DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MARKOV DECISION PROCESS IN INVENTORY MANAGEMENT

by
Abubakari SUMAİLA

July, 2019
İZMİR

MARKOV DECISION PROCESS IN INVENTORY MANAGEMENT

A Thesis Submitted to the
Graduate School of Natural And Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Master of
Science in Statistics, Statistics Program

by
Abubakari SUMAİLA

July, 2019
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled "**MARKOV DECISION PROCESS IN INVENTORY MANAGEMENT**" completed by **ABUBAKARİ SUMAİLA** under supervision of **ASSOC. PROF. DR. UMAY UZUNOĞLU KOÇER** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Assoc. Prof. Dr. Umay Uzunoğlu KOÇER

Supervisor

.....

Assoc. Prof. Dr. Tuğba YILDIZ

Jury Member

.....

Assoc. Prof. Dr. Ali Mert

Jury Member

.....

Prof. Dr. Kadriye ERTEKİN

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

Upon the successful completion of my M.Sc thesis in the Department of Statistics, I am pleased to thank a few people who have contributed immensely towards this success.

First of all, I would like to express my heartfelt gratitude to my supervisor, Assoc. Prof. Dr. Umay Uzunoğlu KOÇER for her ceaseless support during the preparation of this thesis. I consider it a great privilege to have worked under her academic direction and to benefit from her valuable pieces of advice, motivation, constructive criticisms and excellent human qualities. I am truly grateful to her.

Special thanks to Assoc. Prof. Dr. Tuğba YILDIZ and Assoc. Prof. Dr. Ali MERT for accepting to be part of the M.Sc thesis examining body, and the time they freed to be there. I would also like to thank all the instructors in the Department of Statistics, particularly Prof. Dr. Esin Firuzan, Prof. Dr. Özlem Ege Oruç, Prof. Dr. Selma Gürler and Assoc. Prof. Engin Yıldıztepe, who have all played a part in giving me a fairly great foundation in Statistics.

Furthermore, my sincere felicitations to the *Presidency for Turks Abroad and Related Communities (YTB)* for supporting me financially during my three-year stay in Turkey. Their support made it possible for me to take a preparatory language course in Turkish. This unique opportunity has not only broadened my horizon but has also gone a long way to making my life in Turkey very easy.

Finally, I would like to acknowledge the dearest person in my life, my mom, *Mamata Azumi* for all the sacrifices she made for me. Also, to my brothers and sisters, and all friends who in one way or the other made me a better person!!!

Abubakari SUMAİLA

MARKOV DECISION PROCESS IN INVENTORY MANAGEMENT

ABSTRACT

In any industrial production, the need for strategic or contingency planning continues to attract more attention, especially with the growing competitiveness of today's markets. Managers of companies seek to adopt policies that in the long run minimize operational costs. In this thesis, a case study of an inventory control problem is studied and an *optimal stationary* policy is proposed. The problem has to do with optimal production scheduling of limestones, a firm located in Trakya, Turkey. A production schedule that minimizes expected discounted costs is sought. The mathematical modeling framework used for this study is infinite-horizon *Markov decision process*. The problem is formulated as a sequential decision problem and subjected to rigorous computational test. We proposed an initial policy which is obtained through policy iteration methodology and we check whether it is optimal. Based on this, our proposed policy is found to be optimal. Whiles this policy is optimal, we do not claim that it is the only optimal solution to the problem herein studied. Other forms of optimal policies to this same problem may exist.

Keywords: Markov decision process, policy iteration, dynamic programming, inventory control, optimal production scheduling

ENVANTER YÖNETİMİNDE MARKOV KARAR SÜRECİ

ÖZ

Günümüzün sürekli gelişen rekabet ortamında herhangi bir endüstriyel üretim sürecinde stratejik veya olası durum planları geliştirmek ihtiyacı giderek önem kazanmaktadır. İşletme yöneticileri uzun dönemde işletme maliyetlerini en küçükleyecek politikaları uygulamak isterler. Bu tezde envanter kontrol problemi için bir uygulama çalışması ele alınmış ve *optimal durağan politika* önerilmiştir. Problem, Türkiye'nin Trakya bölgesinde bulunan bir işletmenin kireç üretimi için belirleyeceği optimal üretim planı ile ilgilidir. Beklenen bugünkü maliyetleri en küçükleyen optimal üretim planı araştırılmıştır. Bu çalışma için kullanılan matematiksel modelleme yöntemi sınırsız periyot için *Markov karar süreci* olmuştur. Problem, ardışık karar problemi olarak formüle edilmiş ve titiz bir hesaplama sürecine tabi tutulmuştur. Politika iterasyonu metodolojisi ile önce bir başlangıç çözüm önerilmiş ve bu çözümün optimal olup olmadığı kontrol edilmiştir. Bu prosedüre dayanarak önerilen politikanın optimal olduğu bulunmuştur. Bu politika optimal olmakla birlikte tek optimal çözümün bu olmadığı söylenebilir. Aynı problem için başka durağan politikalar da mevcut olabilir.

Anahtar kelimeler: Markov karar süreci, politika iterasyonu, dinamik programlama, envanter kontrolü, optimal üretim planlaması

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

In this chapter, a general introduction of the thesis is given. We first begin by giving the background of the study. Then we proceed to provide the problem statement of our work, followed by the research objectives. We conclude the chapter by outlining the organization of subsequent chapters.

1.1 Background of the study

With the growing complexity of demand forecasting in today's business world, the need for efficient management of inventory has become a serious issue of concern to business managers. This is so because controlling the cost of production and supply chain management does rely on striking a balance between what is demanded and what is needed or produced. Inventory control refers to the process of controlling and managing a company's non-capitalized assets. By this, a company aims at ensuring that stores for inventory are not, in the course of time, overstocked or understocked. Thus, an inventory management problem can be viewed as a decision problem that companies face in attempting to achieve their set objectives. Wild (2017) describes inventory control as an activity which makes items available to customers, and facilitates the purchasing, manufacturing and distributing functions to meet marketing demands. Gourdin (2001) beautifully demonstrates this point "Inventory is one area of logistics that has received a great deal of management attention over the past decade. Executives now realize that holding excessive stocks is simply too expensive. Therefore, a great deal of effort has been expended to eliminate unnecessary inventory without compromising customer service. However, there are numerous situations where inventory simply must be held, particularly when meeting the needs of global customers. Management's goal should be to hold only what is necessary to satisfy customer requirements and manage it effectively" (p. 82).

This thesis seeks an application of Markov decision process (stochastic dynamic

programming) to an inventory control problem. More precisely, the study is geared towards optimal *production scheduling* of limestones. There is abundant research on the applications of MDPs. A survey of this has been done by White (1993). Available applications have ranged from population harvesting, epidemics, sports to inventory control systems. In some cases, the applications are done with respect to real life problems, and some cases, for hypothetical inquisitiveness. One such area for application to real data is the manufacturing industry. In this industry, scheduling a production is extremely important as manufacturing systems can turn out to be very complex, especially in situations where demand curve is highly stochastic. Firm managers are, thus, confronted with problems of lot size and ideal production times. The key questions they would seek to address are; when to produce? By what amount to produce? Therefore, we may define production schedule as the handling of a production problem, in regard of how to design a factory, such that customer demands are consistently and sequentially met.

In literature, various production scheduling methods have been developed for different manufacturing systems. In this study, we restrict ourselves to a production scheduling method described as *"make to stock"*. With this method, the system is configured in a way that production of limestones are large enough to keep warehouse stocks sufficient for customer demands. Available literature shows various production scheduling problems with numerous variants and their solution structure. Scholars such as Patterson et al. (1989) assert that most practical production scheduling problems do not have computationally efficient solutions. Optimal solution methods usually involve Linear programming (LP). However, LP solution methods are over modified in most cases. Anwar & Nagi (1997) gave a formulation of an integrated scheduling and lot-sizing problem. Schneider et al. (1998) applied a Markov decision process (value base function) for production scheduling problem with stochastic demands. Other approaches have been used to tackle production scheduling problems. For instance, Caramanis et al. (2003) studied the importance of Supply Chain (SC) production planning. They produced a model that formulates a strategic planning for weekly production.

The benefits of production scheduling of limestones and their management are numerous and obvious. As a result of the calcium carbonate content present in limestones, its usage is well documented; from the earliest civilizations in history to usage today. It is highly valuable in many industries such as in the building and construction industry, in agriculture and heavy industries like steel production industry. A comprehensive treatise of lime and limestones is discussed in Boynton (1966), Oates (1998) and Dubois (1995). Citing available statistics for usage of limestones in New Zealand alone, it is estimated that in the year 2010, New Zealand used a little over 3.9 million tonnes for construction and agriculture. For the industry and road-construction related uses, the estimate puts the value at over 870 000 tonnes. The calcium carbonate in limestones has long been established and recognized as an agent for regulating acidity in the soil, which facilitates the growth of crops, thereby increasing crop yield. As fillers, bitumen is used with matter referred to as solid aggregate. As a matter of fact, it facilitates in the construction of parks, roads, driveways etc. In mining, powdered limestone functions as suppressant in coal mines. In this way, it insulates the mining site against fire and explosions. In beautifying smart cities, decorative limestone aggregates, chippings and grits are used in landscaping projects such as cycling ways and footpaths, driveways and car parks as well as for ground cover in gardens and rockeries. From the foregoing examples, it quite obvious that optimal inventory control of limestones would benefit production firms in very significant ways.

1.2 Problem statement

In this thesis, the main problem which the study aims at is to achieve an optimal production scheduling of limestones; a case study in the region of Trakya, Turkey. A limestone-processing firm in this region has plants where there is a large-scale processing of limestones. Thus, limestones are processed, stored and sold in large quantities (tonnes). Processing decisions are taken weekly and customer demands are stochastic. An optimal inventory control policy is sought, a policy which minimizes

weekly production, holding and miscellaneous costs.

The running of optimal inventories is quite essential to all firms. To buttress this point, Bertolini & Rizzi (2002) explains further, "The optimal management of inventories is a primary objective for all the firms manufacturing make to stock finished goods. As a matter of fact, inventories have important implications for both the financial and the economic performance of the company; therefore, it is widely acknowledged that an optimal inventory management policy allows companies to achieve higher profitability levels. In general terms, inventory management policies should be aimed at lowering the holding costs through higher inventory rotation, but without triggering substantial stockouts and backorders, caused by demand peaks and/ or lead time delays". By way of summary, the statement problem may be summarized as:

What weekly production schedule of the firm optimally minimizes production costs and holding costs of limestones?

1.3 Research objectives

The main goal of this research is to answer the core problem posed in section 1.2. That is to say, the determination of optimal production scheduling of limestones, a case study. This goal is achieved by applying Markov decision process as a mathematical modeling framework to an inventory management problem. If the inventory position is known prior to commencement of production, then determining how much to produce at the next production period becomes a decision problem.

Sub-objectives which lead to the realization of the main objective are:

i. Characterization of the problem as a Markov decision process.
   Thus, correctly identifying the decision epochs, state space, decision variable, transition probabilities and costs;

ii. Derivation of a recursive model for minimizing production and holding costs;

iii. Policy classification and the determination of optimal ones or near optimal ones.

