OPTIMIZATION PROCEDURES FOR MARKOVIAN AND SEMI-MARKOVIAN DECISION PROCESSES

by

ZHI-ZHONG OSCAR REN
University of Victoria, September 1994.

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Mathematics and Statistics

We accept this thesis as conforming
to the required standard

[redacted]
Dr. Roger R. Davidson, Supervisor (Department of Mathematics & Statistics)

[redacted]
Dr. Jane J. Ye, Department Member (Department of Mathematics & Statistics)

[redacted]
Dr. Arthur Watton, Outside Member (Department of Physics & Astronomy)

[redacted]
Dr. Kenneth G. Stewart, External Examiner (Department of Economics)

Supervisor: Dr. Roger R. Davidson (Department of Mathematics and Statistics)

Abstract

In this paper, we investigate both **Markovian decision processes (MDP)** and **semi-Markovian decision processes (Semi-MDP)**, for either discrete or continuous time, and with or without discounting. Attention is focused primarily on the determination of optimal strategy in **MDP** or **Semi-MDP** with finite states and finite action space.

The structures of system rewards in terms of yields and bonuses associated with state occupancies, transitions among the states in the process and the action taken in each state are presented and incorporated into the appropriate optimization criteria and algorithms. The existence of optimal stationary strategies for an infinite horizon are noted and used in the algorithms for different cases.

The different policy iteration methods involving the appropriate **policy improvement algorithms (PIA)** as well as the **value determination operations (VDO)** used to obtain the optimal stationary strategies and the total expected return values or average gain value per unit time over infinite time horizons are presented; Moreover, the **value iteration procedure (VIP)** used to obtain optimal time-dependent strategies and the corresponding optimized total expected return values for discrete time **MDP** or **Semi-MDP** over finite horizons are also presented. All the algorithms are fully discussed and a number of examples are presented through this paper.

In addition to the discussion of some of the underlying theory and properties of **MDP** and **Semi-MDP**, this paper also provides a set of programs written and tested in **Maple** for implementing the various optimization algorithm.

Examiners:

Dr. Roger R. Davidson, Supervisor (Department of Mathematics & Statistics)

Dr. Jane J. Ye, Department Member (Department of Mathematics & Statistics)

Dr. Arthur Watton, Outside Member (Department of Physics & Astronomy)

Dr. Kenneth G. Stewart, External Examiner (Department of Economics)

Contents

List of Tables

List of Symbols

P_{ij}^n	n-step transition probability from state i to state j
N	Number of states in system
S	State space: $\{1, 2, \ldots, N\}$
K	Policy space: $K_1 \times K_2 \times \cdots \times K_N$
K_i	Action set for state i
k	Single action taken in a state
p_{ij}^k	Markov transition probability associated with action k
r_{ij}^k	Transition reward associated with action k
$R(d)$	System reward vector associated with a decision vector d
r_i^k	i^{th} component of $R(d)$
δ	Strategy or policy over an infinite horizon: $(d_1, d_2, \ldots)$
δ_n	Strategy or policy over a finite horizon n: $(d_1, \ldots, d_n)$
d_n	Decision vector when the time remaining is n
$P_n(\delta_n)$	Non-stationary n-step transition matrix: $P(d_1) \cdots P(d_n)$
d^∞	Stationary strategy: $(d, d, \ldots)$
β	Discount factor for discrete time
$V_\beta^n(\delta_n)$	Total discounted expected return vector for a finite horizon
$v_i(n, \beta, \delta_n)$	i^{th} component of $V_\beta^n(\delta_n)$
δ_n^*	β-optimal strategy for a finite horizon
V_β^n	Optimized total discounted expected return vector
$v_i(n, \beta)$	i^{th} component of V_β^n
VIP	Value Iteration Procedure for a finite horizon
$V_\beta(\delta)$	Total discounted expected return vector for an infinite horizon
$v_i(\beta, \delta)$	i^{th} component of $V_\beta(\delta)$
$T\delta$	Strategy delayed one step for each time n: $(d_2, d_3, \ldots)$
$\mathbf{1}$	Vector whose components are all 1's
$\mathbf{I}$	Identity matrix
$L(d)$	Montone operator: $V \longmapsto R(d) + \beta P(d)V$
δ^*	β-optimal strategy for an infinite horizon
$D(i, d)$	Alternative decision set
$V_\beta(d)$	Total discounted expected return vector under d^∞
$v_i(\beta, d)$	i^{th} component of $V_\beta(d)$
VDO	Value Determination Operation for an infinite horizon
PIA	Policy Improvement Algorithm for an infinite horizon
$V^n(\delta_n)$	Total non-discounted expected return vector for a finite horizon
$v_i(n, \delta_n)$	i^{th} component of $V^n(\delta_n)$
V^n	Optimized total non-discounted expected return vector
$v_i(n)$	i^{th} component of V^n

$G(\delta)$	Average return vector per unit time for an infinite horizon
$V(d)$	Total non-discounted expected return vector under d^∞
$v_i(d)$	i^{th} component of $V(d)$
π	Steady state distribution across the state space S
$g(d)$	Average return value per unit time under d^∞ for an infinite horizon
$V(d)$	Relative value vector under d^∞ for an infinite horizon
$v_i(d)$	i^{th} component of $V(d)$
S_T	Transient state space: $\{2, 3, \ldots, N\}$
$h_{ij}^k(n)$	Holding time probability associated with action k
$w_{ii}^k(n)$	Waiting time probability associated with action k
$w_{ii}^*(n)$	Cumulative waiting time distribution for discrete time
$\bar{h}_{ij}^k$	Mean holding time
$\bar{w}_{ii}^k$	Mean waiting time
$\ddot{h}_{ij}^k(n)$	Complement of the cumulative holding time distribution
$\ddot{w}_{ii}^k(n)$	Complement of the cumulative waiting time distribution
h_{ij}^{gk}	Geometric transform of $h_{ij}^k(n)$ associated with action k
ϕ_{ij}^k	Interval transition probability associated with action k
$\odot$	Direct multiplication of two matrices
$v_i(0)$	Terminal reward for state i
$y_{ij}^k(n)$	Yield associated with the action k
$b_{ij}^k(n)$	Bonus associated with the action k
$\ddot{y}_i^k(n)$	Contribution to the total expected return from $y_{ij}^k(n)$
$\eta(d)$	Mean waiting time vector with components $\bar{w}_{ii}^k$
α	Discount rate for continuous time
$h_{ij}^k(t)$	Holding time density associated with action k
$w_{ii}^k(t)$	Waiting time density associated with action k
$w_{ii}^*(t)$	Cumulative waiting time distribution for continuous time
$\ddot{h}_{ij}^k(t)$	Complement of the cumulative holding time distribution
$\ddot{w}_{ii}^k(t)$	Complement of the cumulative waiting time distribution
h_{ij}^{ek}	Laplace transform of $h_{ij}^k(t)$ associated with action k
$y_{ij}^k(t)$	Yield rate associated with the action k
$b_{ij}^k(t)$	Bonus associated with the action k
$\ddot{y}_i^k(t)$	Contribution to the total expected return from $y_{ij}^k(t)$
$V_\alpha(t, d)$	Total discounted expected return vector under d^∞
$v_i(t, \alpha, d)$	i^{th} component of $V_\alpha(t, d)$
$V(t, d)$	Total non-discounted expected return vector under d^∞
$v_i(t, d)$	i^{th} component of $V(t, d)$
λ_i	Rate parameter of exponential distribution
a_{ij}^k	Transition rate associated with action k for a continuous time MDP
q_{ij}^k	Earning rate associated with action k for a continuous time MDP

