Nash Q-Learning algorithms and formulations, the perishable inventory control problem in an N-player competitive environment can be solved and the equilibrium replenishment policies of players trained with different strategies can be obtained. We contribute to the literature by analyzing MARL and the perishable inventory control problem with the solution approach that we have created by combining the two contexts.

The computational outputs of this thesis show that players' replenishment actions with respect to other players and the joint inventory state, the age distribution of the products in their inventories, and their demand allocation with respect to the fulfillment metric are important for their dominance in a competitive environment.

Finally, there are also sections that can take the work done in this thesis further as future work. A more detailed analysis can be made by defining the common demand in the competitive environment as correlated over time, and the replenishment policies that pharmacies will learn under this correlation can also be observed. In addition, within the scope of this thesis, the entire demand is allocated to the demand assignee determined according to the fulfillment metric. By developing the fulfillment metric, this demand allocation can be designed to be partial.

REFERENCES

1. Chen, X., Z. Pang and L. Pan, “Coordinating Inventory Control and Pricing Strategies for Perishable Products”, *Operations Research*, Vol. 62, No. 2, pp. 284–300, 2014.
2. Nahmias, S., “Perishable Inventory Theory: a Review”, *Operations Research*, Vol. 30, No. 4, pp. 680–708, 1982.
3. Pierskalla, W. P. and C. D. Roach, “Optimal Issuing Policies for Perishable Inventory”, *Management Science*, Vol. 18, No. 11, pp. 603–614, 1972.
4. Chao, X., X. Gong, C. Shi and H. Zhang, “Approximation Algorithms for Perishable Inventory Systems”, *Operations Research*, Vol. 63, No. 3, pp. 585–601, 2015.
5. Watkins, C. J. and P. Dayan, “Q-learning”, *Machine learning*, Vol. 8, No. 3, pp. 279–292, 1992.
6. Tesauro, G., “TD-Gammon, a Self-teaching Backgammon Program, Achieves Master-level Play”, *Neural Computation*, Vol. 6, No. 2, pp. 215–219, 1994.
7. Tesauro, G., “Temporal Difference Learning and TD-Gammon”, *Communications of Association for Computing Machinery*, Vol. 38, No. 3, p. 58–68, 1995.
8. Hu, J. and M. P. Wellman, “Nash Q-learning for General-sum Stochastic Games”, *Journal of Machine Learning Research*, Vol. 4, No. 11, pp. 1039–1069, 2003.
9. 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.
10. Fries, B. E., “Optimal Ordering Policy for a Perishable Commodity with Fixed

- Lifetime”, *Operations Research*, Vol. 23, No. 1, pp. 46–61, 1975.
11. 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.
 12. 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.
 13. Deniz, B., I. Karaesmen and A. Scheller-Wolf, “Managing Perishables with Substitution: Inventory Issuance and Replenishment Heuristics”, *Manufacturing and Service Operations Management*, Vol. 12, No. 2, pp. 319–329, 2010.
 14. Sutton, R. S., “Learning to Predict by the Methods of Temporal Differences”, *Machine Learning*, Vol. 3, No. 1, pp. 9–44, 1988.
 15. Gosavi, A., “Reinforcement Learning: a Tutorial Survey and Recent Advances”, *INFORMS Journal on Computing*, Vol. 21, No. 2, pp. 178–192, 2009.
 16. Bertsekas, D., *A Course in Reinforcement Learning*, Athena Scientific, Belmont, 2023.
 17. Sutton, R. S. and A. G. Barto, *Reinforcement Learning - An Introduction*, The MIT Press, Cambridge, 2018.
 18. Temizoz, T., C. Imdahl, R. Dijkman, D. Lamghari-Idrissi and W. van Jaarsveld, “Deep Controlled Learning for Inventory Control”, arXiv preprint arXiv:2011.15122, 2020.
 19. Littman, M. L., “Markov Games as a Framework for Multi-agent Reinforcement Learning”, *Machine Learning Proceedings*, Vol. 157, p. 157–163, New Brunswick, 1994.

20. Shoham, Y., R. Powers and T. Grenager, *Multi-agent Reinforcement Learning: a Critical Survey*, Tech. rep., Citeseer, 2003.
21. Chang, Y.-H. and L. P. Kaelbling, “Playing is Believing: the Role of Beliefs in Multi-agent Learning”, *Advances in Neural Information Processing Systems*, Vol. 14, 2001.
22. 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, Montreal, 2007.
23. Lanctot, M., V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Pérolat, D. Silver and T. Graepel, “A Unified Game-theoretic Approach to Multiagent Reinforcement Learning”, *Proceedings of the 31st International Conference on Neural Information Processing Systems*, Vol. 157, p. 4193–4206, California, 2017.
24. Nowe, A., P. Vrancx and Y.-M. De Hauwere, *Game Theory and Multi-agent Reinforcement Learning*, Springer, Berlin, 2012.
25. Hu, J. and M. P. Wellman, “Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm”, *International Conference on Machine Learning*, Vol. 98, pp. 242–250, Madison, 1998.
26. Bowling, M., “Convergence Problems of General-sum Multiagent Reinforcement Learning”, *International Conference on Machine Learning*, p. 89–94, Stanford, 2000.
27. Harsanyi, J. C. and R. Selten, *A General Theory of Equilibrium Selection in Games*, The MIT Press, Cambridge, 1988.
28. Littman, M. L., “Friend-or-foe Q-learning in General-sum Games”, *International Conference on Machine Learning*, Vol. 1, p. 322–328, Williamstown, 2001.

29. Nash, J., “Non-Cooperative Games”, *Annals of Mathematics*, Vol. 54, No. 2, pp. 286–295, 1951.
30. Nahmias, S., *Perishable Inventory Systems*, Springer Science & Business Media, New York, 2011.
31. Chazan, D. and S. Gal, “A Markovian Model for a Perishable Product Inventory”, *Management Science*, Vol. 23, No. 5, pp. 512–521, 1977.



APPENDIX A: DETAILED COMPUTATIONAL RESULTS ON THE NASH Q-LEARNING UNDER FIXED REPLENISHMENT

Detailed information on the test results related to the fixed replenishment case is provided in the tables below.

Table A.1. Fixed replenishment: detailed training runtime (hours).

E	m = 2		m = 3	
	T = 6	T = 10	T = 6	T = 10
5K	0.086	0.1237	0.3958	0.6401
10K	0.166	0.2469	0.7885	2.8340
15K	0.246	0.3707	2.7155	3.5178
20K	0.325	0.4942	3.2728	4.5246
25K	0.404	0.6179	3.6892	6.0935
30K	0.483	0.7419	4.5917	7.6623
35K	0.562	0.8657	8.7459	9.2312
40K	0.642	0.9896	9.1199	10.8
45K	0.721	1.1137	9.4930	12.3688
50K	0.800	1.2379	9.8649	13.9377
55K	0.881	1.3634	11.7843	15.5065
60K	0.960	1.4888	12.9691	17.0754
65K	1.039	1.6141	14.1540	18.6442
70K	1.119	1.7394	15.3388	20.2130
75K	1.199	1.8648	16.5236	21.7819
80K	1.279	1.9901	17.7085	23.3507
85K	1.358	2.1154	18.8933	24.9196
90K	1.438	2.2408	20.0781	26.4884

Table A.1. Fixed replenishment: detailed training runtime (hours) (cont.).

E	m = 2		m = 3	
	T = 6	T = 10	T = 6	T = 10
95K	1.519	2.3661	21.2630	28.0572
100K	1.599	2.4914	22.4478	29.6261
105K	1.680	2.6168	23.6326	31.1949
110K	1.761	2.7421	24.8175	32.7638
115K	1.840	2.8674	26.0023	34.3326
120K	1.920	2.9928	27.1871	35.9014
125K	2.000	3.1181	28.3719	37.4703
130K	2.079	3.2434	-	-
135K	2.158	3.3688	-	-
140K	2.240	3.4941	-	-
145K	2.320	3.6194	-	-
150K	2.401	3.7448	-	-

Table A.2. Fixed replenishment: detailed cost gap (%) to dp-based equilibrium.