1.4 Thesis structure

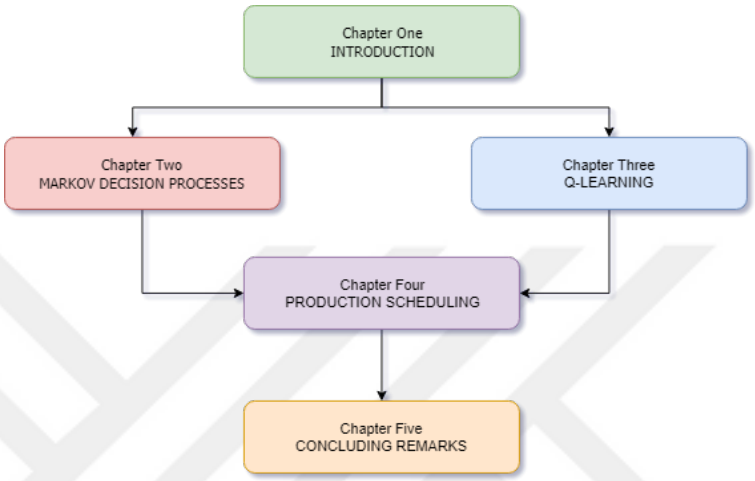

Figure 1.1 Structure of the thesis

The remaining part of this thesis is divided into four chapters. The outline of those four chapters are as follows:

Chapter 2: In this chapter, we give an overview of Markov decision processes (MDPs). The components of an MDP are identified. Various methods for solving decision problems are discussed, including methods for solving finite-horizon MDP models and infinite-horizon MDP models. This chapter also presents an inventory management problem which is then solved using the concepts therein presented. We close the chapter discussing the limitations of MDPs.

Chapter 3: This chapter mainly discusses Q-learning as an alternative to dynamic programming. The main goal of the chapter is to present the concepts of Q-learning as a type of reinforcement learning algorithm that can be used to solve inventory control problems, by 'learning' policies of a dynamic system.

Chapter 4: In this chapter, an application of MDP to an inventory control problem

is studied. The emphasis of this application is on the optimal *production schedule* of limestones. The motivation for the methodology used is drawn from concepts discussed in chapter 2 and chapter 3. This chapter is computationally intensive.

Chapter 5: This chapter concludes the study. It consists of concluding remarks based on the results obtained in chapter 4. The solution structure of the study is interpreted and discussed. We close the chapter by commenting on a possible modification for future study. The structure of the whole thesis is summarized in Figure 1.1

CHAPTER TWO
MARKOV DECISION PROCESSES (MDPS)

2.1 Background

A *Markov decision process (MDP)* can be thought of as a mathematical framework used to make informed decisions in stochastic environments. It forms a special branch of sequential decision theory where decision problems are modeled based on the system's characteristics. The phrase *"Markov decision processes"* was coined by Bellmanet al. (1954). History of Markov decision processes began in the *1950s* when they were used to study stochastic optimal control of systems. Earliest application of these can be found in inventory systems, from determining basic reorder point of products to a more complex versions of decision problems, Puterman (2014). Since then, MDPs have expanded swiftly to encompass a wider spectrum of applications. In this chapter, we give an overview of Markov decision processes. We shall identify the components of an MDP model and outline the various methods, under MDP settings, adopted to solve decision problems. We would particularly look at methods for solving *finite-horizon MDP* models and *infinite-horizon MDP* models. We also discuss optimal decision policies, π^*, as well as limitations of MDPs. Finally, we close the chapter by presenting an inventory management problem, and then proceed to use the concepts presented in this chapter to solve the problem.

2.2 Sequential decision processes

Markov decision processes (MDPs) have in recent years gained tremendous recognition in diverse fields of research. These research areas include; Robotics, Communication engineering, Ecology, Economics, Sports etc. MDPs have served as suitable for modelling sequential decision problems, where the decision maker is confronted with outcomes that are partly stochastic and partly under their control.

Prior to the final decision being made, the decision maker is confronted with series of decisions, goes through them, of which their goal is to reach the *'best'* of the available decisions. This is usually based on a function somewhat described as a *decision rule*.

To describe a sequential decision model such as an MDP, we consider a system as shown in Figure 2.1 on page (9). The decision maker who we otherwise referred to as agent (controller), observes the *states* s_t of the system at discrete times t. With respect to the dynamics of the system, the *agent* chooses an *action* a_t, from the possible available actions of the *action space*. The action taken produces either of two results: the agent receives an immediate reward, r_t or incurs an immediate cost. This action causes the system to evolve into a new state s_{t+1}. The evolution of the state into the new state happens according to a *probability distribution* which is determined by the dynamics of the system being studied. The process is then repeated at subsequent times. At each time the process is repeated, it results in either the present state staying as it were, or moving into a different state (we say it has *transited*). The tendency of the present state of the system to move into a different state is based on the action space of each decision epoch. In a maximization problem, the goal is to maximize total rewards, as it interacts with the system continuously; the goal is to minimize total cost in a minimization problem. Since the system we model is ongoing, the state of the system prior to tomorrow's decision depends on today's decision. The *objective* of a sequential decision-making process is to find a policy that maximizes a measure of long-run expected rewards (or minimizes long run expected costs). For an infinite time-horizon MDP model, future rewards are often discounted over an infinite decision epoch. If the sequential decision process we are dealing with is considered a Markov process, then it is important to note that, for the next state depends only and only on the current state and not on the history of how it has evolved until now. This property is termed as the *Markovian property*. It is also important to note that, the decision process is finite if the system states, the action sets and decision times are finite. In this thesis, our work would deal basically with an infinite-horizon MDP. This would be demostrated in Chpater 4.

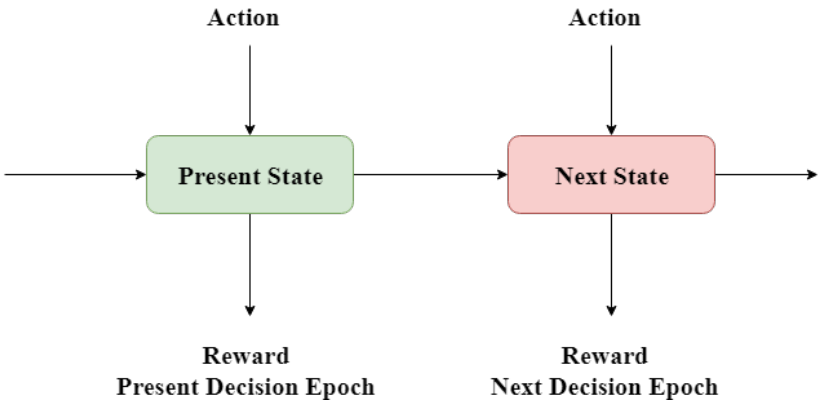

Figure 2.1 A diagrammatic representation of a sequential decision process

2.3 Components of Markov decision processes (MDPs)

In a sequential decision model, the agent or the controller's observation of the interactions in the system are made possible by identifying keys components of the process of the dynamic system. These key aspects of the process are the:

i. set of decision epoch,

ii. set of system states,

iii. action space,

iv. transition probabilities,

v. reward or cost function.

As the system evolves time after time, the agent or the decision maker aims at maximizing their rewards or minimizing their costs. The above components constitute an MDP. In general, a process denoted as an MDP is a *four-tuple*. Thus;

$$\left\{S, A, p(j|i, a \in A_s), r(j|i, a \in A_s)\right\}$$

In the ensuing sub-sections, we take each component and discuss them further.

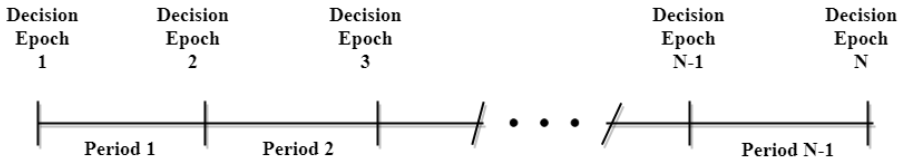

Figure 2.2 Illustration of decision epochs and periods

2.3.1 Decision epochs

Decision epochs refer to points of time where decisions are made. The set of decision epochs is defined as: $t = 1, 2, ..., T$. At each decision epoch *t*, the state s_t of the system is observed. An action a_t is chosen which results in an immediate reward or immediate cost $r_t(s, a)$. For instance, in production scheduling where decision makers observe the inventory position of a system, production decisions could be made weekly, monthly, yearly or some other predetermined timeline. These decision times form what we termed as *decision epochs* (also referred to as periods or stages) of the system. The elements of *T* can classified into broadly two ways. They can be either *discrete sets or continuum*; they can also be categorized as either *finite or infinite sets*. For a discrete set, decisions are made at all decision epochs whiles in a continuum case, decisions can be made in many ways such as:

- making decisions at all epochs,
- at random points of time depending on the problem under study,
- at arbitrary time points, based on the agent's discretion.

In discrete-time problems, we divide the time into *periods or stages*. This is illustrated in Figure 2.2. When this is done, we can then formulate the model such that the decision epochs correspond to the beginning of each stage or period. In that case, decision epochs can either be finite or infinite. If we consider a scenario where the decision epochs are finite, *i.e.* $T = \{1, 2, ..., N\}$, then we would be dealing with a *finite horizon problem*. The last decision is made at time $N - 1$. No decision is made at epoch *N*, as we would factor it in the final evaluation of the state of the system.

2.3.2 States

The *state space S* of an MDP model comprises all the likely combinations of the system states. At a decision epoch, the agent observes the system in state $s \in S$ and may choose an action, *a*, from the sets of allowable actions in the observed state *s*, denoted by A_s. The decision maker needs to decide what pieces of information in the system are vital so that an informed decision can be taken. The value of this vital information is what we termed as the *state* of the system. Consider a production firm where management is concerned with optimal production of items to meet weekly demands. In such a system, the state of the system would be the amount of inventory at the start of the week. It is important to note here that, $A = \cup_{s \in S} A_s$. We usually assume that the state and action space do not vary with time. The state space of the system may be arbitrarily *finite* sets, countably *infinite* sets or non-empty *Borel sets*. This thesis only considers a problem with discrete *discrete states*.

2.3.3 Actions

The action space A, consists of all the available actions to the agent/decision maker, given that the system is in state $s \in S$ at decision epoch *t*. In the production example given in the above section, the action to the system, upon reviewing the inventory position, would be to either produce (raise inventory level) or do nothing (continue with current inventory level). Like the *state sets*, the *action sets* can be either arbitrary finite sets, arbitrary countably infinite sets or non-empty *Borel sets*. At a given stage or period, the agent observes the system in state *s* and may choose actions *randomly or deterministically*.

2.3.4 Transition probabilities

At each decision epoch, the agent chooses an action which causes the system to evolve. The MDP transits to either the same state or to a new state. The transition of

the system solely depends on the action taken on the states. Given that the system is in state s_t at decision epoch *t*, the probability that the system evolves into state s_{t+1} at time *t+1* is denoted as $p(s_{t+1}|s_t, a_t)$. In other words, an action a_t has been taken. This caused the system to move from state s_t to state s_{t+1} at epoch $t+1$. The transition probability distribution, thus, determines the evolution of the states of the system. Referring to the earlier example cited, the transition probability reflects the fact that, inventory position has been raised or depleted.