Acknowledgement

I wish to acknowledge the most generous guidance, valuable assistance, excellent editing skills and great interest of Dr. Roger R. Davidson, my master programme supervisor in the Department of Mathematics and Statistics. I would like to express my most sincere gratitude and special thanks to Dr. Davidson for his constant encouragement and his incredible patience during my two years of graduate studies. I could not possibly have completed this thesis without him.

I would also like to thank Dr. J. Ye, Dr. A. Watton and Dr. K. Stewart for their time to examine my thesis and provide me with their valuable comments.

Finally, I want to sincerely thank my dear parents, my brother Steve for their constant love, moral as well as emotional support since the first day I came to Canada for my graduate studies. I would like to dedicate this thesis to them for my most sincere appreciation.

Oscar Z. Ren

Chapter 1

Discrete Time Markovian Decision Process

Markovian decision processes (MDP) are stochastic sequential processes that describe the evolution of dynamic systems controlled by sequences of decisions or actions. Periodic observations are made on the system and the process is then classified into one of a possible finite number of states in the system. The decision made at each stage influences the probabilities of transition into other states. The objective is to select a decision from among the available alternatives in order to optimize the total or average return or the total or average operating cost generated by the system as the process evolves.

1.1 Classification of States in a Markov Chain

In order to describe a Markovian decision process, it is necessary to present the classification system for the states of a Markov chain and to summarize some of the basic properties of Markov chains (Ross [1]).

Definition 1.1 : Classification of states in a Markov chain

a. *State j is said to be* **accessible** *from state i if the n-step transition probability $P_{ij}^n > 0$ for some positive n.*

This definition implies that state j is accessible from state i if and only if, starting in i, it is possible for the process will ever enter state j in a finite number of transitions.

b. *States i and j are said to* **communicate** *if they are accessible from each other i.e. $i \leftrightarrow j$.*

By using the **Chapman-Kolmogorov** equations, it can be shown that communication among states is an equivalence relation.

c. *A Markov chain is said to be* **irreducible** *if there is only one communicating class i.e. each pair of states in the Markov chain communicate.*

d. *State i is said to be* **recurrent** *if the process, once in state i, will eventually reenter state i with probability 1; otherwise, state i is said to be* **transient**.

From the definition, it follows that the process, once in a recurrent state, will with probability 1 return infinitely often to that state, whereas the process, once in a transient state, will with positive probability never return to that state. Furthermore

$$\sum_{n=1}^{\infty} P_{ii}^n = \infty \Rightarrow \textit{State } i \textit{ is recurrent}$$

$$\sum_{n=1}^{\infty} P_{ii}^n < \infty \Rightarrow \textit{State } i \textit{ is transient}$$

where $\sum_{n=1}^{\infty} P_{ii}^n$ is the expected number of time periods that the process returns to state i.

e. *State i is said to have* **period** *d if $P_{ii}^n = 0$ whenever n is not divisible by d and d is the largest integer with this property. A state with period 1 is said to be* **aperiodic**.

It can be shown that the states belonging to the same communicating class have the same period i.e. periodicity is a class property.

f. *A recurrent state i is said to be* **positive recurrent** *if starting in state i, the expected time until the process returns to state i is finite. A recurrent state that is not positive recurrent is said to be* **null recurrent.**

From the definition, it immediately follows that for a Markov chain with a finite state space, all recurrent states are positive recurrent. It can also be shown that the positive recurrence is a class property.

g. *State i is said to be* **ergodic** *if it is both positive recurrent and aperiodic.*

Ergodicity is also a class property.

h. *A Markov chain is said to be* **completely-ergodic** *if it is irreducible and ergodic.*

1.2 Discrete Time Markovian Decision Process with Discounting

First, let us give a definition for a Markovian decision process.

Definition 1.2 : Markovian Decision Process

A Markovian decision process is a stochastic sequential process that describes the evolution of dynamic systems controlled by sequences of decisions or actions. Such dynamic systems are observed periodically and influenced at the time of observation by the choice of one of a possible number of actions. Furthermore, the Markovian decision process is based on discrete-time Markov chains governing the transitions among the states for the process.

Let us assume that at integer times $t = 0, 1, 2, \ldots$, the system is observed and then classified into one of the states in a finite state space $S = \{1, 2, \ldots, N\}$. For each $i \in S$, there is a finite set K_i of finite actions (or alternatives) available. The

policy space K is then defined as $K = K_1 \times K_2 \times \cdots \times K_N$, the *Cartesian* product of the N action sets.

When the system is in state $i \in S$ and action $k \in K_i$ is taken, the system makes a transition from state i to state j with predetermined transition probability p_{ij}^k which depends on the action k taken when the system is in state i, and a transition reward r_{ij}^k for the transition is produced. The case $j = i$ corresponds to the situation where the process remains in state i. In this way, the Markov process generates a sequence of rewards determined by the actions taken as it makes transitions from state to state. The rewards are random variables with probability distributions governed by the transition probabilities of the Markov process.

The system reward r_i^k associated with taking action k while in state i is the weighted average of the transition rewards r_{ij}^k while in state i, that is

$$r_i^k = \sum_{j=1}^{N} p_{ij}^k r_{ij}^k$$

The primary goal for a MDP with discounting is to find **strategies** that maximize the discounted total expected return over a finite-time or an infinite-time horizon. This is accomplished by implementing a value iteration procedure (**VIP**) for finite horizon and the policy iteration method consisting of the value determination operation (**VDO**) and the policy improvement algorithm (**PIA**) for infinite horizon.