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
5K	Greedy	53.7	56.2	54.2	35.9
	Epsilon-Greedy	39.8	49.7	34.3	48.7
10K	Greedy	59.4	33.6	46.7	52.2
	Epsilon-Greedy	57.2	42.4	45.5	41.3
15K	Greedy	28.1	51.9	51.8	47.7
	Epsilon-Greedy	52.3	36.6	46.9	39.9
20K	Greedy	55.1	53.7	30.2	36.9
	Epsilon-Greedy	13.3	28.6	47.0	31.8
25K	Greedy	38.2	12.2	40.1	32.5
	Epsilon-Greedy	25.3	29.9	34.6	36.2

Table A.2. Fixed replenishment: detailed cost gap (%) to dp-based equilibrium
(cont.).

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
30K	Greedy	34.3	48.1	39.4	37.6
	Epsilon-Greedy	19.6	24.8	35.8	25.8
35K	Greedy	27.9	8.2	28.8	32.9
	Epsilon-Greedy	1.7	25.8	44.8	21.6
40K	Greedy	36.0	2.3	30.3	27.4
	Epsilon-Greedy	56.4	24.2	43.4	27.4
45K	Greedy	19.2	7.5	29.3	25.1
	Epsilon-Greedy	47.5	19.9	29.7	24.3
50K	Greedy	19.9	36.9	38.8	21.9
	Epsilon-Greedy	46.9	17.5	20.3	27.7
55K	Greedy	19.4	17.6	34.8	26.3
	Epsilon-Greedy	80.2	14.6	28.9	21.2
60K	Greedy	17.4	9.9	37.8	28.3
	Epsilon-Greedy	54.5	8.4	21.9	14.8
65K	Greedy	14.9	46.6	32.7	24.8
	Epsilon-Greedy	43.1	9.6	29.6	16.0
70K	Greedy	24.7	19.7	18.8	21.9
	Epsilon-Greedy	22.8	9.7	36.4	17.4
75K	Greedy	17.5	19.5	13.3	26.7
	Epsilon-Greedy	5.4	12.5	19.6	10.2
80K	Greedy	16.9	4.8	17.8	16.9
	Epsilon-Greedy	5.6	13.6	17.5	11.4

Table A.2. Fixed replenishment: detailed cost gap (%) to dp-based equilibrium
(cont.).

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
85K	Greedy	16.8	13.6	22.1	21.3
	Epsilon-Greedy	7.6	15.8	16.9	21.9
90K	Greedy	14.0	14.3	26.0	17.2
	Epsilon-Greedy	6.9	6.1	18.3	13.6
95K	Greedy	13.6	11.3	18.4	14.7
	Epsilon-Greedy	11.4	16.2	18.7	9.5
100K	Greedy	11.3	17.2	17.9	15.2
	Epsilon-Greedy	11.2	2.1	11.5	5.7
105K	Greedy	6.2	18.3	15.8	10.6
	Epsilon-Greedy	10.2	4.8	17.7	3.6
110K	Greedy	6.4	15.4	13.5	15.4
	Epsilon-Greedy	14.4	4.2	16.4	3.6
115K	Greedy	8.2	13.0	9.6	13.3
	Epsilon-Greedy	15.6	5.6	12.2	3.4
120K	Greedy	6.5	7.7	8.0	14.0
	Epsilon-Greedy	6.8	0.7	13.6	4.2
125K	Greedy	4.6	9.0	11.2	7.8
	Epsilon-Greedy	5.3	1.9	12.1	6.1
130K	Greedy	4.0	8.1	9.9	9.9
	Epsilon-Greedy	6.5	3.7	9.1	3.1

Table A.2. Fixed replenishment: detailed cost gap (%) to dp-based equilibrium
(cont.).

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
135K	Greedy	3.8	8.9	5.6	4.8
	Epsilon-Greedy	4.8	2.1	8.9	4.9
140K	Greedy	4.0	2.2	8.5	4.4
	Epsilon-Greedy	6.4	0.6	9.4	2.9
145K	Greedy	2.8	4.9	7.0	5.4
	Epsilon-Greedy	6.0	0.7	4.6	3.0
150K	Greedy	3.6	2.6	5.2	4.3
	Epsilon-Greedy	9.2	0.6	9.6	4.5
Average	Greedy	19.62	19.17	24.11	21.78
	Epsilon-Greedy	23.13	14.43	23.85	16.86

Table A.3. Fixed replenishment: detailed win-rate (%) under $m = 2$.

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
5K	Greedy	40.0	35.0	0.0	0.0
	Epsilon-Greedy	46.0	46.0	100.0	34.0
10K	Greedy	38.0	32.0	48.0	88.0
	Epsilon-Greedy	44.0	55.0	50.0	76.0
15K	Greedy	85.0	30.0	46.0	84.0
	Epsilon-Greedy	16.0	72.0	100.0	58.0
20K	Greedy	48.0	44.0	100.0	80.0
	Epsilon-Greedy	66.0	48.0	56.0	80.0
25K	Greedy	60.0	58.0	66.0	46.0
	Epsilon-Greedy	36.0	54.0	58.0	0.0
30K	Greedy	72.0	54.0	100.0	58.0
	Epsilon-Greedy	48.0	88.0	26.0	100.0
35K	Greedy	82.0	82.0	100.0	62.0
	Epsilon-Greedy	68.0	86.0	48.0	80.0
40K	Greedy	64.0	100.0	100.0	62.0
	Epsilon-Greedy	32.0	100.0	92.0	100.0
45K	Greedy	78.0	74.0	100.0	74.0
	Epsilon-Greedy	54.0	45.0	20.0	100.0

Table A.3. Fixed replenishment: detailed win-rate (%) under $m = 2$ (cont.).

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
50K	Greedy	86.0	65.0	100.0	70.0
	Epsilon-Greedy	66.0	42.0	78.0	58.0
55K	Greedy	44.0	72.0	100.0	84.0
	Epsilon-Greedy	56.0	100.0	38.0	80.0
60K	Greedy	82.0	100.0	100.0	100.0
	Epsilon-Greedy	76.0	88.0	100.0	70.0
65K	Greedy	32.0	64.0	100.0	80.0
	Epsilon-Greedy	50.0	52.0	100.0	100.0
70K	Greedy	32.0	100.0	100.0	62.0
	Epsilon-Greedy	78.0	80.0	50.0	14.0
75K	Greedy	48.0	100.0	100.0	98.0
	Epsilon-Greedy	48.0	92.0	54.0	20.0
80K	Greedy	46.0	100.0	100.0	100.0
	Epsilon-Greedy	100.0	36.0	46.0	82.0
85K	Greedy	40.0	100.0	100.0	100.0
	Epsilon-Greedy	100.0	74.0	94.0	28.0
90K	Greedy	40.0	100.0	100.0	50.0
	Epsilon-Greedy	100.0	88.0	0.0	100.0
95K	Greedy	50.0	52.0	100.0	76.0
	Epsilon-Greedy	100.0	44.0	54.0	100.0
100K	Greedy	98.0	100.0	100.0	68.0
	Epsilon-Greedy	28.0	92.0	0.0	74.0

Table A.3. Fixed replenishment: detailed win-rate (%) under $m = 2$ (cont.).

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
105K	Greedy	44.0	84.0	100.0	56.0
	Epsilon-Greedy	100.0	44.0	84.0	68.0
110K	Greedy	98.0	76.0	64.0	32.0
	Epsilon-Greedy	60.0	24.0	58.0	64.0
115K	Greedy	92.0	92.0	100.0	26.0
	Epsilon-Greedy	82.0	100.0	72.0	60.0
120K	Greedy	90.0	76.0	100.0	100.0
	Epsilon-Greedy	90.0	100.0	100.0	0.0
125K	Greedy	86.0	44.0	100.0	20.0
	Epsilon-Greedy	60.0	100.0	100.0	100.0
130K	Greedy	92.0	94.0	100.0	60.0
	Epsilon-Greedy	80.0	100.0	44.0	100.0
135K	Greedy	96.0	96.0	100.0	72.0
	Epsilon-Greedy	72.0	100.0	52.0	100.0
140K	Greedy	92.0	46.0	60.0	100.0
	Epsilon-Greedy	100.0	100.0	48.0	60.0
145K	Greedy	95.0	62.0	100.0	36.0
	Epsilon-Greedy	92.0	100.0	80.0	100.0
150K	Greedy	94.0	78.0	60.0	88.0
	Epsilon-Greedy	86.0	100.0	92.0	100.0
Average	Greedy	68.13	73.67	88.13	67.73
	Epsilon-Greedy	67.80	75.00	63.13	70.20

Table A.4. Fixed replenishment: detailed win-rate (%) under $m = 3$.