2.3.5 Reward-cost function

The consequence of choosing action $a \in A_s$ in s_t is that, the decision maker receives a reward denoted as $r_t(s, a)$. The system state at the next decision epoch is determined by the probability distribution $p_t(.|s, a)$. For $s \in S$ and $a \in A_t$ at time *t*, the *real-value function* $r_t(s, a)$ denotes the reward received at time *t*. We may refer to it as income when it is *positive* and cost when it is *negative*. Depending on the MDP problem we are dealing with, the reward may be accrued at random times or at fixed times prior to the next decision epoch. It may be accrued repeatedly through the current period or a mixed of both scenarios. Throughout the entire time horizon, the expected reward $r_t(s, a)$ can be computed as:

$$r_t(s, a) = \sum_{j \in S}^{N-1} r_t(s, a, j) p_t(j|s, a). \tag{2.1}$$

From (2.1), the non-negative function $p_t(j|s, a)$ represents the probability that the system is in state $j \in S$ at time $t+1$, given that the decision maker selects action $a \in A_s$ while in *state s* at time *t*. It is important to formulate the model so that transitions which occur between decision epochs do not influence the decision maker in anyway. The *transition probability function* $p_t(j|s, a)$ is usually assumed to sum to 1 over $j \in S$. Thus;

$$\sum_{j \in S}^{N-1} p_t(j|s, a) = 1. \tag{2.2}$$

The goal to minimize or maximize a sequential decision problem therefore, relies completely on the proper characterization of the system's components, viz; periods, states, actions, rewards and transitions probabilities. Figure 2.3 shows a pictorial summary of the characterization of an *infinite-horizon MDP*, with respect to production in inventory management.

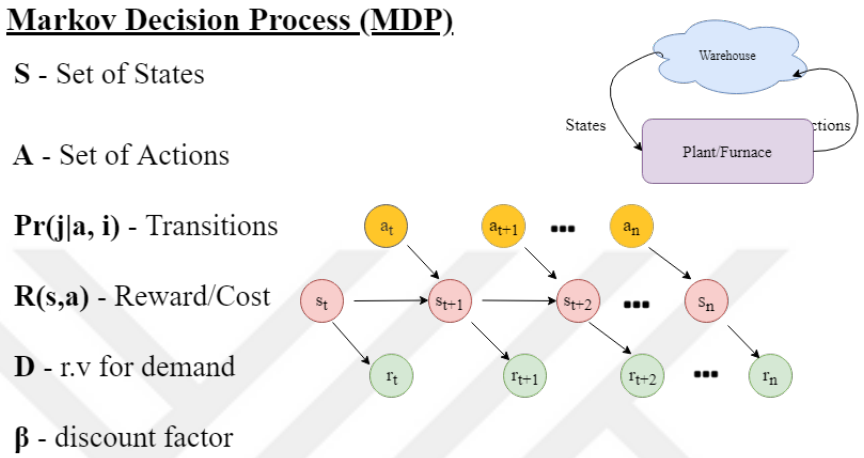

Figure 2.3 Summary components of an MDP

2.4 Solving Markov decision processes (MDPs)

As earlier discussed, an MDP model is defined by five key components. Once these components are accurately identified and defined, the decision maker's goal is to determine *optimal decision policy*. Here, an optimal policy is defined as a *decision rule* which the decision maker adopts in order to minimize expected total costs or maximize total expected rewards (along the time horizon). Thus, the *decision rule*, denoted as d_t, is a function that is capable of directing the decision maker regarding what actions to take in the states at each decision epoch *t*. The commonest way of computing optimal policies is by dynamic programming. We next discuss policies and see how to solve *finite horizon MDP* and *infinite horizon MDP* using the concept of dynamic programming.

2.4.1 Decision rule and policies

At each decision epoch, the *decision rule* is viewed as a rule that directs what actions to take in each state. The decision rules can be related to past history of decisions, all the way to the current state of the system (thus, history dependent). It is possible also that they may depend only on the current state of the system and do not take into consideration past histories, thus, Markovian. In a case where the decisions depend on the previous states only through the current states, they are said to be *Markovian (memoryless)*. The decision rules may also be *deterministic* if and only if actions in the states are chosen with probability of one. A deterministic rule chooses an action with certainty. However, the decision rules may be *probabilistic* if the possible actions are chosen based on a defined probability distribution. Here, there is a probability distribution over possible actions that is used to randomly take an action for each state. A *policy* is defined as a collection of decision rules that provides the decision maker a prescription of what action to take in any given state and at any decision epoch. It is denoted as π. Thus;

$$\pi = (d_1, d_2, ..., d_{n-1}), \forall d_t \in D_t, t \in T$$

In a unique case where we have $\pi = (d, d, ...d)$, such a policy is termed as a *stationary policy*. A deterministic policy is a policy where the decision rules are deterministic. A Markovian policy is a policy where the decision rule for each decision epoch is Markovian. A policy that is Markovian and deterministic can be shown to always exist under very *mild assumptions*. For more elaborate work on this, see Dubins & Savage (1965) and Blackwell (1965).

2.4.2 Optimal policy (Finite-horizon MDP models)

A policy or strategy specifies the decision rule to use at all the decision epochs of the system. The agent or the decision maker's ultimate goal is to find a policy that results in the accrual of maximum total rewards over the length of the horizon. We call this an

optimal policy, usually denoted as π^*. Let's consider a policy π over a finite horizon of length *N* with a beginning state *s*. Then, we can compute;

$$v_N^{\pi}(s) = E_s^{\pi}\left[\sum_{t=1}^{N-1} r_t(X_t, Y_t) + r_N(X_N)\right] \tag{2.3}$$

where:

- X_t is a sequence of random variables representing the progression of the state,
- Y_t is a sequence of random variables representing the progression of the actions taken at each decision epoch based on the policy π,
- the term $r_N(X_N)$ represents the salvage value of being in state X_N at the end of the planning horizon,
- v_N^{π} is the *value function* of the policy π.

Consider equation (2.3). To find the optimal policy π^*, we may find π^* such that,

$$v_N^{\pi^*}(s) \geq v_N^{\pi}(s) \tag{2.4}$$

for all starting states *s* and for all alternative policies π. Quite clearly, equation (2.4) appears cumbersome to solve, especially if the time horizon is very long. This is particularly where dynamic programming comes into play. We solve the problem using the *bottom-up approach*, that is to say backward-induction. To do this, we start from the last period, $N-1$ and work backwards to the first period, $t=1$. Let us define as $u_t^{\pi}(s)$ as the total reward achieved from period *t* onward, then:

$$u_t^{\pi}(s) = E_s^{\pi}\left[\sum_{t=1}^{N-1} r_t(X_t, Y_t) + r_N(X_N)\right] \tag{2.5}$$

Let us also define the function $u_t^*(s)$ as the function that achieves the maximum possible rewards if we are in state *s* at the decision epoch *t*, then we can recursively determine

$u_t^*(s)$ for each state by solving the equations:

$$u_t^*(s) = \max_{a \in A(s)} \left\{ r_t(s,a) + \sum_{j \in S} p_t(j|s,a) u_{t+1}^*(j) \right\} \tag{2.6}$$

The formulation (2.6) is very famous in sequential decision theory, Bellman & Dreyfus (2015). It is popularly known as the *Bellman optimality equations*, named after one of the founding fathers in MDP models, *Richard Bellman*. For the backward-induction method to be used, we need to define a starting point. If the model we are dealing with is a finite horizon model, then the problem become easy to solve. This is because, we would know exactly where to start. Thus, at $N-1$. No decision would be needed at the final decision epoch N. It is treated as a *salvage point* and considered in the final evaluation of the system. As such,

$$u_N^*(s) = r_N(s)$$

for all the states s. We can therefore use the recursive equation (2.6) to solve for $u_{N-1}^*(s), ..., u_2^*(s), u_1^*(s)$. The optimal decision at any given epoch and for any given state is an action A_s, such that:

$$a \in \arg\max_{a \in A(s)} \left\{ r_t(s,a) + \sum_{j \in S} p_t(j|s,a) u_{t+1}^*(j) \right\} \tag{2.7}$$

We now describe below, the *bottom-up* induction algorithm for (2.7). The enumeration is as follows:

Step I. Begin by setting $t = N$ and $u_N^*(s) = r(s), \forall s \in S$

Step II. Set t to $t-1$, compute for each s in the state space

$$u_t^*(s) = \max_{a \in A(s)} \left\{ r_t(s,a) + \sum_{j \in S} p_t(j|s,a) u_{t+1}^*(j) \right\}$$

Step III. For each s in the state space, find an action $a \in A_s$, such that;

$$a \in \arg\max_{a \in A(s)} \left\{ r_t(s,a) + \sum_{j \in S} p_t(j|s,a) u^*_{t+1}(j) \right\} \tag{2.8}$$

Step IV. If $t = 1$, stop else return to step *II*

The function $u^*_t(s)$ is the maximum reward earned (for maxima problems) over the planned period. Clearly, the optimality policy can be obtained from the actions *a* in step 3. The decision maker can use this optimal policy as a guide for optimal action for every state and every decision epoch. As an example, we may consider an inventory problem where placement of orders is affected by both fixed cost and variable cost. Applying the backward induction method for finding optimal policy leads us to the well known (s_t, S_t) *policy* in inventory management systems, White (1993).

2.4.3 Optimal policy (Infinite-horizon MDP models)

In the previous section, we saw that using the backward induction method to solve finite horizon problem is not a daunting task. We easily determine the starting point, $u^*_t(s)$. In most sequential decision processes, the horizon of the model is not finite because the decision process of the system is on-going. This means that N goes to *infinity* which makes the starting point for a backward induction algorithm very difficult one. Luckily enough, the fact that we are confronted with infinite number of decision epochs does not necessarily mean optimal solutions do not exist. They do exist sometimes. For a great number of infinite-horizon models, the optimal solution turns out to be independent of the decision epochs. This implies that, transition probabilities and the rewards are independent of *t*. In this section, we restrict ourselves to policies that do not change from one decision epoch to another decision epoch, *i.e. stationary policies*. From the perspective of infinite-horizon setting, let us take a look at the equation (2.9) where a policy attempts to maximize expected

accrued rewards.

$$v_t^{\pi}(s) = E_s^{\pi}\left[\sum_{t=1}^{\infty} r_t(X_t, Y_t)\right] \tag{2.9}$$

Trying to find an optimal policy to (2.9) may pose problems on three fronts, White (1993).

i. It may not exist.

ii. May exist but fails to achieve maximum

iii. May be infinite for a number of different policies that do not achieve our objective.

How then do we overcome this problem? One way of dealing with it is to discount the future rewards. In other words, we incorporate a discounting reward factor into the model and seek a policy that maximizes long run rewards for each $s \in S$. The discounting factor is denoted by $\beta \in (0, 1)$.

$$v_t^{\pi}(s) = E_s^{\pi}\left[\sum_{t=1}^{\infty} \beta^{t-1} r_t(X_t, Y_t)\right] \tag{2.10}$$

Provided all rewards are bounded, the infinite sum (2.10) is guaranteed to converge. This is an advantage of (2.10) over (2.9). For an optimal stationary, Markovian and deterministic policy to be guaranteed, we make the following necessary assumptions:

i. all rewards and transition probabilities are independent of *t*,

ii. all rewards are bounded,

iii. future rewards are discounted and,

iv. the state space is finite or countable.

Whiles there exists a number of methods for determining optimal policy for infinite-horizon MDPs, the commonest methods are: *policy iteration*, *value iteration* and *linear programming*. Another way handling this is to obtain a policy π that maximizes average

reward for every $s \in S$. This particularly becomes necessary if it does not make sense to discount future rewards. The function to do that is stated as:

$$g^{\pi} = \lim_{N\to\infty} \frac{1}{N} v^{\pi}_{N+1}(s) = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} P^{n-1}_{\pi} r_{d_n} \tag{2.11}$$

Note that, there is no guarantee that the average reward function shown in equation (2.11) will exist. If the policy is stationary, however, (2.11) is guaranteed to exist.