Definition 1.3 : Strategy

A strategy is a sequence of decision vectors: $\delta = (d_1, d_2, \ldots, d_n, \ldots)$, *where* $d_n = (d_n(1), d_n(2), \ldots, d_n(N)) \in D$ *and* $d_n(i)$ *is the decision (action) taken at time* n *when the system is in state* i. D *is a set of functions from the state space* S *to the policy space* K. *The strategy* δ *restricted to time horizon* n *is denoted* $\delta_n = (d_1, \ldots, d_n)$ *in which case* d_m *is the decision vector when the time remaining is* $l = n + 1 - m$ *for* $m = 1, \ldots, n$.

The multi-step transition probability matrix $P_n(\delta_n)$ associated with the strategy δ_n for the process up to time n can be written

$$P_n(\delta_n) = P(d_1) \cdots P(d_n) \quad for\ n = 1, 2, \ldots$$

where $P(d_n)$ is the $N \times N$ transition probability matrix whose ijth element p_{ij}^k depends on the current state i, the next state j to be visited and the action $k \in K_i$ taken while in state i. For $n = 0, P_0 = \mathbf{I}$, the identity matrix of size $N \times N$.

Definition 1.4 : Non-Stationary Markovian Decision Processes

The transitions among states in a Markovian decision process are governed by a non-stationary Markov chain described by $P_n(\delta_n) = P(d_1) \cdots P(d_n)$ *for* $n = 1, 2, \ldots$.

Definition 1.5 : Stationary Strategy

A stationary strategy is a sequence $d^\infty = (d, d, \ldots, d, \ldots)$ *with a repeating decision vector* d *that is independent of time.*

Similar to the definition for a non-stationary Markovian Decision Process, we have

Definition 1.6 : Stationary Markovian Decision Processes

The transitions among states in a Markovian decision process are governed by a stationary Markov chain described by $P_n(\delta_n) = P^n(d)$ *for* $n = 1, 2, \ldots$

1.2.1 Value Iteration Procedure for a Finite Horizon

First consider a **MDP** in which a discount factor β $(0 \leq \beta < 1)$ is present. The discount factor, which can be interpreted as the reciprocal of 1 plus the interest rate, ensures the convergence of the total expected return when the process is allowed to continue for an infinite period of time i.e. for an infinite time horizon. Then the present value of a unit return generated by the system at time n is β^n.

Definition 1.7 : Total discounted expected return for a finite horizon

$$V_\beta^n(\delta_n) = \sum_{m=0}^{n-1} \beta^m P_m(\delta_m) R(d_{m+1})$$

The vector $V_\beta^n(\delta_n) = (v_1(n,\beta,\delta_n),\ldots,v_N(n,\beta,\delta_n))^T$ *represents the total discounted expected return for finite horizon* n *when using strategy* δ_n, *where* $R(d) = (r_1^{k_1},\ldots,r_N^{k_N})^T$ *is the vector of system rewards for decision vector* d *with* $d(i) = k_i,\ i = 1,2,\ldots,N$ (Mine & Osaki [2]).

Using Definition 1.7, it is easy to establish the following recursive relationship for $v_i(n,\beta,\delta_n)$ for time horizon n (Howard [3])

$$\begin{aligned} v_i(n,\beta,\delta_n) &= \sum_{j=1}^{N} p_{ij}^k [r_{ij}^k + \beta v_j(n-1,\beta,\delta_{n-1})] \\ &= r_i^k + \beta \sum_{j=1}^{N} p_{ij}^k v_j(n-1,\beta,\delta_{n-1}) \end{aligned}$$

where $\delta_n = (d,\delta_{n-1})$ with $d(i) = k$ and $v_i(0) = 0$ for $i = 1,2,\ldots,N$ and $n = 1,2,\ldots$

The above equations state that if the system makes a transition from state i to state j after having chosen the action k in state i, it will earn the reward r_{ij}^k plus the expected return when it enters state j with time remaining $n-1$. The quantity r_i^k is the system reward associated with taking action k while in state i.

Definition 1.8 : β-optimal strategy for a finite horizon

A strategy $\delta_n^* = (d_1,\ldots,d_n)$ *is said to be* β*-optimal for time horizon* n *if*

$$V_\beta^n(\delta_n^*) \geq V_\beta^n(\delta_n) \text{ for all strategies } \delta_n$$

where β $(0 \leq \beta < 1)$ *is the fixed discount factor.*

We now introduce the value iteration procedure used to obtain β-optimal policy δ_n^* for time horizon n (Howard [3]).

VIP: Value Iteration Procedure

The maximum total expected return vector $V_\beta^n = \max V^n(\delta_n)$ and the associated β-optimal policy δ_n^* for time horizon n are obtained recursively using V_β^{n-1} and δ_{n-1}^* for horizon n-1 as follows: for each horizon $n = 1, 2, \ldots$, obtain

$$v_i(n, \beta) = \max_k \left\{ r_i^k + \beta \sum_{j=1}^{N} p_{ij}^k v_j(n-1, \beta) \right\}$$

and the action $\tilde{k} \in K_i$ which produces the maximum for $i = 1, 2, \ldots, N$. Then the optimal policy for time horizon n is $\delta_n^* = (d, \delta_{n-1}^*)$ i.e. the optimal policy δ_{n-1}^* preceeded by the decision vector $d = (\tilde{k}_1, \tilde{k}_2, \ldots, \tilde{k}_N)$. In this way, we recursively generate V_β^n and the optimal policy $\delta_n^* = (d_1, \ldots, d_n)$ where d_m is the optimal decision vector for the initial decision when the time remaining is $l = n + 1 - m$.

Remark 1.9 : Implementation of the value iteration procedure

The above value iteration procedure to find a time-dependent optimal policy and the corresponding total discounted expected return over a finite horizon for a discrete time MDP with discounting can be implemented by using the program **dt_MDP** *which uses the procedure* **VIP_mdp** *with appropriate inputs* (Appendices A.1 and B.1).

1.2.2 Stationary β-Optimal Strategies for a Infinite Horizon

Definition 1.10 : Total discounted expected return for an infinite horizon

$$V_\beta(\delta) = \sum_{n=0}^{\infty} \beta^n P_n(\delta_n) R(d_{n+1})$$

The vector $V_\beta(\delta) = (v_1(\beta, \delta), \ldots, v_N(\beta, \delta))^T$ *represents the total discounted expected return for an infinite horizon when using strategy* δ, *where* $R(d) = (r_1^{k_1}, \ldots, r_N^{k_N})^T$ *is*

the vector of system rewards for decision vector d with $d(i) = k_i,\ i = 1, 2, \ldots, N$.