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
5K	Greedy	48	64	15	24
	Epsilon-Greedy	45	30	45	56
10K	Greedy	34	45	28	60
	Epsilon-Greedy	52	48	54	16
15K	Greedy	60	64	23	44
	Epsilon-Greedy	84	36	72	72
20K	Greedy	76	80	30	24
	Epsilon-Greedy	44	28	44	44
25K	Greedy	72	68	45	16
	Epsilon-Greedy	56	45	60	100
30K	Greedy	100	40	32	36
	Epsilon-Greedy	80	40	40	40
35K	Greedy	84	100	40	32
	Epsilon-Greedy	100	44	100	100
40K	Greedy	84	58	42	100
	Epsilon-Greedy	100	80	52	52
45K	Greedy	96	65	38	28
	Epsilon-Greedy	100	60	56	56
50K	Greedy	76	60	50	44
	Epsilon-Greedy	95	56	72	72
55K	Greedy	84	75	56	48
	Epsilon-Greedy	82	68	68	68
60K	Greedy	84	48	50	20
	Epsilon-Greedy	100	72	54	56

Table A.4. Fixed replenishment: detailed win-rate (%) under $m = 3$ (cont.).

E	Exploration Strategy	T = 6		T = 10	
		$\alpha = 0.5$	$\alpha = 0.7$	$\alpha = 0.5$	$\alpha = 0.7$
65K	Greedy	100	68	28	76
	Epsilon-Greedy	52	60	67	28
70K	Greedy	56	52	85	92
	Epsilon-Greedy	48	75	96	100
75K	Greedy	72	50	77	48
	Epsilon-Greedy	60	75	88	40
85K	Greedy	80	24	70	20
	Epsilon-Greedy	64	65	78	60
90K	Greedy	84	52	70	24
	Epsilon-Greedy	70	85	64	64
95K	Greedy	76	68	100	20
	Epsilon-Greedy	48	100	75	28
100K	Greedy	100	52	88	88
	Epsilon-Greedy	100	100	92	75
105K	Greedy	100	70	84	84
	Epsilon-Greedy	56	100	88	52
110K	Greedy	36	78	80	80
	Epsilon-Greedy	92	92	80	56
115K	Greedy	100	95	100	90
	Epsilon-Greedy	100	92	85	75
120K	Greedy	56	82	92	92
	Epsilon-Greedy	100	88	100	100
125K	Greedy	56	68	96	83
	Epsilon-Greedy	100	100	95	100
Average	Greedy	76.56	64.04	59.92	52.68
	Epsilon-Greedy	77.70	64.04	61.72	64.40

APPENDIX B: DETAILED COMPUTATIONAL RESULTS ON THE NASH Q-LEARNING UNDER DYNAMIC REPLENISHMENT

Detailed information on the test results related to the dynamic replenishment case is provided in the tables below.

Table B.1. Dynamic replenishment: detailed training runtime (hours).

N	$m = 2$		$m = 3$	
	T = 6	T = 10	T = 6	T = 10
5K	0.45	0.64	2.05	3.31
10K	0.78	1.16	3.72	13.36
15K	0.96	1.45	10.59	13.72
20K	1.14	1.73	11.48	15.87
25K	1.33	2.03	12.13	20.03
30K	1.53	2.35	14.54	24.26
35K	1.88	2.90	29.29	30.92
40K	2.21	3.41	31.47	37.27
45K	2.38	3.68	31.38	40.89
50K	2.55	3.94	31.42	44.40
55K	2.71	4.20	36.33	47.80
60K	2.88	4.47	38.92	51.24
65K	3.05	4.73	41.50	54.66
70K	3.21	4.99	44.05	58.04
75K	3.38	5.25	46.56	61.38
80K	3.54	5.51	49.05	64.67
85K	3.71	5.77	51.54	67.98
90K	3.87	6.03	54.04	71.29
95K	4.04	6.29	56.52	74.59

Table B.1. Dynamic replenishment: detailed training runtime (hours) (cont.).

N	$m = 2$		$m = 3$	
	T = 6	T = 10	T = 6	T = 10
100K	4.20	6.55	58.98	77.84
105K	4.37	6.80	61.46	81.12
110K	4.74	7.38	66.75	88.12
115K	5.12	7.98	72.34	95.52
120K	5.50	8.58	77.91	102.89
125K	5.89	9.18	83.55	110.35

Table B.2. Dynamic replenishment ($m = 2, T = 6$): detailed cost gap (%) to equilibrium solution.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	30.94	33.14	30.98	44.03	39.36	32.44
	Greedy - E. Greedy	37.36	44.09	29.42	28.02	29.36	28.09
	Epsilon-Greedy - E. Greedy	46.09	53.17	49.36	38.09	41.53	42.35
10K	Greedy - Greedy	31.73	28.00	22.85	38.72	27.92	34.86
	Greedy - E. Greedy	30.83	38.00	27.91	28.14	29.83	18.00
	Epsilon-Greedy - E. Greedy	50.08	55.18	50.83	42.00	36.84	35.35
15K	Greedy - Greedy	17.36	15.40	16.59	26.62	21.32	26.20
	Greedy - E. Greedy	27.62	35.48	27.62	15.48	27.62	15.48
	Epsilon-Greedy - E. Greedy	41.80	42.87	47.23	35.65	32.98	35.36
20K	Greedy - Greedy	14.70	17.94	17.56	23.05	21.11	21.81
	Greedy - E. Greedy	25.38	36.51	25.00	16.72	15.38	16.51
	Epsilon-Greedy - E. Greedy	31.76	35.51	45.38	36.51	21.40	21.32
25K	Greedy - Greedy	17.17	10.41	13.18	26.67	15.06	16.06
	Greedy - E. Greedy	20.05	28.22	20.05	18.22	20.05	18.22
	Epsilon-Greedy - E. Greedy	34.12	38.79	40.63	28.22	19.83	13.34
30K	Greedy - Greedy	18.17	12.38	15.90	22.83	19.60	21.89
	Greedy - E. Greedy	19.34	33.68	9.88	13.13	19.34	13.68
	Epsilon-Greedy - E. Greedy	30.73	33.76	39.34	23.68	13.82	13.36
35K	Greedy - Greedy	13.86	18.48	12.13	22.01	11.24	15.15
	Greedy - E. Greedy	15.93	23.89	16.44	13.02	19.96	7.89
	Epsilon-Greedy - E. Greedy	24.29	28.17	39.96	23.89	15.89	19.82
40K	Greedy - Greedy	9.58	15.96	9.21	17.22	10.77	11.20
	Greedy - E. Greedy	12.25	26.11	16.35	16.28	16.25	7.11
	Epsilon-Greedy - E. Greedy	22.29	28.47	36.07	16.11	13.38	12.28

Table B.2. Dynamic replenishment ($m = 2, T = 6$): detailed cost gap (%) to equilibrium solution (cont.).