2.5 Worked example: Inventory control problem

In this section, we briefly give a typical inventory problem and go ahead to solve the problem using the concepts herein presented. This problem is taken from the book: *Operations Research: Applications and Algorithm*, Winston & Goldberg (2004).

Question:

"A warehouse has an end-of-period capacity of 3 units. During a period in which production takes place, a setup cost of \$4.00 is incurred. A \$1.00 holding cost is assessed against each unit of a period's ending-inventory. Also, a variable production cost of \$1.00 per unit is incurred. During each period, demand is equally likely to be 1 or 2 units. All demands must be met on time, and $\beta = 0.80$. The goal is to minimize expected discounted costs over an infinite horizon. Use *policy iteration* to determine an *optimal stationary policy*" Winston et. al (2004).

2.5.1 Suggested solution

We solve this problem sequentially as shown in the steps below;

Step I. *Given parameters:*

The given parameters are;

Set-up cost = \$4.00

variable cost = $1.00

maximum allowed inventory = 3 units

chance of demand in time t = 1 or 2 units

Step II. *State space:*

Since the maximum allowed inventory is 3 units, the inventory level at any time is between 0 and 3 units.

Let the state variable be the inventory level at time, t.

State space, $S = \{0, 1, 2, 3\}$

Step III. *Decision space:*

During each period, the number of units to produce is decided.

Hence, the action space, $A = \{0, 1, 2, 3, 4\}$

$X_t = 0, \Rightarrow$ 0 units are produced

$X_t = 1, \Rightarrow$ 1 unit is produced

$X_t = 2, \Rightarrow$ 2 units are produced, etc

Step IV. *Transition probabilities:*

Demands are stochastic, i.e. 1 or 2 units.

ending inventory = initial inventory - demand during time t.

Assuming we start with an initial inventory of 0 and produce 2 units, then after demand is met, the ending inventory is either 1 or 0 with a probability of 0.50.

Let *i* be the ending inventory with demand of 1 or 2 units, then;

$\forall X_t = 0$;

$$p(I_{i-1}|I_i, a_t = 0) = 0.50$$

$$p(I_{i-2}|I_i, a_t = 0) = 0.50$$

$\forall X_t = 1$;

$$p(I_i|I_i, a_t = 1) = 0.50$$

$$p(I_{i-2}|I_i, a_t = 1) = 0.50$$

$\forall X_t = 2$;

$$p(I_{i+1}|I_i, a_t = 2) = 0.50$$

$$p(I_i|I_i, a_t = 2) = 0.50$$

$\forall X_t = 3$;

$$p(I_{i+2}|I_i, a_t = 3) = 0.50$$

$$p(I_{i+1}|I_i, a_t = 3) = 0.50$$

$\forall X_t = 4$;

$$p(I_{i+3}|I_i, a_t = 4) = 0.50$$

$$p(I_{i+2}|I_i, a_t = 4) = 0.50$$

Step V. *Production cost, c(x):*

x is the units produced to meet demand

Hence, *c(x)* $= 4 + x$

By definition, *c(0) = 0*.

Step VI. *Discounted costs:*

Let $V_\delta(i)$ be the long run costs over an infinite time t. Then by using recursive (2.12), we obtain value determination equations under a stationary policy.

$$V_\delta(i) = r_{i,\delta(i)} + \beta \sum_{j=1}^{\infty} p[j|i, \delta(i)] V_\delta(j) \tag{2.12}$$

β is the discount factor, 0.80

$r_{i,\delta(i)}$ is the cost from the beginning of the period under the decision $\delta(i)$

Step VII. *Stationary policy:*

To obtain the value determination equations, we use the following policy and test to see if they are optimal.

$\delta(0) \Rightarrow a_t = 4$; produce 4 units

$\delta(1) \Rightarrow a_t = 3$; produce 3 units

$\delta(2) \Rightarrow a_t = 0$; produce 0 units

$\delta(3) \Rightarrow a_t = 0$; produce 0 units

Step VIII. *Production cost, Holding cost and Total cost:*

When the state is $s = 0$, 4 units are produced

Production cost, *c(x)* $= 4 + 4$ = \$8.00

Average holding cost, *h(x)* $= \frac{3+2}{2}$ = \$2.50

Total cost, *T(x)* $= 8.00 + 2.50 = \$10.50$

Similarly, when the state is $s = 1$, 3 units are produced

Production cost, *c(x)* $= 4 + 3$ = \$7.00

Average holding cost, *h(x)* $= \frac{3+2}{2}$ = \$2.50

Total cost, *T(x)* $= 7.00 + 2.50 = \$9.50$

In state $s = 2$, *T(x)* $= \$0.50$

In state $s = 3$, *T(x)* $= \$1.50$

Step IX. *Value Determination Equations:*

Using equation (2.12), we obtain:

$$V_\delta(0) = 10.50 + 0.80\Big[0.5V_\delta(3) + 0.5V_\delta(2)\Big]$$

$$V_\delta(1) = 9.50 + 0.80\Big[0.5V_\delta(3) + 0.5V_\delta(2)\Big]$$

$$V_\delta(2) = 0.50 + 0.80\Big[0.5V_\delta(1) + 0.5V_\delta(0)\Big]$$

$$V_\delta(3) = 1.50 + 0.80\Big[0.5V_\delta(2) + 0.5V_\delta(1)\Big]$$

Solving the above equations simultaneously, we obtain:

$V_\delta(0) = 32.70$

$V_\delta(1) = 31.70$

$V_\delta(2) = 29.50$

$V_\delta(3) = 26.00$

Next, we use Howard's policy iteration to test for policy optimality. The iteration is achieved by using:

$$T_\delta(i) = \min \left\{ r_{i,\delta(i)} + \beta \sum_{j=1}^{\infty} p[j|i, \delta(i)] V_\delta(j) \right\} \tag{2.13}$$

$$T_\delta(I_1) = min \begin{cases} 5.5 + 0.8\left[0.5V_\delta(1) + 0.5V_\delta(0)\right] = \mathbf{31.26} \\ 7.5 + 0.8\left[0.5V_\delta(2) + 0.5V_\delta(1)\right] = 31.90 \\ 9.5 + 0.8\left[0.5V_\delta(3) + 0.5V_\delta(2)\right] = 31.70 \end{cases}$$

$$T_\delta(2) = min \begin{cases} 0.5 + 0.8\left[0.5V_\delta(1) + 0.5V_\delta(0)\right] = \mathbf{26.26} \\ 6.5 + 0.8\left[0.5V_\delta(2) + 0.5V_\delta(1)\right] = 30.90 \\ 8.5 + 0.8\left[0.5V_\delta(3) + 0.5V_\delta(2)\right] = 29.70 \end{cases}$$

$$T_\delta(3) = min \begin{cases} 1.5 + 0.8\left[0.5V_\delta(2) + 0.5V_\delta(1)\right] = \mathbf{26.00} \\ 7.5 + 0.8\left[0.5V_\delta(3) + 0.5V_\delta(2)\right] = 29.70 \end{cases}$$

We observe that, for each state *i*, $T_\delta(i) = V_\delta(i)$.

Therefore, an **optimal solution** is obtained by producing:

4 units when inventory is 0

3 units when inventory is 1

0 units when inventory is 2 or 3

2.6 Limitations of Markov decision processes

An MDP is both powerful and efficient modeling approach to handle uncertainty in sequential decisions. In spite of the diverse and appealing applications of Markov

decision processes, in many applications, the MDP theory faces a major challenge. Bellman coined the term *curse of dimensionality*, Bellman & Dreyfus (2015). This term has become very famous in sequential decision sciences. By far, the ”curse of dimensionality” has become a major challenge to MDP theorists. Sometimes, the size of the *state space* increasingly becomes too large to make the computing of the model possible, not even to available modern computational tools, . In recent decades, a field of research called approximate dynamic programming (ADP) has developed in an attempt to solve larger MDP models, Sturmberg & Martin (2013). Bethke et al. (2008) further elaborates on this limitations.The fundamental idea is to assume that the value function has a given functional form that can be characterized by a reasonable number of parameters. Thus, rather than seeking to present a look-up table that outlines the optimal action for every state, the ADP methodology seeks to find the optimal parameter values that characterize the assumed form of the value function in order to get the best approximation possible. Current research is geared towards methods that are more efficient in terms of the memory of the iteration.

2.7 Summary

In this chapter, we introduced MDPs; a powerful and efficient modelling approach to handle uncertainty in sequential decisions. We identified and described the five key components of an MDP model. We also discussed how MDP problems are solved, in particular *finite* and *infinite-horizon MDP* models. In discussing finite-horizon models, we pointed out the existence of optimal policy by method of backward induction. In the case of infinite-horizon model, we mentioned the difficulty encountered when the decision epoch is infinite and discussed a modified policy by incorporating a discounting factor. We also mentioned the three most commonly used methods for finding optimal policy for infinite-horizon MDP models; policy iteration, value iteration and linear programming. Briefly, we discussed the limitations of MDPs in some applications, a major challenge being the *”curse of dimensionality”*. Finally, we close this chapter by highlighting an inventory problem and solving it.

CHAPTER THREE
Q-LEARNING (REINFORCEMENT)

3.1 Background

In Chapter 2 (refer to page 18), we saw that for a typical sequential decision problem, the *optimal policy* can be achieved by various approaches (e.g. Dynamic programming). Other than these approaches, *Reinforcement Learning* algorithm maybe adopted. One type of *RL algorithm* that can be used to solve for an optimal policy is *Q-learning*. This chapter mainly discusses *Q-learning* as an alternative to dynamic programming. The main goal of the chapter is to present an introductory concept of Q-learning, and by extension, link it with the application in chapter 4.

3.2 Reinforcement learning

Sutton et al. (1998) has described Reinforcement learning one of the approaches where sequential decision problem (complex) could be modeled. This area of machine learning is actually a learning framework, where the decision maker (who we referred here as *controller*) has a goal of achieving maximum future rewards, as it interacts with the environment. The interaction with the environment is somewhat done in a trial-and-error fashion. The principal question is how should actions be chosen to ensure that maximum rewards are accumulated along the planning horizon. The popularity of RL is tremendous in telecommunications, robotics, computer games, inventory control systems, etc. For a comprehensive research on the core applications of RL, the reader is recommended to see Li (2017), Sutton et al. (1998). Like an MDP in earlier discussion, a standard Reinforcement Learning setting can be described as a *four-tuple* consisting of: *a finite set of states, a finite set of actions, a reward function, and a transition model*.

3.3 Policy

The decision maker/agent intends to collect as much as possible rewards whiles progressively interacting with the environment. An action performed in s_t causes the state to evolve into another state s_{t+1} with a corresponding reward. The goal to accruing maximum rewards is at the core of the question, *"For each state, what action need be taken?"*. A strategy or a contingency plan which specifies the decision rule to use at all the decision epochs in the learning environment is called *policy*. A policy in which the action is taken with *certainty* is called a *deterministic policy*, denoted as:

$$\pi(s_t) = a_t, \;\; s_t \in S, \;\; a_t \in A$$

Where the S and A are respectively the state and action space. On the other hand, if the actions are stochastic, then the policy takes the form of a probability distribution conditioned on the states s_t, represented as:

$$\pi(s_t|a_t) = p_i, \;\; s_t \in S, \;\; a_t \in A, \;\; 0 \leq p_i \leq 1$$

From all the possible policies in the reinforcement learning environment, the policy that realizes the most rewards is the one we would call *optimal*, denoted as π_*. An optimal policy converges and can be obtained using the *Bellman's optimality principle*, as seen in (2.6) of Chapter 2.