(Mine & Osaki [2])

Lemma 1.11

$$V_\beta(\delta) = R(d_1) + \beta P(d_1) V_\beta(T\delta)$$

where $T\delta = (d_2, d_3, \ldots)$ is the strategy delayed one step for each time n.

Proof:

$$\begin{aligned} V_\beta(\delta) &= R(d_1) + \sum_{n=1}^{\infty} \beta^n P_n(\delta_n) R(d_{n+1}) \\ &= R(d_1) + \beta P(d_1) \sum_{n=0}^{\infty} \beta^n P_n(T\delta) R(d_{n+2}) \\ &= R(d_1) + \beta P(d_1) V_\beta(T\delta) \end{aligned}$$

□

Lemma 1.12 : Finiteness of system return values for infinite horizon

$$\frac{r^L}{(1-\beta)}\mathbf{1} \leq V_\beta(\delta) \leq \frac{r^U}{(1-\beta)}\mathbf{1}$$

where

$$r^U = \max_{i,k} r_i^k \text{ and } r^L = \min_{i,k} r_i^k$$

and $\mathbf{1}$ is the N dimensional vector with all elements equal to one.

Proof:

$$V_\beta(\delta) = \sum_{n=0}^{\infty} \beta^n P_n(\delta_n) R(d_{n+1}) \leq \sum_{n=0}^{\infty} \beta^n P_n(\delta_n) r^U \mathbf{1} = r^U \sum_{n=0}^{\infty} \beta^n \mathbf{1} = \frac{r^U}{(1-\beta)}\mathbf{1}$$

since $P_n(\delta_n)$ has row sums equal to one i.e. $P_n(\delta_n)\mathbf{1} = \mathbf{1}$. Similarily it follows that

$$V_\beta(\delta) \geq \frac{r^L}{(1-\beta)}\mathbf{1}$$

□

Definition 1.13 : Monotone mapping

Let L(d) to be an operator such that $V \overset{L(d)}{\longmapsto} R(d) + \beta P(d)V$ *for any N dimensional vector V. L(d) is a* **monotone** *mapping since*

$$L(d)(V_1) - L(d)(V_2) = \beta P(d)(V_1 - V_2) \geq 0 \quad if \ V_1 \geq V_2$$

Remark 1.14

It follows immediately that

$$L(d)(V_\beta(\delta)) = R(d) + \beta P(d)V_\beta(\delta) = V_\beta(d, d_1, \ldots, d_n, \ldots) = V_\beta(d, \delta)$$

i.e. L(d) applied to the total expected return value for strategy δ *yields the total expected return value for* (d, δ)*, the strategy in which* δ *is preceeded by decision vector d* (Mine & Osaki [2]).

Definition 1.15 : β-optimal strategy for an infinite horizon

A strategy $\delta^* = (d_1, d_2, \ldots, d_n, \ldots)$ *is said to be* β *-optimal if* $V_\beta(\delta^*) \geq V_\beta(\delta)$ *for all* δ*, where* β $(0 \leq \beta < 1)$ *is the fixed discount factor.*

Theorem 1.16 : β-Optimal strategy (Property 1)

If $V_\beta(\delta^*) \geq V_\beta(d, \delta^*)$ *for all* $d \in D$*, then* δ^* *is a* β*-optimal strategy.*

Proof: For each $d \in D$,

$$V_\beta(d, \delta^*) = R(d) + \beta P(d)V_\beta(\delta^*) = L(d)(V_\beta(\delta^*))$$

The last equality holds by Remark 1.14 and hence by the assumption, it follows that $L(d)(V_\beta(\delta^*)) \leq V_\beta(\delta^*)$ holds for any decision vector $d \in D$. Since $L(d)$ is a monotone

operator, applying the above inequality to $L(d_1)L(d_2)\cdots L(d_{n-1})$ recursively yields the following

$$\begin{aligned} & V_\beta(d_1, d_2, \cdots, d_n, \delta^*) \\ &= L(d_1)L(d_2)\cdots L(d_{n-1})(L(d_n)(V_\beta(\delta^*)) \\ &\leq L(d_1)L(d_2)\cdots L(d_{n-1})(V_\beta(\delta^*)) \\ &\leq \vdots \\ &\leq L(d_1)(V_\beta(\delta^*)) \\ &\leq V_\beta(\delta^*). \end{aligned}$$

Then for any strategy $\delta = (d_1, d_2, \ldots)$

$$V_\beta(\delta) = \lim_{n\to\infty} V_\beta(d_1, d_2, \cdots, d_n, \delta^*) \leq V_\beta(\delta^*).$$

Thus δ^* is a β-optimal strategy.

□

Theorem 1.17

If $V_\beta(d, \delta) > V_\beta(\delta)$ for a given $d \in D$, then $V_\beta(d^\infty) > V_\beta(\delta)$.

Proof: Repeated application of the monotone linear operator L gives

$$L^n(d)(V_\beta(\delta)) = L^{n-1}(d)(V_\beta(d, \delta)) = L^{n-2}(d)(V_\beta(d, d, \delta)) = \cdots = V_\beta(\overbrace{d, d, \ldots, d}^{n}, \delta)$$

Thus by the assumed inequality $V_\beta(d, \delta) > V_\beta(\delta)$, one obtains

$$L^n(d)(V_\beta(\delta)) \geq L^{n-1}(d)(V_\beta(\delta)) \geq \cdots \geq L^1(d)(V_\beta(\delta)) = V_\beta(d, \delta) > V_\beta(\delta)$$

Then as $n \to \infty$, one obtains that $V_\beta(d^\infty) > V_\beta(\delta)$ for the given $d \in D$.

□

Corollary 1.18 : Stationary β-optimal strategy

If for a given $d \in D$, $V_\beta(d,\delta) > V_\beta(\delta)$ for all strategies δ, then d^∞ is a stationary β-optimal strategy (Mine & Osaki [2]).

Definition 1.19 : Alternative decision set

For each $i \in S$, let $D(i,d)$ be the set of all $k \in K_i$ for which

$$r_i^k + \beta \sum_{j \in S} p_{ij}^k v_j(d) > v_i(d)$$

where $v_i(d)$ is the i^{th} element of $V_\beta(d^\infty)$.