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
45K	Greedy - Greedy	5.39	9.55	4.25	14.95	12.45	12.58
	Greedy - E. Greedy	14.20	22.33	14.10	12.58	14.20	6.33
	Epsilon-Greedy - E. Greedy	28.74	25.23	34.20	12.33	8.25	9.60
50K	Greedy - Greedy	6.48	2.35	5.36	13.43	17.03	16.30
	Greedy - E. Greedy	14.46	21.46	12.58	9.49	20.46	4.46
	Epsilon-Greedy - E. Greedy	26.28	20.22	24.46	17.46	13.32	8.76
55K	Greedy - Greedy	8.60	5.26	5.06	15.19	11.25	9.97
	Greedy - E. Greedy	10.28	22.56	10.28	5.56	20.28	5.56
	Epsilon-Greedy - E. Greedy	22.85	25.03	20.28	15.56	11.11	8.60
60K	Greedy - Greedy	6.14	6.43	3.47	17.24	15.33	12.63
	Greedy - E. Greedy	6.42	18.39	6.22	4.65	16.42	4.39
	Epsilon-Greedy - E. Greedy	14.60	23.00	26.48	14.39	7.73	6.06
65K	Greedy - Greedy	7.32	2.79	3.27	10.72	8.50	10.74
	Greedy - E. Greedy	8.92	15.48	8.05	2.13	18.92	2.48
	Epsilon-Greedy - E. Greedy	16.80	25.69	28.92	5.48	4.14	5.86
70K	Greedy - Greedy	2.91	3.65	2.37	12.21	6.89	13.23
	Greedy - E. Greedy	8.74	12.36	2.12	3.25	12.74	2.36
	Epsilon-Greedy - E. Greedy	17.07	21.62	22.74	12.36	4.03	1.42
75K	Greedy - Greedy	2.61	1.60	2.46	10.42	8.32	9.55
	Greedy - E. Greedy	6.25	14.39	6.48	2.33	16.25	2.39
	Epsilon-Greedy - E. Greedy	25.11	22.22	20.25	8.39	3.56	4.03
80K	Greedy - Greedy	1.58	2.51	3.51	14.88	5.12	6.02
	Greedy - E. Greedy	4.54	16.02	5.51	3.23	12.54	3.02
	Epsilon-Greedy - E. Greedy	22.43	23.81	21.54	6.02	5.22	5.30
85K	Greedy - Greedy	1.90	2.18	4.76	15.83	5.97	7.45
	Greedy - E. Greedy	5.26	13.68	5.78	3.18	5.26	3.68
	Epsilon-Greedy - E. Greedy	22.50	24.05	20.26	3.68	3.73	5.55
90K	Greedy - Greedy	2.18	3.13	2.90	6.66	6.21	10.95
	Greedy - E. Greedy	3.67	9.34	3.15	1.18	6.67	1.34
	Epsilon-Greedy - E. Greedy	22.87	21.81	16.67	1.34	2.94	4.91
95K	Greedy - Greedy	2.39	1.65	1.57	3.38	6.05	10.40
	Greedy - E. Greedy	3.24	8.09	1.29	3.52	6.24	1.09
	Epsilon-Greedy - E. Greedy	14.56	19.11	16.24	3.09	6.44	1.82
100K	Greedy - Greedy	1.96	1.00	1.54	4.09	5.70	8.74
	Greedy - E. Greedy	2.88	8.38	1.48	0.24	8.88	0.38
	Epsilon-Greedy - E. Greedy	11.04	17.34	18.88	1.38	6.64	2.61
105K	Greedy - Greedy	0.76	2.32	0.18	4.60	9.78	7.42
	Greedy - E. Greedy	1.35	6.73	0.35	0.73	7.35	0.73
	Epsilon-Greedy - E. Greedy	5.12	13.83	17.47	1.41	4.47	0.83
110K	Greedy - Greedy	0.63	0.53	0.82	4.65	9.54	4.03
	Greedy - E. Greedy	1.19	4.28	0.19	0.28	3.19	0.28
	Epsilon-Greedy - E. Greedy	8.11	9.61	11.59	2.27	0.85	0.29
115K	Greedy - Greedy	0.78	0.33	0.78	2.89	7.77	7.31
	Greedy - E. Greedy	0.06	6.13	0.06	0.13	8.06	0.13
	Epsilon-Greedy - E. Greedy	8.46	4.65	8.44	1.18	0.63	0.19
120K	Greedy - Greedy	0.44	0.33	0.92	2.93	6.56	6.26
	Greedy - E. Greedy	0.55	4.37	0.17	0.26	5.55	0.37
	Epsilon-Greedy - E. Greedy	9.25	7.12	5.55	0.37	0.22	0.50
125K	Greedy - Greedy	0.71	0.53	0.61	2.45	7.08	6.07
	Greedy - E. Greedy	0.80	2.16	0.40	0.24	5.80	0.16
	Epsilon-Greedy - E. Greedy	6.51	9.39	5.80	0.16	0.47	0.20
Avg	Greedy - Greedy	8.25	7.91	7.29	15.11	12.64	13.57
	Greedy - E. Greedy	11.26	18.89	10.15	8.13	14.67	6.57
	Epsilon-Greedy - E. Greedy	22.54	25.19	26.71	14.05	11.18	10.39

Table B.3. Dynamic replenishment ($m = 2, T = 10$): detailed cost gap (%) to equilibrium solution.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	32.45	35.13	32.01	45.78	40.12	33.49
	Greedy - E. Greedy	38.69	45.00	30.41	28.85	29.91	29.79
	Epsilon-Greedy - E. Greedy	47.37	54.86	51.01	39.70	43.06	43.77
10K	Greedy - Greedy	33.25	29.13	24.78	39.46	29.11	36.75
	Greedy - E. Greedy	31.85	39.13	29.41	29.36	31.11	19.79
	Epsilon-Greedy - E. Greedy	50.59	55.78	52.52	42.87	38.60	36.71
15K	Greedy - Greedy	18.28	16.63	17.47	28.61	22.77	26.88
	Greedy - E. Greedy	28.42	37.45	29.23	16.95	28.33	17.15
	Epsilon-Greedy - E. Greedy	42.99	44.07	48.92	36.76	33.58	36.90
20K	Greedy - Greedy	15.69	19.68	19.53	24.72	21.71	22.87
	Greedy - E. Greedy	27.11	37.44	27.30	18.09	16.09	17.58
	Epsilon-Greedy - E. Greedy	33.24	37.46	46.68	38.10	22.20	22.74
25K	Greedy - Greedy	18.92	12.12	14.09	28.00	16.20	17.06
	Greedy - E. Greedy	21.86	28.78	21.08	19.85	21.77	18.97
	Epsilon-Greedy - E. Greedy	34.72	40.79	41.14	30.04	21.10	14.54
30K	Greedy - Greedy	18.84	13.44	16.53	24.80	21.39	22.53
	Greedy - E. Greedy	20.03	35.24	11.00	15.11	21.08	14.91
	Epsilon-Greedy - E. Greedy	32.63	34.94	41.18	24.36	15.03	14.71
35K	Greedy - Greedy	14.76	19.18	13.82	23.02	12.51	16.80
	Greedy - E. Greedy	17.17	24.66	17.50	15.06	21.48	8.63
	Epsilon-Greedy - E. Greedy	25.23	29.41	41.27	25.13	17.80	20.47
40K	Greedy - Greedy	11.05	17.10	10.28	18.03	11.65	11.74
	Greedy - E. Greedy	13.09	26.96	18.01	17.65	17.16	15.26
	Epsilon-Greedy - E. Greedy	23.50	29.77	36.83	16.99	15.20	18.03
45K	Greedy - Greedy	7.12	10.97	5.84	16.74	13.23	13.12
	Greedy - E. Greedy	16.19	23.34	15.86	13.57	15.75	7.29
	Epsilon-Greedy - E. Greedy	30.30	26.80	34.90	13.41	8.95	10.45
50K	Greedy - Greedy	7.06	4.02	6.70	14.52	17.59	18.26
	Greedy - E. Greedy	15.52	22.19	13.51	10.80	21.98	5.96
	Epsilon-Greedy - E. Greedy	26.84	21.82	25.83	18.79	14.02	9.48
55K	Greedy - Greedy	9.26	6.59	5.96	16.11	12.50	10.71
	Greedy - E. Greedy	10.92	23.61	12.13	7.28	21.99	7.13
	Epsilon-Greedy - E. Greedy	23.99	25.97	21.74	16.20	13.05	10.38
60K	Greedy - Greedy	7.23	7.65	4.49	18.85	16.16	14.44
	Greedy - E. Greedy	7.46	18.95	7.84	6.15	17.26	5.51
	Epsilon-Greedy - E. Greedy	16.33	24.93	27.95	16.24	8.42	6.81

Table B.3. Dynamic replenishment ($m = 2, T = 10$): detailed cost gap (%) to equilibrium solution (cont.).