3.3.1 Value function

For $s_t \in S$, a value function assesses the usefulness of a policy. A *policy*, π is assessed useful by virtue of discounted sum rewards accrued in the learning environment. Let us define our V^π is value function. Using "trial-and-error" method, the value function can be estimated from varying samples of a given task. One advantage of this value function is that, it can be calculated using the recursive

formulation. Sutton et al. (1998) have shown that it can be recursively expressed as:

$$\begin{aligned} V^{\pi}(s) = E_{\pi}\Big\{R_t|s_t = s\Big\} &= E_{\pi}\Big\{\sum_{i=0}^{\infty}\beta^i r_{t+i+1|s_t=s}\Big\} \\ &= E_{\pi}\Big\{r_{t+1} + \sum_{i=0}^{\infty}\beta^i r_{t+i+1|s_t=s}\Big\} \end{aligned} \tag{3.1}$$

In the case of a stochastic policy π, equation (3.1) 'decomposes' further into the *Bellman equation* of V^{π} Sutton et al. (1998):

$$V^{\pi}(s) = \sum_{a}\pi(a|s)\sum_{s_{t+1}}p(s_{t+1}|s_t,a)\Big\{r(s,a|s_{t+1}) + \beta V^{\pi}(s_{t+1})\Big\} \tag{3.2}$$

3.3.2 Quality function

The quality function, also referred to as *Q-value* is also one approached used to obtain *near-to-optimal* policy. It does so by estimating the quality of accrued rewards when a certain policy is adapted, through applying an action to a particular state (to be discussed under Q-learning) in the next section. It comprises the long run rewards of taking actions in the states and then following the considered policy in subsequent decision encounters such that Sutton et al. (1998):

$$\begin{aligned} Q: S\times A \to R \;\; Q(s,a)^{\pi} &= E_{\pi}\Big\{R_t|s_t = s, a_t = a\Big\} \\ &= E_{\pi}\Big\{\sum_{i=0}^{\infty}\beta^i r_{t+i+1|s_t=s,a_t=a}\Big\} \end{aligned} \tag{3.3}$$

3.4 Q-learning

Let us consider an environment where the controller continuously interacts with the system by learning the actions needed to be taken in order to earn rewards, or by actions which result in it being punished. In the reinforcing a learning process, emphasis are laid on three key components, viz; states, actions and rewards. Here, the knowledge of transition probabilities is completely 'eclipsed', and as a result of that, the learning

process does not require a model. *Q-learning* can be seen as a type of *RL algorithm* that seeks to find the *Q-value*, $Q(s, a)$ for any $s \in S$ and $a = \pi(s)$, (see section 3.3.2). Whiles in the environment, the *Q-learning* approach begins with an initial guess for $Q(s, a)$ for all the states and actions. Next, it proceeds to update them, based on the iterative formulation (3.4).

$$\begin{aligned} Q(s_t, a_t) =& (1 - \alpha_t)Q(s_t, a_t) + \alpha_t\Big(r_{t+1} + \beta \max_a Q(s_{t+1}, a)\Big) \\ & \forall t = 1, 2, \ldots \end{aligned} \tag{3.4}$$

where α_t is the learning rate at time step t such that $0 \leq \alpha \leq 1$. The learning rate helps in overriding old Q-value. β is the discount factor such that $0 \leq \beta \leq 1$, which decreases the Q-value for future states. Whiles in an observed state, the agent chooses an action through an ϵ-greedy algorithm: with probability ϵ_t in time t, the algorithm chooses an action randomly, and with probability $1 - \epsilon_t$, it chooses the action with the highest cumulative action value, i.e, $a_{t+1} = \arg\max_a Q(s_{t+1}, a)$. The process of randomly choosing actions is called *exploration*. One advantage of this method is that, the solution spectrum can be explored and the optimality of the algorithm is guaranteed if $\epsilon_t \to 0$ and $t \to \infty$ Sutton et al. (1998). In finding optimal Q^*, we may recover the optimal policy as $\pi^*(s) = \arg\max_a Q^*(s, a)$.

Clearly, the principal idea of Q-learning is to estimate the action-value function with the help of the *Bellman equation*. It is done iteratively and converges to optimal as our iteration tends to infinity. The Q-learning algorithm can be summarized in the following pseudo-code.

3.5 Summary

In this chapter, we briefly discussed Reinforcement learning as an alternative approach to finding optimal policy in sequential decision problems. Other than using dynamic programming, linear programming, etc., we pointed out that Q-learning (a type of *RL*) can be used to determine optimal or near-optimal policies. The chapter

Algorithm 1 *Q-learning mechanism*

Initialize $Q(s, a) = 0$ or arbitrarily for all $s \in S$ and $a \in A$
Repeat until the end of the episode:
$s_t \leftarrow$ initial state
for each episodic step do
Select a_t**, based on exploration strategy from** s_t
Take action a_t**, observe** r_{t+1}, s_{t+1}
$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\Big(r_{t+1} + \beta \max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\Big)$
$s_t \leftarrow s_{t+1}$
if then $s ==$ terminate **then**
$Q(s_{t+1}, a_{t+1}) = 0$
end if
end for

also briefly highlighted on the value and quality function, and concluded by presenting summarized Q-learning (off-policy) algorithm.

CHAPTER FOUR
CASE STUDY: PRODUCTION SCHEDULING

4.1 Background

In this chapter, we seek to elaborate on the application of sequential decision theory to a real inventory control problem. The key goal of this application is to find an *optimal production scheduling* of limestones; a case study in a limestones production factory in Trakya, Turkey. We deal with the problem by using the concepts studied in the preceding chapters, in order to characterize the problem as a *Markov decision process*. In our study, we restrict ourselves to only *stationary policies*, by evaluating various policies with respect to the decision rules. This is done by using a policy iteration method, and thereafter, using Howard's policy to check for *optimality*. We conclude the chapter by giving computational results from *Mathematica* software.

4.2 Problem description and assumptions

4.2.1 Problem description

The application part of this study focuses on an international limestone production firm. The firm has *processing plants* where there is a large scale processing of limestones. The firm has many factories in different locations. In each location, there are four plants where limestones are processed, stored and sold in tonnes. It is known that demand in a given period is stochastic. Since demand requirements are stochastic from period to period, it is often economical to produce more limestones than needed, and store the excess quantity to meet future demands. In the course of production, limestones are processed in a number of furnaces. In our application, we take a case study of one limestone production factory in Trakya (a region in the northwestern part of Turkey). Due to the factory's strict policy of confidentiality of *demand data*, the factory has chosen to share neither demand nor cost information. As a result of that,

demand data has been simulated by using the *mean* and *variance* values they declared. Also, cost data has been generated by using the information the firm shared. Table 4.1 summarizes the production capacity during operation, whilst Table 4.2 elaborates on the costs associated with holding unit inventory. The data on weekly random demand is shown in Table A.1 and Table A.2 of Appendix A.1. Additionally, the distribution of weekly demand is shown in Table 4.3.

Table 4.1 Production and storage capacity

	Tonnes	**Kilogrammes**	**Bags (1 bag = 25kg)**	**Batches (1 batch = 1000kg)**
Storage capacity	500	500000	20000	20
Production capacity	700	700000	28000	28

Table 4.2 Production and storage costs

Total cost:		**Unit selling price/ton**	**Weekly interest**	**Weekly unit cost**
Production cost (90%)				15.31
Holding cost (10%)	0.30TL/KG	300	0.0057	1.70

Table 4.3 Probability distribution (weekly demand)

$D \times 10^3$/ton	D/batch	$P(D = d)$
$585 < D < 625$	26	0.0377
$625 < D < 665$	27	0.0609
$665 < D < 705$	28	0.0874
$705 < D < 745$	29	0.1124
$745 < D < 785$	30	0.1293
$785 < D < 825$	31	0.1486
$825 < D < 865$	32	0.1323
$865 < D < 905$	33	0.1105
$905 < D < 945$	34	0.0741
$945 < D < 985$	35	0.0585
$985 < D < 1025$	36	0.0285
$1025 < D < 1065$	37	0.0198

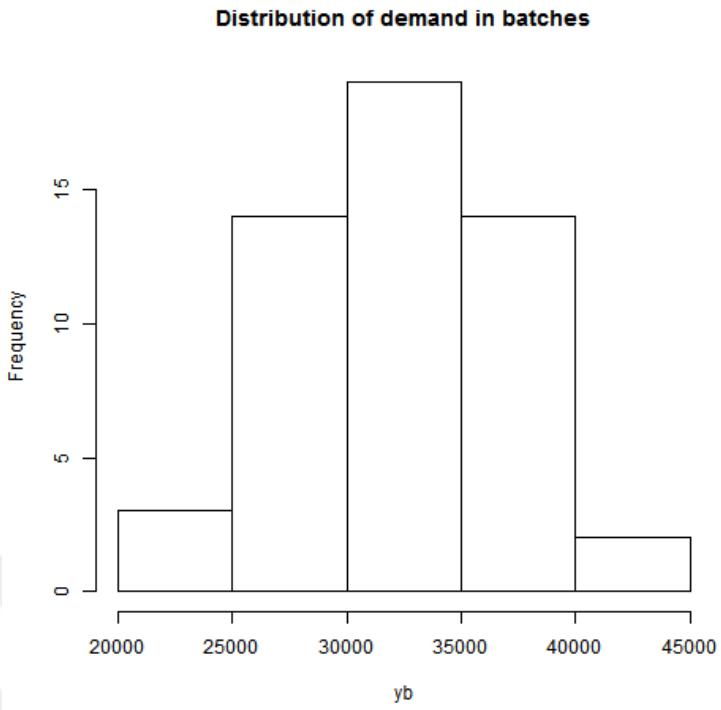

Figure 4.1 Distribution of demand of limestones in tonnes

4.2.2 Model assumptions and sequence of events

In determining the production schedule for the next N weeks, we restrict our model to a single processing site, which is a factory located in Trakya. We assume the 4 furnaces process limestones continuously throughout the year, and operate independently of one another during operations. Each furnace processes on the average 250 tonnes per day. Instead of separately considering the average processing rate for each furnace, we aggregate all the 4 averages into a single measure. This enable us formulate the problem as a single item inventory system with periodic review. Each furnace works 330 days in a year. The remaining 1 month is allocated for maintenance and repairs. Excess production is stored in a storage area which has a capacity of 28 batches. Further assumptions of the model are summarized as follows:

i. Demands for limestones are stochastic during the planned period;

ii. The factory processes more limestones than necessary, such that excess amount is stored for future time;

iii. Demands are filled before the beginning of the next time period;

iv. Fluctuations in the cost of fuel, electricity and labour are negligible throughout the 52 week period.

Throughout the system, the associated sequence of events are:

i. The system's state (inventory level) for period t is reviewed;

ii. Based on the system state, the production amount for that period is determined;

iii. The demand of the period is realized;

iv. The cost incurred is assessed.

4.3 Model formulation

In order to attempt for optimal policy, the problem is formulated as a *discrete-time MDP*. The planning horizon is considered infinite. The formulation of the problem as a Markov decision process is explained below. We consider time in discrete periods, which is examined weekly.