Theorem 1.20 : β-Optimal strategy (Property 2)

If $D(i,d) = \emptyset$ for all states $i \in S$, then d^∞ is β-optimal. If there is a $\tilde{d} \in D$ such that $\tilde{d}(i) \in D(i,d)$ for some $i \in S$ and $\tilde{d}(i) = d(i)$ whenever $\tilde{d}(i) \notin D(i,d)$, then $V_\beta(\tilde{d}^\infty) > V_\beta(d^\infty)$

Proof: For each $\tilde{d} \in D$, it follows that

$$V_\beta(\tilde{d}, d^\infty) = R(\tilde{d}) + \beta P(\tilde{d}) V_\beta(d^\infty).$$

(1) If $D(i,d) = \emptyset$ for all states $i \in S$, then for any $\tilde{d} \neq d$

$$r_i^k + \beta \sum_{j \in S} p_{ij}^k v_j(d) \leq v_i(d) \quad where\ k = \tilde{d}(i)$$

It then follows that$V_\beta(\tilde{d}, d^\infty) \leq V_\beta(d^\infty)$ and hence d^∞ is β-optimal by Theorem 1.16.

(2) If $\tilde{d}(i) \in D(i,d)$ for some $i \in S$ and $\tilde{d}(i) = d(i)$ whenever $\tilde{d}(i) \notin D(i,d)$, then we have the following results

$$r_i^k + \beta \sum_{j \in S} p_{ij}^k v_j(d) > v_i(d) \ for\ each\ state\ i\ for\ which\ \tilde{d}(i) \in D(i,d)\ \&\ k = \tilde{d}(i);$$

$$r_i^k + \beta \sum_{j \in S} p_{ij}^k v_j(d) = v_i(d) \ for\ each\ state\ i\ for\ which\ \tilde{d}(i) \notin D(i,d)\ \&\ k = d(i).$$

It then follows that $V_\beta(\tilde{d}, d^\infty) > V_\beta(d^\infty)$ and hence $V_\beta(\tilde{d}^\infty) > V_\beta(d^\infty)$

□

Theorem 1.21 : Existence of stationary β-optimal strategy

There exists a stationary β-optimal strategy for a Markovian decision process with a finite state space S and a finite policy space K.

Proof: Fix an arbitrary stationary strategy d^∞ for $d \in D$.

(1) If $D(i,d) = \emptyset$ for all states $i \in S$, then d^∞ is β-optimal according to the first part of Theorem 1.20.

(2) If $D(i,d) \neq \emptyset$ for some $i \in S$, then there is at least one $\tilde{d} \in D$ such that $\tilde{d}(i) \in D(i,d)$ for some $i \in S$ and $\tilde{d}(i) = d(i)$ whenever $\tilde{d}(i) \notin D(i,d)$. Then $V_\beta(\tilde{d}^\infty) > V_\beta(d^\infty)$ by the second part of Theorem 1.20. Hence, $\tilde{d}^\infty$ is an *improved* stationary strategy relative to d^∞.

Since the set D is finite i.e. there are only finitely many stationary strategies available, it follows that a stationary β-optimal strategy exists.

□

From this point on, $V_\beta(d)$ will be used in place of $V_\beta(d^\infty)$ for a stationary strategy $d^\infty = (d, d, \ldots)$ in the discounting case as long as no confusion arises.

1.2.3 Policy Iteration Method for an Infinite Horizon

For an infinite time horizon, we have the following policy iteration method consisting of the Value Determination Operation (**VDO**) and the Policy Improvement Algorithm (**PIA**) for a discrete-time MDP with discounting.

VDO: Value Determination Operation

Obtain the total discounted expected return vector $V_\beta(d)$ for stationary strategy d^∞ where $d \in D$ by solving the linear system

$$v_i(\beta, d) = r_i^k + \beta \sum_{j=1}^{N} p_{ij}^k v_j(\beta, d)$$

with $d(i) = k$ for $i = 1, 2, \ldots, N$.

PIA: Policy Improvement Algorithm

For each state $i \in S$ use $V_\beta(d)$ obtained from the **VDO** to find an action $\tilde{k} \in K_i$ which maximizes

$$r_i^k + \beta \sum_{j=1}^{N} p_{ij}^k v_j(\beta, d)$$

and let $\tilde{d}(i) = \tilde{k}$. If the maximum value exceeds $v_i(\beta, d)$ for any state i, then $\tilde{d}^\infty$ is an improved stationary strategy relative to d^∞.

The policy iteration method alternates the **VDO** and **PIA** steps until no improved stationary strategy is produced.

Remark 1.22 : Selection of initial stationary strategy

For improved efficiency in terms of possibly reducing the number of iterations required to obtain a stationary β-optimal strategy, the initial stationary strategy d_0^∞ should be chosen to maximize the system reward r_i^k for each $i \in S$.

Remark 1.23 : Implementation of the policy improvement method

The above policy improvement method to find a stationary β-optimal strategy and the corresponding total discounted expected return over an infinite horizon for a discrete time MDP with discounting can be implemented through the program **dt_MDP** *which uses the procedures* **PIA_mdp** *and* **VDO_mdp** *with appropriate inputs* (Appendices A.1, B.2 and B.3).

1.2.4 Example: Machine Maintenance Problem

We now present an example illustrating how the policy improvement method involving the **VDO** and the **PIA** yields a stationary β-optimal strategy and corresponding optimized total discounted expected return for a fixed discount factor β (Mine & Osaki [2]).

A machine which operates continuously is checked at the end of a fixed period of time, say at the end of each hour or each day. At each time there are two possible states: the machine is in state one if it is operable or in state two if it has failed. When the machine is in working condition, we can either let it keep working without intervention or carry out preventive maintenance in the hope it will operate longer. If the machine fails, we can choose to perform either normal repair or extended repair. Of course, preventive maintenance costs more than no intervention and extended repair costs more than normal repair.

The actions for the system in each of the two states can be defined as follows:
For state i=1: action k=1: no intervention; action k=2: preventive maintenance.
For state i=2: action k=1: normal repair; action k=2: extended repair.

Therefore, the state space and policy space are

$$S = \{1, 2\} \qquad K = \{(1,1), (1,2), (2,1), (2,2)\}$$

The transition probabilities and system rewards, both of which are associated with the actions taken, are given in the following table:

Table 1.1: Data of Machine Maintenance Problem

State i	Action k	Probability p_{i1}^k	Probability p_{i2}^k	System Reward r_i^k
1	1	0.7	0.3	3
	2	0.8	0.2	2
2	1	0.6	0.4	-1
	2	0.9	0.1	-2

Step 1 (**VDO**):

Take initial stationary strategy d_0^∞ where $d_0 = (1, 1)$ according to Remark 1.22 and solve the following linear system for $V_\beta(d_0)$

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \end{pmatrix} + 0.9 \begin{pmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \Longrightarrow V_\beta(d_0) = \begin{pmatrix} 18.13 \\ 13.74 \end{pmatrix}$$

Step 2 (**PIA**):

For each i we will identify the action k which maximizes $r_i^k + 0.9 \sum_{j=1}^2 p_{ij}^k v_j(d_0)$.