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
65K	Greedy - Greedy	8.51	3.70	4.96	11.34	9.97	11.73
	Greedy - E. Greedy	9.96	17.27	9.62	4.02	20.33	3.84
	Epsilon-Greedy - E. Greedy	18.42	26.89	30.84	6.68	5.95	7.76
70K	Greedy - Greedy	4.80	5.02	3.35	12.73	7.71	14.98
	Greedy - E. Greedy	9.56	13.64	3.37	4.48	14.47	4.13
	Epsilon-Greedy - E. Greedy	18.76	22.67	23.24	14.29	5.80	1.95
75K	Greedy - Greedy	3.88	2.78	3.86	11.67	9.33	10.15
	Greedy - E. Greedy	7.67	14.92	6.86	4.36	16.91	3.52
	Epsilon-Greedy - E. Greedy	26.95	23.82	21.89	9.95	4.80	4.65
80K	Greedy - Greedy	2.41	4.42	4.10	16.45	7.10	7.88
	Greedy - E. Greedy	5.79	17.26	7.29	4.53	13.91	4.58
	Epsilon-Greedy - E. Greedy	23.88	25.49	23.10	7.02	6.08	5.86
85K	Greedy - Greedy	3.75	4.16	5.30	16.69	7.82	9.43
	Greedy - E. Greedy	6.86	15.29	5.87	5.67	6.90	4.24
	Epsilon-Greedy - E. Greedy	24.45	24.68	21.14	4.37	4.64	7.33
90K	Greedy - Greedy	3.88	3.90	4.17	7.91	7.73	12.63
	Greedy - E. Greedy	4.62	11.07	4.24	2.60	7.31	3.32
	Epsilon-Greedy - E. Greedy	23.63	22.89	17.45	2.21	4.44	6.63
95K	Greedy - Greedy	4.07	3.27	3.09	4.32	6.82	12.22
	Greedy - E. Greedy	3.92	8.86	2.10	4.63	7.73	1.82
	Epsilon-Greedy - E. Greedy	16.44	20.49	17.03	3.73	7.61	3.51
100K	Greedy - Greedy	3.59	1.80	3.49	5.30	7.02	9.79
	Greedy - E. Greedy	3.77	10.14	3.29	2.05	10.29	1.56
	Epsilon-Greedy - E. Greedy	12.34	18.77	20.81	2.58	7.44	3.24
105K	Greedy - Greedy	1.62	3.86	1.43	5.47	11.29	9.31
	Greedy - E. Greedy	1.89	7.53	1.05	2.23	8.27	2.48
	Epsilon-Greedy - E. Greedy	6.84	14.38	18.48	3.22	17.85	2.35
110K	Greedy - Greedy	1.80	1.51	1.57	6.61	10.71	5.86
	Greedy - E. Greedy	2.07	4.83	1.28	2.20	3.92	2.16
	Epsilon-Greedy - E. Greedy	9.37	11.02	13.08	3.47	1.95	0.98
115K	Greedy - Greedy	2.24	1.38	1.50	4.44	9.64	8.01
	Greedy - E. Greedy	1.85	7.62	0.82	1.14	9.88	1.35
	Epsilon-Greedy - E. Greedy	9.84	5.29	9.01	2.00	1.37	1.07
120K	Greedy - Greedy	1.56	1.87	1.44	4.59	8.20	8.20
	Greedy - E. Greedy	1.76	5.96	2.27	2.36	6.81	1.13
	Epsilon-Greedy - E. Greedy	11.00	8.95	7.05	1.62	1.80	1.45
125K	Greedy - Greedy	2.43	2.35	1.66	3.59	7.83	7.45
	Greedy - E. Greedy	1.98	3.35	2.33	0.80	6.67	0.74
	Epsilon-Greedy - E. Greedy	7.17	9.97	7.37	1.10	1.09	2.07
Avg	Greedy - Greedy	9.54	9.27	8.46	16.39	13.84	14.89
	Greedy - E. Greedy	12.40	20.02	11.35	9.59	15.89	8.11
	Epsilon-Greedy - E. Greedy	23.87	26.48	28.02	15.23	12.87	11.75

Table B.4. Dynamic replenishment: detailed win-rate (%) under $m = 2$ and $T = 6$.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	40.00	60.00	27.00	73.00	48.00	52.00
	Greedy - E. Greedy	54.00	46.00	35.00	65.00	54.00	46.00
	Epsilon-Greedy - E. Greedy	40.00	50.00	53.00	47.00	75.00	25.00
10K	Greedy - Greedy	48.00	52.00	44.00	56.00	48.00	52.00
	Greedy - E. Greedy	28.00	72.00	28.00	72.00	52.00	48.00
	Epsilon-Greedy - E. Greedy	0.00	100.00	46.00	54.00	28.00	70.00
15K	Greedy - Greedy	55.00	45.00	21.00	79.00	36.00	64.00
	Greedy - E. Greedy	32.00	68.00	32.00	68.00	60.00	40.00
	Epsilon-Greedy - E. Greedy	20.00	72.00	65.00	35.00	62.00	38.00
20K	Greedy - Greedy	58.00	42.00	35.00	61.00	60.00	40.00
	Greedy - E. Greedy	52.00	48.00	40.00	60.00	65.00	35.00
	Epsilon-Greedy - E. Greedy	18.00	82.00	62.00	38.00	28.00	72.00
25K	Greedy - Greedy	76.00	24.00	14.00	86.00	56.00	44.00
	Greedy - E. Greedy	38.00	62.00	15.00	85.00	44.00	56.00
	Epsilon-Greedy - E. Greedy	88.00	12.00	57.00	43.00	52.00	48.00
30K	Greedy - Greedy	36.00	64.00	52.00	48.00	40.00	60.00
	Greedy - E. Greedy	72.00	28.00	20.00	80.00	28.00	72.00
	Epsilon-Greedy - E. Greedy	86.00	14.00	51.00	49.00	72.00	28.00
35K	Greedy - Greedy	58.00	42.00	49.00	51.00	60.00	40.00
	Greedy - E. Greedy	76.00	24.00	45.00	55.00	60.00	40.00
	Epsilon-Greedy - E. Greedy	80.00	20.00	49.00	51.00	48.00	52.00
40K	Greedy - Greedy	46.00	54.00	34.00	66.00	46.00	54.00
	Greedy - E. Greedy	80.00	20.00	38.00	62.00	100.00	0.00
	Epsilon-Greedy - E. Greedy	100.00	0.00	48.00	52.00	54.00	46.00
45K	Greedy - Greedy	52.00	48.00	54.00	46.00	56.00	44.00
	Greedy - E. Greedy	56.00	44.00	58.00	42.00	85.00	15.00
	Epsilon-Greedy - E. Greedy	84.00	16.00	45.00	55.00	55.00	45.00
50K	Greedy - Greedy	65.00	35.00	68.00	32.00	44.00	56.00
	Greedy - E. Greedy	68.00	32.00	43.00	57.00	68.00	32.00
	Epsilon-Greedy - E. Greedy	44.00	56.00	72.00	28.00	52.00	48.00
55K	Greedy - Greedy	44.00	56.00	66.00	34.00	54.00	46.00
	Greedy - E. Greedy	45.00	55.00	86.00	14.00	60.00	40.00
	Epsilon-Greedy - E. Greedy	12.00	88.00	51.00	49.00	30.00	70.00
60K	Greedy - Greedy	36.00	64.00	54.00	46.00	60.00	40.00
	Greedy - E. Greedy	57.00	43.00	75.00	25.00	52.00	48.00
	Epsilon-Greedy - E. Greedy	50.00	50.00	60.00	40.00	38.00	62.00

Table B.4. Dynamic replenishment: detailed win-rate (%) under $m = 2$ and $T = 6$
(cont.).