4.3.1 Model parameter and definitions

Table 4.4 Model notation and their definitions

Notations		**Definitions**
Sets	S	Sets of all system states, representing the inventory level (measure in tonnes or batches)
	A	Sets of action rules, $A = \{a_1, a_2, \ldots, a_N\}$
	T	Time periods with $t \in \{1, 2, \ldots, \tau\}, \tau < \infty$
State variables	I_t	Available inventory at time t, measured in tonnes/batches
Random variables	X_t	Production amount of limestones during time t
	Y_t	progression of the actions taken at each decision epoch
	D	Demand of limestones measured in tonnes/batches
Cost function	$R(s, a)$	Cost of holding s inventory, given an action a

4.3.2 Objective function

The key objective of this application is to obtain an optimal inventory policy that minimizes the weekly demand for an infinite planning horizon. To achieve the goal we use dynamic programming approach described as described in equation (2.6) (page 16), (for more, see Smith (1996), Ross (2014), Shapiro (1993)). Let us define the function $u_t^*(s)$ as the function that achieves the minimum cost if we are in state s at the decision epoch t, then we can recursively determine $u_t^*(s)$ for each state by solving the equations:

$$u_t^*(s) = \min_{a \in A(s)} \left\{ r_t(s,a) + \sum_{j \in S} p_t(j|s,a) u_{t+a}^*(j) \right\}$$

The objective is to find an optimal policy $\pi^* \in \Pi$ such that;

$$v_N^{\pi^*}(s) \leq v_N^{\pi}(s)$$

4.3.3 Decision epoch

Production decisions are made on weekly basis before the beginning of the production next week. This production decisions continue throughout the fifty-two week period. The model here is a *infinite time horizon* model with a finite set of decision epochs. Thus,

$$T = \{1, 2, \ldots, \tau\}, \tau < \infty$$

4.3.4 State of the system

Since the state variable expresses the system description in a given time period, it is defined as the *entering inventory*. Let state variable I_t represent the inventory level (in batches) at the beginning of time period t. The inventory level in period t depends only on the previous period; it does not depend on the previous history of the system states. Hence I_t describes a Markov chain with discrete state space and the state variable may

take values in discrete state space $S = \{0, 1, \ldots, 20\}$.

4.3.5 Actions

Actions describe the decisions to be made in the corresponding time periods. Decisions are made by considering the system state I_t. For our problem, inventory level on hand is reviewed and a decision is made relative to the production amount in the corresponding period. Let the decision variable X_t represent the production amount to be made in period t after reviewing inventory position. Production is made large enough so that stock-outs do not occur. To enforce this condition, it is ensured that the summation of the entering inventory and the production amount should be at least maximum demand occurrence in batches. Mathematically;

$$i + x \geq 37$$

As an upper bound, there is a production capacity given by the company. The production amount may be at maximum of *28 batches* in a week. On the other hand, as another restriction; the transferred amount $(i + x - d)$ may be 20 batches at most, else there may be more inventory than the maximum capacity on the storage (which is impossible).
Let S^*_{max} be the maximum storage capacity, D_{min} be the minimum demand and P be the production capacity. We may write;

$$i + x \leq S^*_{max} + D_{min}$$

$$\Rightarrow i + x \leq 46$$

In this case, the restriction for the production amount is expressed as:

$$D_{max} - i \leq x \leq \min\Big\{P, (S^*_{max} + D_{min} - i)\Big\}$$

$$\Rightarrow 37 - i \leq x \leq \min\Big\{(28, 46 - i)\Big\}$$

4.3.6 Transition probabilities

Transition probabilities are determined by using the demand distribution and the inventory position. Refer to Table 4.3 for the demand distribution. After specifying the state space of the system, the actions (production amount) influences the evolution of the next state. Table A.3 and Table A.4 of Appendix A.1 show the transition probabilities. The events that cause a transition from one state to the next are:

i. The inventory level of the limestones I_t at the start of the weeks. Thus, how many batches of limestones are available at the warehouse.

ii. Production amount X_t given that the entering inventory is i.

4.3.7 Cost function

Expected total cost is calculated weekly and has two main components, namely: production and holding cost. Let us consider $c(x)$ be the expected cost in the current period as a result of making decision I_t. In other words, $c(x)$ is the cost of processing $x_t \in X_t$ units of limestones. Given that the unit production cost is 15 units per week, and h, the unit weekly holding cost. It is calculated based on annual interest rate, and found to be 2 units. To find the expected holding cost, the unit holding cost is multiplied by average inventory level for period t. The expected demand in batches is thus computed as:

$$\begin{aligned} E(D) &= \mu_D \\ &= 26 \times p(D = 26) + \ldots + 37 \times p(D = 37) \\ &= 32.10 \end{aligned}$$

For an inventory level i and production amount x, the total cost for the current period is:

$$c_{ix} = c(x) + h(i + x - \mu_D)$$

We define the minimum expected net cost incurred during the periods $t, t+1, \ldots$ to be $V_t(i)$, given that the inventory at the beginning of period t is i units and a unit holding cost of 1.70₺. For an infinite horizon, the appropriate recursive dynamic programming formulation (MDP) would be;

$$V_\delta(i) = \min \left\{ c(x) + h(i + x - \mu_D) + \beta \sum_{j \in S}^{N} p(j|i, d) V_\delta(j) \right\} \tag{4.1}$$

The minimization is over only nonnegative integer values, in the range:

$$37 - i \leq x \leq \min\{(28, 46 - i)\}$$

Since stockouts are not allowed, demand is met only when entering inventory and production amount are at least 37 batches. Thus, $i + x \geq 37$ or $x \geq 37 - i$. We know the firm's production capacity to be 28 batches and the maximum demand to be 37 batches (Refer to Table 4.3). Note also that the period of each ending inventory must be $i_t > 0$, $i_t \leq 20$. This means that during each period, $i_t \in \{0, 1, 2, \ldots, 20\}$.

4.4 Policy iteration (Computational approach)

As discussed in Chapter 2, three methods are mostly used to determine optimal policies (*policy iteration, linear programming* and *value iteration*). In our attempt to find an optimal policy to this production problem, we resort to the use of policy iteration, discussed below.

The state space of our production problem is $i \in S = \{0, 1, 2, \cdots, 20\}$. Let's define δ to be any *stationary policy* and $\delta_{(i)}$ to be the production amount chosen by the stationary policy for the inventory i. We also define $V_\delta(i)$ as a policy which minimizes production and holding costs for the i^{th} entering inventory position. To obtain value determination equations for each of $i \in S$, we adopt the following *stationary policy* (i.e. how many batches of limestones to produce?):

$\forall i \in \{0, 1, 2, \ldots, 9\},$ produce $X_t = 28$ batches;

$\forall i = 10,$ produce $X_t \in \{27, 28\}$

$\forall i = 11,$ produce $X_t \in \{26, 27, 28\}$

$\forall i = 12,$ produce $X_t \in \{25, 26, 27, 28\}$

$\forall i = 13,$ produce $X_t \in \{24, 25, 26, 27, 28\}$

$\forall i = 14,$ produce $X_t \in \{23, 24, 25, 26, 27, 28\}$

$\forall i = 15,$ produce $X_t \in \{22, 23, 24, 25, 26, 27, 28\}$

$\forall i = 16,$ produce $X_t \in \{21, 22, 23, 24, 25, 26, 27, 28\}$

$\forall i = 17,$ produce $X_t \in \{20, 21, 22, 23, 24, 25, 26, 27, 28\}$

$\forall i = 18,$ produce $X_t \in \{19, 20, 21, 22, 23, 24, 25, 26, 27, 28\}$

$\forall i = 19,$ produce $X_t \in \{18, 19, 20, 21, 22, 23, 24, 25, 26, 27\}$

$\forall i = 20,$ produce $X_t \in \{17, 18, 19, 20, 21, 22, 23, 24, 25, 26\}$

Using equation (4.1) and the transition probabilities in Appendix A.2, we form a system of twenty-one linear equations as shown below;

$$V_\delta(0) = 421.71 + 0.9\Big[0.9014V_\delta(0) + 0.0609V_\delta(1) + 0.0377V_\delta(2)\Big]$$

$$V_\delta(1) = 423.41 + 0.9\Big[0.814V_\delta(0) + 0.0874V_\delta(1) + 0.0609V_\delta(2) + 0.0377V_\delta(3)\Big]$$

$$V_\delta(2) = 425.11 + 0.9\Big[0.7016V_\delta(0) + 0.1124V_\delta(1) + 0.0874V_\delta(2) + 0.0609V_\delta(3) + 0.0377V_\delta(4)\Big]$$

$$V_\delta(3) = 426.81 + 0.9\Big[0.5723V_\delta(0) + 0.1293V_\delta(1) + 0.1124V_\delta(2) + 0.0874V_\delta(3) + 0.0609V_\delta(4) + 0.0377V_\delta(5)\Big]$$

$$V_\delta(4) = 428.51 + 0.9\Big[0.4237V_\delta(0) + 0.1486V_\delta(1) + 0.1293V_\delta(2) + 0.1124V_\delta(3) + 0.0874V_\delta(4) + 0.0609V_\delta(5) + 0.0377V_\delta(6)\Big]$$

$$V_\delta(5) = 430.21 + 0.9\Big[0.2914V_\delta(0) + 0.1323V_\delta(1) + 0.1486V_\delta(2) + 0.1293V_\delta(3) + 0.1124V_\delta(4) + 0.0874V_\delta(5) + 0.0609V_\delta(6) + 0.0377V_\delta(7)\Big]$$

$$V_\delta(6) = 431.91 + 0.9\Big[0.1809V_\delta(0) + 0.1105V_\delta(1) + 0.1323V_\delta(2) + 0.1486V_\delta(3) + 0.1293V_\delta(4) + 0.1124V_\delta(5) + 0.0874V_\delta(6) + 0.0609V_\delta(7) + 0.0377V_\delta(8)\Big]$$

$$V_\delta(7) = 433.61 + 0.9\Big[0.1068V_\delta(0) + 0.0741V_\delta(1) + 0.1105V_\delta(2) + 0.1323V_\delta(3) + 0.1486V_\delta(4) + 0.1293V_\delta(5) + 0.1124V_\delta(6) + 0.0874V_\delta(7) + 0.0609V_\delta(8) + 0.0377V_\delta(9)\Big]$$

$$V_\delta(8) = 435.31 + 0.9\Big[0.0483V_\delta(0) + 0.0585V_\delta(1) + 0.0741V_\delta(2) + 0.1105V_\delta(3) + 0.1323V_\delta(4) + 0.1486V_\delta(5) + 0.1293V_\delta(6) + 0.1124V_\delta(7) + 0.0874V_\delta(8) + 0.0609V_\delta(9) + 0.0377V_\delta(10)\Big]$$

$$V_\delta(9) = 437.01 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(10) = 421.7 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(11) = 404.7 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(12) = 387.69 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(13) = 370.68 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(14) = 353.67 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(15) = 336.66 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(16) = 319.65 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(17) = 302.64 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(18) = 285.63 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(19) = 268.62 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

$$V_\delta(20) = 251.61 + 0.9\Big[0.01983V_\delta(0) + 0.0285V_\delta(1) + 0.0585V_\delta(2) + 0.0741V_\delta(3) + 0.1105V_\delta(4) + 0.1323V_\delta(5) + 0.1486V_\delta(6) + 0.1293V_\delta(7) + 0.1124V_\delta(8) + 0.0874V_\delta(9) + 0.0609V_\delta(10) + 0.0377V_\delta(11)\Big]$$

With *Mathematica* program, the solution to these equations yielded the following values for $V_\delta(0), V_\delta(1), V_\delta(2), \ldots, V_\delta(20)$, as shown in Table 4.5 (on page 42).