(1) For $i = 1$:

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} + 0.9 \begin{pmatrix} 0.7 & 0.3 \\ 0.8 & 0.2 \end{pmatrix} \begin{pmatrix} 18.13 \\ 13.74 \end{pmatrix} = \begin{pmatrix} 18.13 \\ 17.53 \end{pmatrix}$$

Hence $\tilde{k} = 1$ results in maximum for $i = 1$;

(2) For $i = 2$:

$$\begin{pmatrix} -1 \\ -2 \end{pmatrix} + 0.9 \begin{pmatrix} 0.6 & 0.4 \\ 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} 18.13 \\ 13.74 \end{pmatrix} = \begin{pmatrix} 13.74 \\ 13.92 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 2$. Therefore d_1^∞ where $d_1 = (1, 2)$ is an improved stationary strategy.

Step 3 (**VDO**):

Using the improved stationary strategy d_1^∞, solve the following linear system for $V_\beta(d_1)$

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 3 \\ -2 \end{pmatrix} + 0.9 \begin{pmatrix} 0.7 & 0.3 \\ 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \Longrightarrow V_\beta(d_1) = \begin{pmatrix} 18.56 \\ 14.32 \end{pmatrix}$$

Step 4 (**PIA**):

For each i we will identify the action k which maximizes $r_i^k + 0.9 \sum_{j=1}^{2} p_{ij}^k v_j(d_1)$.

(1) For $i = 1$:

$$\begin{pmatrix} 3 \\ 2 \end{pmatrix} + 0.9 \begin{pmatrix} 0.7 & 0.3 \\ 0.8 & 0.2 \end{pmatrix} \begin{pmatrix} 18.56 \\ 14.32 \end{pmatrix} = \begin{pmatrix} 18.56 \\ 17.94 \end{pmatrix}$$

Hence $\tilde{k} = 1$ results in maximum for $i = 1$;

(2) For $i = 2$:

$$\begin{pmatrix} -1 \\ -2 \end{pmatrix} + 0.9 \begin{pmatrix} 0.6 & 0.4 \\ 0.9 & 0.1 \end{pmatrix} \begin{pmatrix} 18.56 \\ 14.32 \end{pmatrix} = \begin{pmatrix} 14.18 \\ 14.32 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 2$. Therefore, no improved stationary strategy can be found and hence the stationary β-optimal strategy is d_1^∞ where $d_1 = (1, 2)$ and the optimized discounted total expected return values (if we allow the machine to operate indefinitely) are v_1=\$18.56 if the machine is in the operating state and v_2=\$14.32 if it is in repair state.

1.3 Discrete Time Markovian Decision Process without Discounting

1.3.1 Value Iteration Procedure for a Finite Horizon

Definition 1.24 : Total expected return in non-discounting case

$$V^n(\delta_n) = \sum_{m=0}^{n-1} P_m(\delta_m) R(d_{m+1})$$

The vector $V^n(\delta_n) = (v_1(n,\delta_n),\ldots,v_N(n,\delta_n))^T$ *represents the total discounted expected return for finite horizon* n *when using strategy* δ_n, *where* $R(d) = (r_1^{k_1},\ldots,r_N^{k_N})^T$ *is the vector of system rewards for decision vector* d *with* $d(i) = k_i,\ i = 1,2,\ldots,N$ (Mine & Osaki [2]).

Analogous to the discounting case discussed earlier, we have the following recursive relationship for $v_i(n,\delta_n)$ for time horizon n (Howard [3]):

$$v_i(n,\delta_n) = r_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(n-1,\delta_{n-1})$$

where $\delta_n = (d,\delta_{n-1})$ with $d(i) = k$ and $v_i(0) = 0$ for $i = 1,2,\ldots,N$ and $n = 1,2,\ldots$

We now introduce the value iteration procedure used to obtain an optimal policy δ_n^* for time horizon n (Howard [3]).

VIP: Value Iteration Procedure

The maximum total expected return vector $V^n = \max V^n(\delta_n)$ and the associated optimal policy δ_n^* for time horizon n are obtained recursively using V^{n-1} and δ_{n-1}^* for horizon $n-1$ as follows: for each horizon $n = 1,2,\ldots$, obtain

$$v_i(n) = \max_k \left\{ r_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(n-1) \right\}$$

and the action $\tilde{k} \in K_i$ which produces the maximum for $i = 1,2,\ldots,N$. In this way, we recursively generate V^n and the optimal policy $\delta_n^* = (d_1,\ldots,d_n)$ where d_m is the optimal decision vector for the initial decision when the time remaining is $l = n+1-m$.

Remark 1.25 : Implementation of the value iteration procedure

The above value iteration procedure to find a time-dependent optimal policy and the corresponding total expected return over a finite horizon for a discrete time MDP without discounting can be implemented by using the program **dt_MDP** *which uses the procedure* **VIP_mdp** *with appropriate inputs* (Appendices A.1 and B.1).

1.3.2 Average Return and Stationary Optimal Strategy

When there is no discounting in the discrete time Markovian Decision Process, then the total expected return will generally grow without bound i.e. $V(\delta)$ will usually be divergent over an infinite time period. We thus introduce an alternative optimization criterion based on the long run average return per unit time.

Definition 1.26 : Average return per unit time

For strategy $\delta = (d_1, d_2, \ldots)$, the average return vector $G(\delta)$ per unit time over an infinite time horizon is defined as

$$G(\delta) = \liminf_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} P_m(\delta_m) R(d_{m+1})$$

where $V^n(\delta) = \sum_{m=0}^{n-1} P_m(\delta_m) R(d_{m+1})$ is the vector of total expected returns up to time n for the chosen strategy δ (Mine & Osaki [2]).

Since $V^n(\delta)$ will usually be divergent as $n \to \infty$ in the non-discounting case, our objective is to find a strategy δ that maximizes this long run average return value per unit time.