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
65K	Greedy - Greedy	24.00	76.00	58.00	18.00	44.00	56.00
	Greedy - E. Greedy	65.00	35.00	42.00	58.00	48.00	52.00
	Epsilon-Greedy - E. Greedy	80.00	20.00	45.00	55.00	42.00	58.00
70K	Greedy - Greedy	72.00	28.00	65.00	35.00	55.00	45.00
	Greedy - E. Greedy	74.00	26.00	45.00	55.00	40.00	60.00
	Epsilon-Greedy - E. Greedy	100.00	0.00	43.00	57.00	50.00	50.00
75K	Greedy - Greedy	42.00	56.00	67.00	33.00	60.00	40.00
	Greedy - E. Greedy	20.00	80.00	73.00	27.00	35.00	65.00
	Epsilon-Greedy - E. Greedy	0.00	100.00	41.00	59.00	34.00	66.00
80K	Greedy - Greedy	57.00	43.00	78.00	22.00	55.00	45.00
	Greedy - E. Greedy	72.00	28.00	70.00	30.00	38.00	62.00
	Epsilon-Greedy - E. Greedy	28.00	72.00	38.00	62.00	57.00	43.00
85K	Greedy - Greedy	54.00	46.00	82.00	18.00	68.00	32.00
	Greedy - E. Greedy	72.00	28.00	55.00	45.00	45.00	55.00
	Epsilon-Greedy - E. Greedy	88.00	12.00	40.00	60.00	60.00	40.00
90K	Greedy - Greedy	45.00	55.00	65.00	35.00	52.00	48.00
	Greedy - E. Greedy	65.00	35.00	53.00	47.00	48.00	52.00
	Epsilon-Greedy - E. Greedy	28.00	72.00	25.00	75.00	94.00	6.00
95K	Greedy - Greedy	48.00	52.00	62.00	38.00	46.00	54.00
	Greedy - E. Greedy	90.00	10.00	48.00	52.00	22.00	78.00
	Epsilon-Greedy - E. Greedy	60.00	40.00	28.00	72.00	40.00	60.00
100K	Greedy - Greedy	56.00	44.00	70.00	30.00	24.00	76.00
	Greedy - E. Greedy	95.00	5.00	42.00	58.00	15.00	85.00
	Epsilon-Greedy - E. Greedy	58.00	42.00	20.00	80.00	54.00	46.00
105K	Greedy - Greedy	54.00	46.00	92.00	8.00	60.00	40.00
	Greedy - E. Greedy	74.00	26.00	56.00	44.00	20.00	80.00
	Epsilon-Greedy - E. Greedy	26.00	74.00	10.00	90.00	30.00	70.00
110K	Greedy - Greedy	42.00	58.00	80.00	20.00	36.00	64.00
	Greedy - E. Greedy	82.00	18.00	47.00	53.00	20.00	80.00
	Epsilon-Greedy - E. Greedy	18.00	82.00	8.00	92.00	58.00	42.00
115K	Greedy - Greedy	60.00	40.00	78.00	22.00	46.00	54.00
	Greedy - E. Greedy	60.00	38.00	47.00	51.00	0.00	95.00
	Epsilon-Greedy - E. Greedy	40.00	60.00	10.00	88.00	32.00	68.00
120K	Greedy - Greedy	55.00	45.00	82.00	18.00	36.00	64.00
	Greedy - E. Greedy	85.00	15.00	48.00	52.00	8.00	92.00
	Epsilon-Greedy - E. Greedy	26.00	74.00	0.00	100.00	24.00	76.00
125K	Greedy - Greedy	50.00	50.00	80.00	20.00	70.00	30.00
	Greedy - E. Greedy	100.00	0.00	38.00	62.00	10.00	90.00
	Epsilon-Greedy - E. Greedy	72.00	28.00	0.00	100.00	72.00	28.00
Avg	Greedy - Greedy	50.92	49.00	59.08	39.80	50.40	49.60
	Greedy - E. Greedy	64.48	35.44	47.16	52.76	43.08	56.72
	Epsilon-Greedy - E. Greedy	49.84	49.44	38.68	61.24	49.64	50.28

Table B.5. Dynamic replenishment: detailed win-rate (%) under $m = 2$ and $T = 10$.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	35.00	65.00	35.00	65.00	44.00	56.00
	Greedy - E. Greedy	47.00	53.00	43.00	57.00	58.00	42.00
	Epsilon-Greedy - E. Greedy	47.00	43.00	58.00	42.00	27.00	73.00
10K	Greedy - Greedy	48.00	52.00	34.00	66.00	34.00	66.00
	Greedy - E. Greedy	35.00	65.00	18.00	82.00	48.00	52.00
	Epsilon-Greedy - E. Greedy	52.00	48.00	55.00	45.00	34.00	64.00
15K	Greedy - Greedy	32.00	68.00	44.00	56.00	48.00	52.00
	Greedy - E. Greedy	29.00	71.00	13.00	87.00	45.00	55.00
	Epsilon-Greedy - E. Greedy	48.00	44.00	78.00	22.00	56.00	44.00
20K	Greedy - Greedy	55.00	45.00	27.00	69.00	55.00	45.00
	Greedy - E. Greedy	44.00	56.00	35.00	65.00	46.00	54.00
	Epsilon-Greedy - E. Greedy	58.00	42.00	74.00	26.00	45.00	55.00
25K	Greedy - Greedy	54.00	46.00	18.00	82.00	52.00	48.00
	Greedy - E. Greedy	46.00	54.00	36.00	64.00	60.00	40.00
	Epsilon-Greedy - E. Greedy	24.00	76.00	58.00	42.00	75.00	25.00
30K	Greedy - Greedy	58.00	42.00	46.00	54.00	50.00	50.00
	Greedy - E. Greedy	65.00	35.00	48.00	52.00	63.00	37.00
	Epsilon-Greedy - E. Greedy	38.00	62.00	65.00	35.00	78.00	22.00
35K	Greedy - Greedy	63.00	37.00	52.00	48.00	50.00	50.00
	Greedy - E. Greedy	83.00	17.00	45.00	55.00	74.00	26.00
	Epsilon-Greedy - E. Greedy	36.00	64.00	42.00	58.00	29.00	71.00
40K	Greedy - Greedy	59.00	41.00	56.00	44.00	32.00	68.00
	Greedy - E. Greedy	80.00	20.00	32.00	68.00	86.00	14.00
	Epsilon-Greedy - E. Greedy	26.00	74.00	50.00	50.00	34.00	66.00
45K	Greedy - Greedy	44.00	56.00	58.00	42.00	27.00	73.00
	Greedy - E. Greedy	52.00	48.00	73.00	27.00	67.00	33.00
	Epsilon-Greedy - E. Greedy	68.00	32.00	50.00	50.00	41.00	59.00
50K	Greedy - Greedy	55.00	45.00	50.00	50.00	65.00	35.00
	Greedy - E. Greedy	63.00	37.00	45.00	55.00	51.00	49.00
	Epsilon-Greedy - E. Greedy	74.00	26.00	48.00	52.00	40.00	60.00
55K	Greedy - Greedy	38.00	62.00	42.00	58.00	58.00	42.00
	Greedy - E. Greedy	73.00	27.00	66.00	34.00	48.00	52.00
	Epsilon-Greedy - E. Greedy	85.00	15.00	44.00	56.00	55.00	45.00
60K	Greedy - Greedy	27.00	73.00	65.00	35.00	72.00	28.00
	Greedy - E. Greedy	57.00	43.00	82.00	18.00	35.00	65.00
	Epsilon-Greedy - E. Greedy	55.00	45.00	35.00	65.00	54.00	46.00
65K	Greedy - Greedy	54.00	46.00	60.00	16.00	88.00	12.00
	Greedy - E. Greedy	68.00	32.00	48.00	52.00	42.00	58.00
	Epsilon-Greedy - E. Greedy	58.00	42.00	52.00	48.00	58.00	42.00
70K	Greedy - Greedy	50.00	50.00	68.00	32.00	51.00	49.00
	Greedy - E. Greedy	70.00	30.00	46.00	54.00	40.00	60.00
	Epsilon-Greedy - E. Greedy	52.00	48.00	57.00	43.00	65.00	35.00
75K	Greedy - Greedy	51.00	47.00	70.00	30.00	48.00	52.00
	Greedy - E. Greedy	35.00	65.00	57.00	43.00	36.00	64.00
	Epsilon-Greedy - E. Greedy	54.00	46.00	44.00	56.00	60.00	40.00
80K	Greedy - Greedy	38.00	62.00	72.00	28.00	42.00	58.00
	Greedy - E. Greedy	55.00	45.00	55.00	45.00	49.00	51.00
	Epsilon-Greedy - E. Greedy	45.00	55.00	35.00	65.00	72.00	28.00
85K	Greedy - Greedy	22.00	78.00	65.00	35.00	35.00	65.00
	Greedy - E. Greedy	86.00	14.00	65.00	35.00	52.00	48.00
	Epsilon-Greedy - E. Greedy	53.00	47.00	36.00	64.00	24.00	76.00
90K	Greedy - Greedy	85.00	15.00	78.00	22.00	59.00	41.00
	Greedy - E. Greedy	65.00	35.00	51.00	49.00	30.00	70.00
	Epsilon-Greedy - E. Greedy	47.00	53.00	40.00	60.00	52.00	48.00

Table B.5. Dynamic replenishment: detailed win-rate (%) under $m = 2$ and $T = 10$
(cont.).