4.4.1 Howard's policy iteration

The above linear equations are solved. The solution yielded the values $V_\delta(0) =$ 4198.12₺, $V_\delta(1) =$ 4174.77₺, $\ldots, V_\delta(20) =$ 3561.38₺. We seek to check for *optimality* of our policy by using the Howard's Howard (1960) policy iteration method. We briefly describe the algorithm before applying it to our problem.

Step 1. Begin by formulating a stationary policy for the state space. This is called policy evaluation.

Step 2. Solve the value determination equations to determine $V_{(i)},\ \forall i = 1, 2, \ldots, N$.

Step 3. Check for optimality, otherwise, for the improvement of the policy in *Step .1*,

Table 4.5 Solution to value determination equations

i	x	$V_\delta(i)$	$T_\delta(i)$
0	28	4198.12	4198.12
1	28	4174.77	4197.76
2	28	4179.88	4179.87
3	28	4165.02	4165.40
4	28	3669.42	4145.04
5	28	4069.09	4124.70
6	28	4064.72	4106.07
7	28	4086.06	4092.57
8	28	4079.36	4079.76
9	28	4066.74	4066.80
10	27	4052.35	4060.41
11	27	3714.47	4006.89
12	26	3697.46	3975.59
13	25	3680.45	3930.21
14	24	3663.34	3883.18
15	23	3646.43	3831.96
16	22	3629.42	3786.08
17	21	3612.41	3749.80
18	20	3595.40	3717.73
19	19	3578.39	3700.72
20	18	3561.38	3683.71

compute for $i = 1, 2, \ldots, N$

$$T_\delta(i) = \min_{x \in X(i)} \left\{ r_{(ix)} + \beta \sum_{j \in S}^{j=N} p(j|i,x) V_\delta(j) \right\}$$

Considering $x = \delta(i) \in X(i)\ \forall i = 1, 2, \ldots, N$; if a stationary policy $x = \delta(i)$ is chosen which results in $T_\delta(i) \leq V_\delta(i)$ for one or more values of the state space, then the policy δ is not an **optimal policy**. As a result of that, modify δ to a new policy δ^* which attains minimum. This is done by returning to *Step. 1*. For this study, we stick to our initial stationary policy as our beginning stationary policy. We then test for policy optimality (by using Howard's method). Since there is only one choice for $V_\delta(0), V_\delta(1), V_\delta(2), \ldots, V_\delta(9)$, we do not compute their $T_\delta(i)$'s again. The

computations for the others are as follows;

$$T_\delta(10) = min \begin{cases} 421.70 + 0.9\Big[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\Big] = 4052.35 \\ 421.71 + 0.9\Big[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\Big] = \mathbf{4036.41} \end{cases}$$

$$T_\delta(11) = min \begin{cases} 406.39 + 0.9\Big[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\Big] = 4035.41 \\ 404.70 + 0.9\Big[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\Big] = 4019.40 \\ 421.71 + 0.9\Big[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\Big] = \mathbf{4006.89} \end{cases}$$

$$T_\delta(12) = min \begin{cases} 391.08 + 0.9\Big[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\Big] = 4020.10 \\ 387.69 + 0.9\Big[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\Big] = 4003.39 \\ 404.70 + 0.9\Big[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\Big] = 3989.88 \\ 421.71 + 0.9\Big[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\Big] = \mathbf{3975.59} \end{cases}$$

$$T_\delta(13) = min \begin{cases} 375.77 + 0.9\Big[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\Big] = 4004.79 \\ 370.68 + 0.9\Big[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\Big] = 3985.38 \\ 387.69 + 0.9\Big[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\Big] = 3972.87 \\ 404.70 + 0.9\Big[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\Big] = 3958.58 \\ 421.71 + 0.9\Big[0.01983V_\delta(4) + \ldots + 0.0377V_\delta(15)\Big] = \mathbf{3930.21} \end{cases}$$

$$T_\delta(14) = min \begin{cases} 360.46 + 0.9\Big[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\Big] = 3989.48 \\ 353.67 + 0.9\Big[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\Big] = 3968.37 \\ 370.68 + 0.9\Big[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\Big] = 3955.86 \\ 387.69 + 0.9\Big[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\Big] = 3941.57 \\ 404.70 + 0.9\Big[0.01983V_\delta(4) + \ldots + 0.0377V_\delta(15)\Big] = 3913.20 \\ 421.71 + 0.9\Big[0.01983V_\delta(5) + \ldots + 0.0377V_\delta(16)\Big] = \mathbf{3883.18} \end{cases}$$

$$T_\delta(15) = min \begin{cases} 345.15 + 0.9\left[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\right] = 3974.17 \\ 336.66 + 0.9\left[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\right] = 3951.36 \\ 353.67 + 0.9\left[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\right] = 3938.85 \\ 370.68 + 0.9\left[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\right] = 3924.56 \\ 387.69 + 0.9\left[0.01983V_\delta(4) + \ldots + 0.0377V_\delta(15)\right] = 3896.19 \\ 404.70 + 0.9\left[0.01983V_\delta(5) + \ldots + 0.0377V_\delta(16)\right] = 3866.17 \\ 421.71 + 0.9\left[0.01983V_\delta(6) + \ldots + 0.0377V_\delta(17)\right] = \mathbf{3831.96} \end{cases}$$

$$T_\delta(16) = min \begin{cases} 329.84 + 0.9\left[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\right] = 3958.86 \\ 319.65 + 0.9\left[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\right] = 3934.35 \\ 336.66 + 0.9\left[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\right] = 3921.84 \\ 353.67 + 0.9\left[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\right] = 3907.55 \\ 370.68 + 0.9\left[0.01983V_\delta(4) + \ldots + 0.0377V_\delta(15)\right] = 3879.18 \\ 387.69 + 0.9\left[0.01983V_\delta(5) + \ldots + 0.0377V_\delta(16)\right] = 3849.16 \\ 404.70 + 0.9\left[0.01983V_\delta(6) + \ldots + 0.0377V_\delta(17)\right] = 3814.95 \\ 421.71 + 0.9\left[0.01983V_\delta(7) + \ldots + 0.0377V_\delta(18)\right] = \mathbf{3786.08} \end{cases}$$

$$T_\delta(17) = min \begin{cases} 314.53 + 0.9\left[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\right] = 3943.55 \\ 302.64 + 0.9\left[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\right] = 3917.34 \\ 319.65 + 0.9\left[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\right] = 3904.83 \\ 336.66 + 0.9\left[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\right] = 3890.54 \\ 353.67 + 0.9\left[0.01983V_\delta(4) + \ldots + 0.0377V_\delta(15)\right] = 3862.17 \\ 370.68 + 0.9\left[0.01983V_\delta(5) + \ldots + 0.0377V_\delta(16)\right] = 3832.15 \\ 387.69 + 0.9\left[0.01983V_\delta(6) + \ldots + 0.0377V_\delta(17)\right] = 3797.94 \\ 404.70 + 0.9\left[0.01983V_\delta(7) + \ldots + 0.0377V_\delta(18)\right] = 3769.07 \\ 421.71 + 0.9\left[0.01983V_\delta(8) + \ldots + 0.0377V_\delta(19)\right] = \mathbf{3749.80} \end{cases}$$

$$T_\delta(18) = min \begin{cases} 299.22 + 0.9\left[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\right] = 3928.24 \\ 285.63 + 0.9\left[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\right] = 3900.33 \\ 302.64 + 0.9\left[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\right] = 3887.82 \\ 319.65 + 0.9\left[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\right] = 3873.53 \\ 336.66 + 0.9\left[0.01983V_\delta(4) + \ldots + 0.0377V_\delta(15)\right] = 3845.16 \\ 353.67 + 0.9\left[0.01983V_\delta(5) + \ldots + 0.0377V_\delta(16)\right] = 3815.14 \\ 370.68 + 0.9\left[0.01983V_\delta(6) + \ldots + 0.0377V_\delta(17)\right] = 3780.93 \\ 387.69 + 0.9\left[0.01983V_\delta(7) + \ldots + 0.0377V_\delta(18)\right] = 3752.06 \\ 404.70 + 0.9\left[0.01983V_\delta(8) + \ldots + 0.0377V_\delta(19)\right] = 3732.79 \\ 421.71 + 0.9\left[0.01983V_\delta(9) + \ldots + 0.0377V_\delta(20)\right] = \mathbf{3717.73} \end{cases}$$

$$T_\delta(19) = min \begin{cases} 283.91 + 0.9\left[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\right] = 3912.93 \\ 268.62 + 0.9\left[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\right] = 3883.32 \\ 285.63 + 0.9\left[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\right] = 3870.81 \\ 302.64 + 0.9\left[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\right] = 3856.52 \\ 319.65 + 0.9\left[0.01983V_\delta(4) + \ldots + 0.0377V_\delta(15)\right] = 3828.15 \\ 336.66 + 0.9\left[0.01983V_\delta(5) + \ldots + 0.0377V_\delta(16)\right] = 3798.13 \\ 353.67 + 0.9\left[0.01983V_\delta(6) + \ldots + 0.0377V_\delta(17)\right] = 3763.92 \\ 370.68 + 0.9\left[0.01983V_\delta(7) + \ldots + 0.0377V_\delta(18)\right] = 3735.05 \\ 387.69 + 0.9\left[0.01983V_\delta(8) + \ldots + 0.0377V_\delta(19)\right] = 3715.78 \\ 404.70 + 0.9\left[0.01983V_\delta(9) + \ldots + 0.0377V_\delta(20)\right] = \mathbf{3700.72} \end{cases}$$

$$T_\delta(20) = min \begin{cases} 268.60 + 0.9\left[0.01983V_\delta(0) + \ldots + 0.0377V_\delta(11)\right] = 3897.62 \\ 251.61 + 0.9\left[0.01983V_\delta(1) + \ldots + 0.0377V_\delta(12)\right] = 3866.31 \\ 268.62 + 0.9\left[0.01983V_\delta(2) + \ldots + 0.0377V_\delta(13)\right] = 3853.80 \\ 285.63 + 0.9\left[0.01983V_\delta(3) + \ldots + 0.0377V_\delta(14)\right] = 3839.51 \\ 302.64 + 0.9\left[0.01983V_\delta(4) + \ldots + 0.0377V_\delta(15)\right] = 3811.14 \\ 319.65 + 0.9\left[0.01983V_\delta(5) + \ldots + 0.0377V_\delta(16)\right] = 3781.12 \\ 336.66 + 0.9\left[0.01983V_\delta(6) + \ldots + 0.0377V_\delta(17)\right] = 3746.91 \\ 353.67 + 0.9\left[0.01983V_\delta(7) + \ldots + 0.0377V_\delta(18)\right] = 3718.04 \\ 370.68 + 0.9\left[0.01983V_\delta(8) + \ldots + 0.0377V_\delta(19)\right] = 3698.77 \\ 387.69 + 0.9\left[0.01983V_\delta(9) + \ldots + 0.0377V_\delta(20)\right] = \mathbf{3683.71} \end{cases}$$

For each state $i = 0, 1, 2, \ldots, 20$, refer to Table 4.5 for the comparison of $V_{(i)}$'s values and the values of $T_{(i)}$. From the table, we see that for each i, $V_{(i)} \leq T_{(i)}$. This observation implies that our initial stationary policy is optimal, and tends to minimized expected discounted production and holding costs. For instance, if at the start of production in period 1, we observe our inventory position to be zero ($i = 0$) units and decide to produce or process 28 batches of limestones, an expected discounted cost of *4198.12₺* may be incurred *(Currency: Turkish Lira)*. Again, assuming we observe inventory position to be 15units (batches), then we can minimize cost by producing a quantity between 22 and 28 batches. This production range minimizes, on the average, a discounted cost of *3646.43₺*.