Definition 1.27 : Optimal strategy for non-discounted MDP

A strategy δ^* is said to be an optimal strategy for the non-discounted MDP if

$$G(\delta^*) \geq G(\delta) \quad \textit{for all } \delta$$

Lemma 1.28

For any $N \times N$ non-negative P, if

$$\sum_{m=0}^{n-1} P^m/n \longrightarrow P^* \text{ as } n \to \infty \quad (\textit{called Cesaro summability})$$

then it follows that:

$$\lim_{\beta \to 1^-} (1-\beta) \sum_{n=0}^{\infty} \beta^n P^n = P^*$$

(Mine & Osaki [2])

Lemma 1.29

For any $N \times N$ a Markov transition probability matrix, then

a.

$$\lim_{n \to \infty} \sum_{m=0}^{n-1} P^m / n = P^*$$

where P^* is also a Markov transition probability matrix such that

$$PP^* = P^*P = P^*P^* = P^*$$

b. *$I - (P - P^*)$ is nonsingular and*

$$H(\beta) = \sum_{n=0}^{\infty} \beta^n (P^n - P^*) - P^* \longrightarrow H = (I - P + P^*)^{-1} - P^* \text{ as } \beta \to 1^-$$

and

$$H(\beta)P^* = P^*H(\beta) = HP^* = P^*H = 0; \ (I-P)H = H(I-P) = I - P^*$$

(Mine & Osaki [2])

Theorem 1.30 : Existence of stationary optimal strategy

There exists a stationary optimal strategy in the non-discounting case for a Markovian decision process with a finite state space S and a finite policy space K.

Proof: There are only a finite number of stationary strategies since the policy space K is finite. Thus according to Theorem 1.21, it is possible to choose a sequence $\{\beta_\nu\}$ of discount factors converging to 1 for which the corresponding stationary β-optimal strategies $d_{\beta_\nu}^{\infty}$ are all the same i.e.

$$\lim_{n \to \infty} \beta_\nu = 1; \quad \delta^* = d^{*\infty} \quad \text{where } d^* = d_{\beta_\nu} \text{ for } \nu = 1, 2, \ldots$$

Then using Lemma 1.29 and Lemma 1.28, it follows that for any strategy δ

$$\begin{aligned}
&\liminf_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} P_m(\delta_m) R(d_{m+1}) \\
&\leq \lim_{\nu\to\infty} (1-\beta_\nu) V_{\beta_\nu}(\delta) \\
&\leq \lim_{\nu\to\infty} (1-\beta_\nu) V_{\beta_\nu}(\delta^*) \\
&= \lim_{\nu\to\infty} (1-\beta_\nu) \sum_{m=0}^{\infty} \beta_\nu^m P^m(d^*) R(d^*) \\
&= P^*(d^*) R(d^*)
\end{aligned}$$

where $V_\beta(\delta)$ is the discounted total expected return over infinite time period for the chosen strategy δ at discount rate β $(0 \leq \beta < 1)$. Then $\delta^* = d^{*\infty}$ is a stationary optimal strategy (Mine & Osaki [2]) □.

1.3.3 Completely-Ergodic Markovian Decision Process

Throughout the rest of this chapter, we will restrict our discussion to two special classes of Markov processes: the **completely-ergodic MDP** and the **terminating MDP**. Each of these cases lead to an appropriate policy iteration method.

Definition 1.31 : Completely-Ergodic Markovian Decision Process

A Markovian decision process is said to be completely ergodic if the Markov chain governing transitions among states is ergodic regardless of the decision (action) taken at each stage of the process.

From this point on, $G(d)$ and $V(d)$ will be used in place of $G(d^\infty)$ and $V(d^\infty)$ for a stationary strategy $d^\infty = (d, d, \ldots)$ in the non-discounting case as long as no confusion arises.

Remark 1.32 : Average return per unit time for completely-ergodic MDP

a. *For an ergodic Markov process with transition probability matrix P, there exists a*

unique steady state distribution π across the state space S that satisfies

$$\pi = \pi P \quad \textit{and } \pi\mathbf{1} = 1 \textit{ where } \pi_i > 0 \textit{ for all } i \in S$$

Although π may depend on the decisions taken in the process, it does not depend on the initial state distribution.

b. *For a stationary strategy d^∞ of a completely ergodic process*

$$G(d) = \lim_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} P^m(d)R(d) = P^*(d)R(d)$$

where P^* is the limiting matrix whose rows are all identical i.e. $P^*(d) = \mathbf{1}\pi(d)$, where $\pi(d)$ is the steady state distribution associated with decision d. Consequently G(d) has identical components, namely G(d)=g(d)$\mathbf{1}$ where

$$g(d) = \sum_{i=1}^{N} \pi_i(d) r_i(d) = \pi(d)R(d)$$

Therefore, g(d), the average gain or return per unit time associated with stationary strategy d^∞, is the same for all initial states (Mine & Osaki [2]; Ross [1]).

The next two theorems provide the basis for the policy iteration method used to find the stationary optimal strategy in the non-discounting case.

Theorem 1.33

If $P^n(d) \to P^*(d)$ as $n \to \infty$, then we have

$$V^n(d) = ng(d)\mathbf{1} + V(d) + \varepsilon(n,d) \textit{ for } n > 0$$

where $\varepsilon(n,d) \to 0$ as $n \to \infty$, and $V(d) = H(d)R(d)$.

Proof: For any stationary strategy $d^\infty = (d, d, \ldots)$, the non-discounted total expected return value vector $V^n(d)$ up to time n with $V^0 = 0$ yields the following for $n = 1, 2, \ldots$

$$
\begin{aligned}
V^n(d) &= \sum_{m=0}^{n-1} P^m(d)R(d) = \sum_{m=0}^{n-1} \left[P^m(d) - P^*(d) + P^*(d)\right] R(d) \\
&\quad \textit{by the relation } P^m - P^* = [P - P^*]^m \textit{ for } m = 1, 2, \ldots \textit{ from Lemma } 1.29a \\
&= \left(\sum_{m=0}^{n-1} [P(d) - P^*(d)]^m + (n-1)P^*(d)\right) R(d) \\
&= nP^*(d)R(d) + \left(\sum_{m=0}^{n-1} [P(d) - P^*(d)]^m - P^*(d)\right) R(d) \\
&= nP^*(d)R(d) + \left(\sum_{m=0}^{\infty} [P(d) - P^*(d)]^m - P^*(d) - \sum_{m=n}^{\infty} [P(d) - P^*(d)]^m\right) R(d) \\
&\quad \textit{by complete ergodicity that leads to } [P(d) - P^*(d)]^m \ \to 0 \\
&= ng(d)\mathbf{1} + \left[(I - P(d) + P^*(d))^{-1} - P^*(d)\right] R(d) - \sum_{m=n}^{\infty} [P(d) - P^*(d)]^m R(d) \\
&= ng(d)\mathbf{1} + V(d) + \varepsilon(n, d)
\end{aligned}
$$

where

$$V(d) = \left[(I - P(d) + P^*(d))^{-1} - P^*(d)\right] R(d) = H(d)R(d)$$

as defined in Lemma 1.29b, and

$$\varepsilon(n, d) = -\sum_{m=n}^{\infty} [P(d) - P^*(d)]^m R(d) \to 0 \textit{ as } n \to \infty$$