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
95K	Greedy - Greedy	64.00	36.00	80.00	20.00	68.00	32.00
	Greedy - E. Greedy	90.00	10.00	46.00	54.00	35.00	65.00
	Epsilon-Greedy - E. Greedy	42.00	58.00	28.00	72.00	48.00	52.00
100K	Greedy - Greedy	55.00	45.00	82.00	18.00	27.00	73.00
	Greedy - E. Greedy	95.00	5.00	40.00	60.00	33.00	67.00
	Epsilon-Greedy - E. Greedy	35.00	65.00	22.00	78.00	34.00	66.00
105K	Greedy - Greedy	44.00	56.00	100.00	0.00	60.00	40.00
	Greedy - E. Greedy	77.00	23.00	54.00	46.00	28.00	72.00
	Epsilon-Greedy - E. Greedy	88.00	12.00	15.00	85.00	38.00	62.00
110K	Greedy - Greedy	48.00	52.00	88.00	12.00	44.00	56.00
	Greedy - E. Greedy	84.00	16.00	45.00	55.00	0.00	100.00
	Epsilon-Greedy - E. Greedy	54.00	46.00	12.00	88.00	67.00	33.00
115K	Greedy - Greedy	48.00	52.00	95.00	5.00	48.00	52.00
	Greedy - E. Greedy	60.00	38.00	45.00	53.00	3.00	92.00
	Epsilon-Greedy - E. Greedy	15.00	85.00	8.00	90.00	74.00	26.00
120K	Greedy - Greedy	32.00	68.00	98.00	2.00	55.00	45.00
	Greedy - E. Greedy	75.00	25.00	46.00	54.00	10.00	90.00
	Epsilon-Greedy - E. Greedy	54.00	46.00	8.00	92.00	38.00	62.00
125K	Greedy - Greedy	75.00	25.00	100.00	0.00	74.00	26.00
	Greedy - E. Greedy	92.00	8.00	36.00	64.00	12.00	88.00
	Epsilon-Greedy - E. Greedy	52.00	48.00	0.00	100.00	36.00	64.00
Avg	Greedy - Greedy	49.36	50.56	63.32	35.56	51.44	48.56
	Greedy - E. Greedy	65.04	34.88	46.80	53.12	42.04	57.76
	Epsilon-Greedy - E. Greedy	50.40	48.88	40.56	59.36	49.36	50.56

Table B.6. Dynamic replenishment: detailed win-rate (%) under $m = 3$ and $T = 6$.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	42.00	58.00	28.00	72.00	55.00	45.00
	Greedy - E. Greedy	48.00	52.00	56.00	44.00	51.00	49.00
	Epsilon-Greedy - E. Greedy	35.00	55.00	52.00	48.00	35.00	65.00
10K	Greedy - Greedy	48.00	52.00	48.00	52.00	45.00	55.00
	Greedy - E. Greedy	52.00	48.00	65.00	35.00	49.00	51.00
	Epsilon-Greedy - E. Greedy	58.00	42.00	44.00	56.00	17.00	81.00
15K	Greedy - Greedy	55.00	45.00	22.00	78.00	42.00	58.00
	Greedy - E. Greedy	50.00	50.00	52.00	48.00	58.00	42.00
	Epsilon-Greedy - E. Greedy	17.00	75.00	66.00	34.00	50.00	50.00
20K	Greedy - Greedy	68.00	32.00	37.00	59.00	48.00	52.00
	Greedy - E. Greedy	45.00	55.00	46.00	54.00	64.00	36.00
	Epsilon-Greedy - E. Greedy	24.00	76.00	62.00	38.00	52.00	48.00
25K	Greedy - Greedy	83.00	17.00	14.00	86.00	60.00	40.00
	Greedy - E. Greedy	56.00	44.00	48.00	52.00	40.00	60.00
	Epsilon-Greedy - E. Greedy	67.00	33.00	57.00	43.00	58.00	42.00
30K	Greedy - Greedy	42.00	58.00	57.00	43.00	58.00	42.00
	Greedy - E. Greedy	56.00	44.00	34.00	66.00	22.00	78.00
	Epsilon-Greedy - E. Greedy	60.00	40.00	50.00	50.00	24.00	76.00
35K	Greedy - Greedy	50.00	50.00	53.00	47.00	54.00	46.00
	Greedy - E. Greedy	60.00	40.00	55.00	45.00	58.00	42.00
	Epsilon-Greedy - E. Greedy	64.00	36.00	48.00	52.00	48.00	52.00

Table B.6. Dynamic replenishment: detailed win-rate (%) under $m = 3$ and $T = 6$
(cont.).

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
40K	Greedy - Greedy	26.00	74.00	36.00	64.00	64.00	36.00
	Greedy - E. Greedy	62.00	38.00	60.00	40.00	83.00	17.00
	Epsilon-Greedy - E. Greedy	82.00	18.00	46.00	54.00	32.00	68.00
45K	Greedy - Greedy	66.00	34.00	59.00	41.00	56.00	44.00
	Greedy - E. Greedy	56.00	44.00	62.00	38.00	86.00	14.00
	Epsilon-Greedy - E. Greedy	54.00	46.00	43.00	57.00	35.00	65.00
50K	Greedy - Greedy	38.00	62.00	75.00	25.00	22.00	78.00
	Greedy - E. Greedy	60.00	40.00	82.00	18.00	67.00	33.00
	Epsilon-Greedy - E. Greedy	51.00	49.00	73.00	27.00	75.00	25.00
55K	Greedy - Greedy	46.00	54.00	72.00	28.00	35.00	65.00
	Greedy - E. Greedy	44.00	56.00	84.00	16.00	58.00	42.00
	Epsilon-Greedy - E. Greedy	47.00	53.00	50.00	50.00	82.00	18.00
60K	Greedy - Greedy	73.00	27.00	59.00	41.00	40.00	60.00
	Greedy - E. Greedy	35.00	65.00	70.00	30.00	49.00	51.00
	Epsilon-Greedy - E. Greedy	48.00	52.00	60.00	40.00	35.00	65.00
65K	Greedy - Greedy	56.00	44.00	63.00	37.00	46.00	54.00
	Greedy - E. Greedy	47.00	53.00	55.00	45.00	44.00	56.00
	Epsilon-Greedy - E. Greedy	56.00	44.00	43.00	57.00	45.00	55.00
70K	Greedy - Greedy	60.00	40.00	71.00	29.00	43.00	57.00
	Greedy - E. Greedy	72.00	28.00	44.00	56.00	35.00	65.00
	Epsilon-Greedy - E. Greedy	65.00	35.00	41.00	59.00	47.00	53.00
75K	Greedy - Greedy	64.00	36.00	73.00	27.00	76.00	24.00
	Greedy - E. Greedy	54.00	46.00	35.00	65.00	30.00	70.00
	Epsilon-Greedy - E. Greedy	19.00	81.00	39.00	61.00	51.00	49.00
80K	Greedy - Greedy	57.00	43.00	86.00	14.00	65.00	35.00
	Greedy - E. Greedy	18.00	82.00	37.00	63.00	33.00	67.00
	Epsilon-Greedy - E. Greedy	27.00	73.00	35.00	65.00	54.00	46.00
85K	Greedy - Greedy	54.00	46.00	90.00	10.00	57.00	43.00
	Greedy - E. Greedy	72.00	28.00	38.00	62.00	41.00	59.00
	Epsilon-Greedy - E. Greedy	33.00	67.00	37.00	63.00	58.00	42.00
90K	Greedy - Greedy	35.00	65.00	71.00	29.00	48.00	52.00
	Greedy - E. Greedy	68.00	32.00	22.00	78.00	44.00	56.00
	Epsilon-Greedy - E. Greedy	56.00	44.00	21.00	79.00	67.00	33.00
95K	Greedy - Greedy	22.00	78.00	68.00	32.00	32.00	68.00
	Greedy - E. Greedy	88.00	12.00	50.00	50.00	35.00	65.00
	Epsilon-Greedy - E. Greedy	60.00	40.00	24.00	76.00	72.00	28.00

Table B.6. Dynamic replenishment: detailed win-rate (%) under $m = 3$ and $T = 6$
(cont.).

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
100K	Greedy - Greedy	0.00	100.00	77.00	23.00	45.00	55.00
	Greedy - E. Greedy	84.00	16.00	37.00	63.00	34.00	66.00
	Epsilon-Greedy - E. Greedy	57.00	43.00	15.00	85.00	75.00	25.00
105K	Greedy - Greedy	48.00	52.00	90.00	10.00	52.00	48.00
	Greedy - E. Greedy	82.00	18.00	42.00	58.00	28.00	72.00
	Epsilon-Greedy - E. Greedy	26.00	74.00	4.00	96.00	45.00	55.00
110K	Greedy - Greedy	50.00	50.00	88.00	12.00	36.00	64.00
	Greedy - E. Greedy	90.00	10.00	30.00	70.00	13.00	87.00
	Epsilon-Greedy - E. Greedy	55.00	45.00	17.00	83.00	52.00	48.00
115K	Greedy - Greedy	53.00	47.00	86.00	14.00	40.00	60.00
	Greedy - E. Greedy	80.00	18.00	22.00	76.00	13.00	82.00
	Epsilon-Greedy - E. Greedy	69.00	31.00	15.00	83.00	68.00	32.00
120K	Greedy - Greedy	60.00	40.00	90.00	10.00	46.00	54.00
	Greedy - E. Greedy	82.00	18.00	45.00	55.00	15.00	85.00
	Epsilon-Greedy - E. Greedy	82.00	18.00	8.00	92.00	28.00	72.00
125K	Greedy - Greedy	44.00	56.00	88.00	12.00	43.00	57.00
	Greedy - E. Greedy	88.00	12.00	35.00	65.00	2.00	98.00
	Epsilon-Greedy - E. Greedy	70.00	30.00	5.00	95.00	55.00	45.00
Avg	Greedy - Greedy	49.60	50.32	64.04	34.84	48.32	51.68
	Greedy - E. Greedy	61.96	37.96	48.24	51.68	42.08	57.72
	Epsilon-Greedy - E. Greedy	51.28	48.00	38.20	61.72	50.40	49.52

Table B.7. Dynamic replenishment: detailed win-rate (%) under $m = 3$ and $T = 10$.