4.5 Summary

In this chapter, an attempt has been made to solve an inventory control problem using Markov decision process. The aim was to find an optimal production schedule for the production of limestones, a case study in the region of Trakya. The problem definition was clearly stated, followed by a characterization of the problem as an MDP.

We assumed the system is completely observable, in respect of its state characteristics. Policy iteration methodology was used to arrive at an optimal policy. Thus, an initial stationary policy was defined and its value determination equations solved. We tested for optimality of the stationary policy by using Howard's policy iteration method. We closed the chapter by briefly explaining what the results of the computations mean.

CHAPTER FIVE
CONCLUSION

In any industrial production, the need for strategic or contingency planning continues to attract more attention, especially with the growing complexity of today's markets. In this thesis, the focal point of our study is based on production scheduling. The work focused on a case study of an inventory control problem of a firm that processes limestones on a large scale. The main task has been to propose a policy that minimizes the firm's production and holding costs over an infinite time horizon. After carefully studying the problem, we proposed a policy and subjected it to a rigorous computational test, by way of testing the *optimality* of our proposed policy. The underlying mathematical framework used is *Markov decision processes*. The approach used in this study and the conclusion reached are summarized below.

Prior to proposing a policy for the inventory control problem, it has been assumed that the system's characteristics (demands, maximum production capacity, maximum storage capacity etc.) are completely observable, and as a matter of fact, give accurate information about the state of the system. This has aided in the characterization of the state structure of the problem, by virtue of the sequence of production dynamics. Considering the production capacity, demand distribution and maximum allowable storage space, our proposed stationary policy has been that; for $i \in \{0, 1, 2, \ldots, 9\}$, $x_t = 28$. Thus, every time the firm's inventory position is in this range, a production decision of 28 *batches* minimizes on the average a discounted cost of *4075.32₺*. For $i = 10,\ x_t = 27$. In this case, the expected discounted cost is *4052.35₺*. The same interpretation is given to rest of our policy as shown in Table 4.5 (page 42). Returning back to the goal of this study, we may ask: Is this proposed policy *optimal*? The fact that a policy is stationary does not necessarily imply it is optimal. Testing our proposed stationary policy against the *Howard's policy iteration* method has showed that it is an optimal policy and does minimize long expected discounted production costs.

Alhough the stationary policy proposed in this thesis has been found to be an optimal one, it must be mentioned that it may not be the only optimal policy that minimizes the value function or utility function of the problem under consideration. Other policies may minimize discounted cost better than the one we have proposed. In short, the production problem we studied in this thesis is an infinite-horizon MDP and we do not claim that our policy would outperforms other optimal policies if actually there exist other optimal policies. In fact, the problem studied in this work is opened to further policy evaluations.

The q-learning concept introduced in chapter 3 would come into play if we were not privy to a comprehensive information of the transition probabilities. In that case, the search for an optimal solution could be approached by 'learning' the system based on the information we have on the states, actions and rewards. As one of the possible further research directions, we recommend the use of *q-learning* approach in situations where the problem before us is not completely observable, but by a learning scheme. Whiles a Markov decision process in this study has been applied to find an optimal inventory control policy, it is possible that in some situations, the approach rather turns out to be onerous. There may be instances where the value function is unbounded (for an infinite-horizon MDP). In this case, the search for an optimal policy may only result in a worthless exercise. If one considers the assumptions made in this study, relaxing or adding more constraints to the already known ones could actually make the problem more complex or rather simpler to model; as long as the system's characteristics are completely observable. In other words, we could learn more about the system by expanding the problem and taking it into different research directions. If a bounded value function could be obtained, we stand to draw more insights into the solution structure. Blackwell (1962), in particular, has shown that, for a bounded utility function, an *optimal policy always exists*. But in the case of an unbounded value function, the existence of an optimal policy is not always obvious.

REFERENCES

Anwar, M. F., & Nagi, R. (1997). Integrated lot-sizing and scheduling for just-in-time production of complex assemblies with finite set-ups. *International Journal of Production Research*, *35*(5), 1447–1470.

Bellman, R. et al. (1954). The theory of dynamic programming. *Bulletin of the American Mathematical Society*, *60*(6), 503–515.

Bellman, R. E., & Dreyfus, S. E. (2015). *Applied dynamic programming*, Princeton: Princeton University Press.

Bertolini, M., & Rizzi, A. (2002). A simulation approach to manage finished goods inventory replenishment economically in a mixed push/pull environment. *Logistics Information Management*, *15*(4), 281–293.

Bethke, B., How, J. P., & Ozdaglar, A. (2008). Approximate dynamic programming using support vector regression. *Proceeding Book of 47th IEEE Conference on Decision and Control*, 3811-3816. Retrieved November 23, from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4739322&isnumber=4738560.

Blackwell, D. (1962). Discrete dynamic programming. *The Annals of Mathematical Statistics*, *33*(2), 719–726.

Blackwell, D. (1965). Discounted dynamic programming. *The Annals of Mathematical Statistics*, *36*(1), 226–235.

Boynton, R. S. (1966). *Chemistry and technology of lime and limestone*. New York: John Wiley and Sons.

Caramanis, M. C., Anli, O. M., & Paschalidis, I. C. (2003). Supply chain (sc) production planning with dynamic lead time and quality of service constraints. *Proceeding Book of 42nd IEEE International Conference on Decision and Control,* 5478–5485. Retrieved December 23, from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1272509&isnumber=28459

Dubins, L. E., & Savage, L. J. (1965). *Inequalities for stochastic processes: How to gamble if you must*. USA: Dover Publications.

Dubois, J. (1995). Borehole radar experiment in limestone: *Analysis and data processing. First Break*, *13*(2), 57–66.

Gourdin, K. (2001). *Global logistics management: a competitive advantage for the 21st century.* USA: Wiley-Blackwell.

Howard, R. A. (1960). *Dynamic programming and Markov processes*. USA:Blackwell

Li, Y. (2017). *Deep reinforcement learning: An overview.* arXiv preprint arXiv:1701.07274.

Oates, J. (1998). *Lime and limestone*. New York: John Wiley & Sons.

Patterson, J., Slowinski, R., Talbot, F., & Weglarz, J. (1989). *Advances in project scheduling* (1st ed.). North Holland: Elsevier.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. (1st ed.). New York: John Wiley & Sons.

Ross, S. M. (2014). *Introduction to stochastic dynamic programming*. New York: Academic press.

Schneider, J. G., Boyan, J. A., & Moore, A. W. (1998). Value function based production scheduling. In *International Conference on Machine Learning* (522–530). Massachusetts: Morgan Kaufmann

Shapiro, J. F. (1993). Mathematical programming models and methods for production planning and scheduling. In *Handbooks in Operations Research and Management Science, 4*, (371–443). New York: John Wiley & Sons.

Smith, D. K. (1996). Dynamic programming and optimal control. *Journal of the Operational Research Society*, *47*(6), 833–834.

Sturmberg, J. P., & Martin, C. (2013). *Handbook of Systems and Complexity in Health*. USA: Springer Science & Business Media.

Sutton, R. S., Barto, A. G. et al. (1998). *Introduction to reinforcement learning*, *135*. Cambridge: MIT press.

White, D. J. (1993). A survey of applications of Markov decision processes. *Journal of the Operational Research Society, 44*(11), 1073–1096.

Wild, T. (2017). *Best practice in inventory management*. UK: Routledge.

Winston, W. L., & Goldberg, J. B. (2004). *Operations Research: Applications and Algorithms*, (3rd ed.). New York: John Wiley & Sons.

APPENDICES

A.1: Weekly random demands and transition probabilities

Table A.1 Demand in tonnes and batches (a)

$D_1 \times 10^3$	D_2
585	250
591	280
594	390
627	230
634	260
658	270
660	360
674	290
679	330
682	320
686	280
687	280
691	240
700	340
706	280
706	340
707	320
713	320
721	300
732	260
744	370
750	310
770	270
780	250
785	350
⋮	⋮

Table A.2 Demand in tonnes and batches (b)

$D_1 \times 10^3$	D_2
798	290
801	310
805	330
806	270
812	280
824	370
831	330
835	370
853	300
856	320
860	290
869	410
873	390
874	270
896	240
909	410
917	370
918	350
923	350
924	310
940	320
966	270
977	390
978	340
979	380
1024	390
1033	360

Table A.3 Transition probabilities(a)

i	x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	INIT.VAL.	4198,12	4174,77	4179,88	4165,02	3669,42	4069,09	4064,72	4086,06	4079,76	4066,84	4052,35	3714,47	3697,46	3680,45	3663,34	3646,43	3629,42	3612,41	3595,4	3578,39	3561,38
0	28	0,9014	0,0609	0,0377																		
1	28	0,814	0,0874	0,0609	0,0377																	
2	28	0,7016	0,1124	0,0874	0,0609	0,0377																
3	28	0,5723	0,1293	0,1124	0,0874	0,0609	0,0377															
4	28	0,4237	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377														
5	28	0,2914	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377													
6	28	0,1809	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377												
7	28	0,1068	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377											
8	28	0,0483	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377										
9	28	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377									
10	27	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377									
	28		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
11	26	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377									
	27		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	28			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
12	25	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377									
	26		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	27			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	28				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
13	24	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377									
	25		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	26			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	27				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377						
	28					0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377					
14	23	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377									
	24		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	25			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	26				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377						
	27					0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377					
	28						0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377				
15	22	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377									
	23		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	24			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	25				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377						
	26					0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377					
	27						0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377				
	28							0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377			
	28							0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377			

Table A.4 Transition probabilities(b)

i	x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	INIT.VAL.	4198,12	4174,77	4179,88	4165,02	3669,42	4069,09	4064,72	4086,06	4079,76	4066,84	4052,35	3714,47	3697,46	3680,45	3663,34	3646,43	3629,42	3612,41	3595,4	3578,39	3561,38
16	21	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377									
	22		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	23			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	24				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	25					0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377						
	26						0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377					
	27							0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377				
	28								0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377			
17	20	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377										
	21		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	22			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	23				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	24					0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377						
	25						0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377					
	26							0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377				
	27								0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377			
	28									0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377		
18	19	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377										
	20		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	21			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	22				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	23					0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377						
	24						0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377					
	25							0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377				
	26								0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377			
	27									0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377		
	28										0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377	
19	18	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377										
	19		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	20			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	21				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	22					0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377						
	23						0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377					
	24							0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377				
	25								0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377			
	26									0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377		
	27										0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377	
20	17	0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377										
	18		0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	19			0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377								
	20				0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377							
	21					0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377						
	22						0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377					
	23							0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377				
	24								0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377			
	25									0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377		
	26										0,0198	0,0285	0,0585	0,0741	0,1105	0,1323	0,1486	0,1293	0,1124	0,0874	0,0609	0,0377	