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
5K	Greedy - Greedy	65.00	35.00	27.00	73.00	56.00	44.00
	Greedy - E. Greedy	38.00	62.00	27.00	73.00	56.00	44.00
	Epsilon-Greedy - E. Greedy	75.00	15.00	73.00	27.00	75.00	25.00
10K	Greedy - Greedy	44.00	56.00	45.00	55.00	66.00	34.00
	Greedy - E. Greedy	37.00	63.00	65.00	35.00	44.00	56.00
	Epsilon-Greedy - E. Greedy	32.00	68.00	70.00	30.00	65.00	33.00
15K	Greedy - Greedy	52.00	48.00	58.00	42.00	52.00	48.00
	Greedy - E. Greedy	45.00	55.00	60.00	40.00	41.00	59.00
	Epsilon-Greedy - E. Greedy	45.00	47.00	68.00	32.00	62.00	38.00
20K	Greedy - Greedy	58.00	42.00	50.00	46.00	58.00	42.00
	Greedy - E. Greedy	48.00	52.00	52.00	48.00	42.00	58.00
	Epsilon-Greedy - E. Greedy	52.00	48.00	65.00	35.00	44.00	56.00
25K	Greedy - Greedy	32.00	68.00	52.00	48.00	60.00	40.00
	Greedy - E. Greedy	52.00	48.00	55.00	45.00	58.00	42.00
	Epsilon-Greedy - E. Greedy	58.00	42.00	56.00	44.00	59.00	41.00
30K	Greedy - Greedy	25.00	75.00	44.00	56.00	66.00	34.00
	Greedy - E. Greedy	50.00	50.00	58.00	42.00	61.00	39.00
	Epsilon-Greedy - E. Greedy	55.00	45.00	60.00	40.00	27.00	73.00

Table B.7. Dynamic replenishment: detailed win-rate (%) under $m = 3$ and $T = 10$
(cont.).

E	Exploration Strategy	$\alpha_1 = 0.5, \alpha_2 = 0.5$		$\alpha_1 = 0.5, \alpha_2 = 0.7$		$\alpha_1 = 0.7, \alpha_2 = 0.7$	
		Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2	Pharmacy 1	Pharmacy 2
35K	Greedy - Greedy	64.00	36.00	40.00	60.00	42.00	58.00
	Greedy - E. Greedy	41.00	59.00	82.00	18.00	74.00	26.00
	Epsilon-Greedy - E. Greedy	57.00	43.00	45.00	55.00	18.00	82.00
40K	Greedy - Greedy	48.00	52.00	41.00	59.00	68.00	32.00
	Greedy - E. Greedy	46.00	54.00	74.00	26.00	87.00	13.00
	Epsilon-Greedy - E. Greedy	42.00	58.00	42.00	58.00	45.00	55.00
45K	Greedy - Greedy	47.00	53.00	37.00	63.00	68.00	32.00
	Greedy - E. Greedy	60.00	40.00	44.00	56.00	66.00	34.00
	Epsilon-Greedy - E. Greedy	17.00	83.00	40.00	60.00	50.00	50.00
50K	Greedy - Greedy	28.00	72.00	65.00	35.00	35.00	65.00
	Greedy - E. Greedy	65.00	35.00	50.00	50.00	48.00	52.00
	Epsilon-Greedy - E. Greedy	28.00	72.00	32.00	68.00	52.00	48.00
55K	Greedy - Greedy	55.00	45.00	60.00	40.00	42.00	58.00
	Greedy - E. Greedy	68.00	32.00	44.00	56.00	44.00	56.00
	Epsilon-Greedy - E. Greedy	40.00	60.00	25.00	75.00	58.00	42.00
60K	Greedy - Greedy	54.00	46.00	72.00	28.00	28.00	72.00
	Greedy - E. Greedy	42.00	58.00	40.00	60.00	30.00	70.00
	Epsilon-Greedy - E. Greedy	22.00	78.00	28.00	72.00	48.00	52.00
70K	Greedy - Greedy	28.00	72.00	80.00	20.00	35.00	65.00
	Greedy - E. Greedy	58.00	42.00	35.00	65.00	35.00	65.00
	Epsilon-Greedy - E. Greedy	62.00	38.00	25.00	75.00	62.00	38.00
75K	Greedy - Greedy	36.00	62.00	55.00	45.00	71.00	29.00
	Greedy - E. Greedy	72.00	28.00	35.00	65.00	31.00	69.00
	Epsilon-Greedy - E. Greedy	52.00	48.00	30.00	70.00	63.00	37.00
80K	Greedy - Greedy	55.00	45.00	59.00	41.00	86.00	14.00
	Greedy - E. Greedy	70.00	30.00	33.00	67.00	46.00	54.00
	Epsilon-Greedy - E. Greedy	58.00	42.00	35.00	65.00	80.00	20.00
90K	Greedy - Greedy	54.00	46.00	70.00	30.00	22.00	78.00
	Greedy - E. Greedy	75.00	25.00	58.00	42.00	24.00	76.00
	Epsilon-Greedy - E. Greedy	60.00	40.00	20.00	80.00	37.00	63.00
95K	Greedy - Greedy	52.00	48.00	65.00	35.00	32.00	68.00
	Greedy - E. Greedy	70.00	30.00	55.00	45.00	30.00	70.00
	Epsilon-Greedy - E. Greedy	43.00	57.00	16.00	84.00	55.00	45.00
100K	Greedy - Greedy	55.00	45.00	85.00	15.00	73.00	27.00
	Greedy - E. Greedy	65.00	35.00	60.00	40.00	28.00	72.00
	Epsilon-Greedy - E. Greedy	57.00	43.00	33.00	67.00	34.00	66.00
105K	Greedy - Greedy	67.00	33.00	82.00	18.00	40.00	60.00
	Greedy - E. Greedy	88.00	12.00	62.00	38.00	22.00	78.00
	Epsilon-Greedy - E. Greedy	50.00	50.00	22.00	78.00	38.00	62.00
110K	Greedy - Greedy	72.00	28.00	90.00	10.00	56.00	44.00
	Greedy - E. Greedy	84.00	16.00	40.00	60.00	8.00	92.00
	Epsilon-Greedy - E. Greedy	54.00	46.00	18.00	82.00	38.00	62.00
115K	Greedy - Greedy	70.00	30.00	92.00	8.00	20.00	80.00
	Greedy - E. Greedy	54.00	44.00	32.00	66.00	5.00	90.00
	Epsilon-Greedy - E. Greedy	28.00	72.00	18.00	80.00	27.00	73.00
120K	Greedy - Greedy	32.00	68.00	90.00	10.00	24.00	76.00
	Greedy - E. Greedy	80.00	20.00	30.00	70.00	15.00	85.00
	Epsilon-Greedy - E. Greedy	85.00	15.00	8.00	92.00	58.00	42.00
125K	Greedy - Greedy	18.00	82.00	88.00	12.00	38.00	62.00
	Greedy - E. Greedy	85.00	15.00	18.00	82.00	20.00	80.00
	Epsilon-Greedy - E. Greedy	57.00	43.00	0.00	100.00	50.00	50.00
Avg	Greedy - Greedy	48.72	51.20	63.36	35.52	47.64	52.36
	Greedy - E. Greedy	61.20	38.72	46.92	53.00	40.08	59.72
	Epsilon-Greedy - E. Greedy	49.56	49.72	35.28	64.64	49.12	50.80