Corollary 1.34

As we let $n \to \infty$

$$g(d)\mathbf{1} + V(d) = R(d) + P(d)V(d)$$

Proof: Choose n sufficiently large so that $|\varepsilon(n, d)|$ is arbitrary small. Then

$$
\begin{aligned}
V^n(d) &= R(d) + P(d)V^{n-1}(d) \\
&\approx R(d) + P(d)[(n-1)g(d)\mathbf{1} + V(d)] \quad \textit{by Theorem } 1.33 \\
&= R(d) + (n-1)g(d)P(d)\mathbf{1} + P(d)V(d) \\
&= R(d) + (n-1)g(d)\mathbf{1} + P(d)V(d) \quad \textit{since } P(d)\mathbf{1} = \mathbf{1}
\end{aligned}
$$

Thus for n sufficiently large

$$V^n(d) \approx ng\mathbf{1} + V(d) \approx R(d) + (n-1)g(d)\mathbf{1} + P(d)V(d)$$

Letting $n \to \infty$, we obtain $g(d)\mathbf{1} + V(d) = R(d) + P(d)V(d)$

□

Remark 1.35 : Relative values

For any constant c, $V(d) + c\mathbf{1}$ also satisfies $g(d)\mathbf{1} + V(d) = R(d) + P(d)V(d)$. Thus $v_i(d)$ are referred to as the **relative** *values associated with the stationary strategy d^∞*

Theorem 1.36

If there is a stationary strategy $\tilde{d}^\infty \in D$ for which

$$R(\tilde{d}) + P(\tilde{d})V(d) > R(d) + P(d)V(d)$$

then

$$g(\tilde{d}) > g(d)$$

Proof: For stationary strategies d^∞ and $\tilde{d}^\infty$, it follows from Corollary 1.34 that

$$g(\tilde{d})\mathbf{1} + V(\tilde{d}) = R(\tilde{d}) + P(\tilde{d})V(\tilde{d}) \quad g(d)\mathbf{1} + V(d) = R(d) + P(d)V(d)$$

Now let

$$\Delta g = g(\tilde{d}) - g(d),\ \Delta V = V(\tilde{d}) - V(d),\ \gamma = R(\tilde{d}) - R(d) + [P(\tilde{d}) - P(d)]V(d)$$

Since $\gamma > 0$ from the assumption in the theorem, it follows that

$$\begin{aligned}\Delta g\mathbf{1} + \Delta V &= R(\tilde{d}) - R(d) + P(\tilde{d})V(\tilde{d}) - P(d)V(d) \\ &= R(\tilde{d}) - R(d) + [P(\tilde{d}) - P(d)]V(d) + P(\tilde{d})(V(\tilde{d}) - V(d)) \\ &= \gamma + P(\tilde{d})\Delta V\end{aligned}$$

If we multiply the steady state probability vector $\pi(\tilde{d})$ corresponding to $\tilde{d}^\infty$ on both sides of the last equation, then one obtains $\Delta g = \pi(\tilde{d})\gamma > 0$ and hence $\tilde{d}^\infty$ is an improved stationary strategy such that $g(\tilde{d}) > g(d)$.

□

1.3.4 Policy Iteration Method for an Infinite Horizon

VDO: Value Determination Operation

Obtain the average return $g(d)$ per unit time and the relative value vector $V(d)$ for stationary strategy d^∞ where $d \in D$ by solving the linear system

$$g(d) + v_i(d) = r_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(d)$$

with $v_N(d) = 0$ and $d(i) = k$ for $i = 1, 2, \ldots, N$.

PIA: Policy Improvement Algorithm

For each state $i \in S$ use $g(d)$ and $V(d)$ obtained from the **VDO** to find an action $\tilde{k} \in K_i$ which maximizes

$$r_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(d)$$

and let $\tilde{d}(i) = \tilde{k}$. If the maximum value exceeds $g(d) + v_i(d)$ for any state i, then $\tilde{d}^\infty$ is an improved stationary strategy relative to d^∞.

The policy iteration method alternates the **VDO** and **PIA** steps until no improved stationary strategy is produced.

Remark 1.37 : Implementation of the policy improvement method

The above policy improvement method to find a stationary optimal strategy and the corresponding average return per unit time over an infinite horizon for a discrete time, completely ergodic MDP without discounting can be implemented through the program **dt_MDP** *which uses the procedures* **PIA_mdp** *and* **VDO_mdp** *with appropriate inputs* (Appendices A.1, B.2 and B.3).

1.3.5 Example: Taxicab Problem

Now let us present an example to illustrate how the policy improvement method involving the **PIA** and **VDO** yields a stationary optimal strategy and the corresponding optimized average return under the stationary optimal strategy (Mine & Osaki [2]).

Consider the three possible courses of action a taxicab driver can choose if his serving territory encompasses three towns A, B and C:

Action 1: cruise around hoping to pick up a passenger on the road by being hailed;

Action 2: drive to the nearest cab stand and wait in line for a passenger;

Action 3: park and wait for a radio-car call with instructions.

The third action is not available in town B since there is no radio-car service in that town. The rewards represent the driver's income from a trip after all necessary expenses have been deducted. For example, if action 1 and 2 are taken, then the cost of cruising or driving to the nearest cab stand must be included in calculating the rewards. The transition probabilities and the net incomes for transitions associated with different actions are given in the following table.

Table 1.2: Data of Taxicab Problem

State i	Action k	Probability p_{i1}^k	p_{i2}^k	p_{i3}^k	Trip Reward r_{i1}^k	r_{i2}^k	r_{i3}^k	System Reward $r_i^k = \sum_{j=1}^3 p_{ij}^k r_{ij}^k$
1	1	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{4}$	10	4	8	8.00
	2	$\frac{1}{16}$	$\frac{3}{4}$	$\frac{3}{16}$	8	2	4	2.75
	3	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{5}{8}$	4	6	4	4.25
2	1	$\frac{1}{2}$	0	$\frac{1}{2}$	14	0	18	16.00
	2	$\frac{1}{16}$	$\frac{7}{8}$	$\frac{1}{16}$	8	16	8	15.00
3	1	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	10	2	8	7.00
	2	$\frac{1}{8}$	$\frac{3}{4}$	$\frac{1}{8}$	6	4	2	4.00
	3	$\frac{3}{4}$	$\frac{1}{16}$	$\frac{3}{16}$	4	0	8	4.50

Step 1 (**VDO**):

Take initial strategy d_0^∞ where $d_0 = (1, 1, 1)$ according to Remark 1.22, and solve the following linear system for $g(d_0)$, $V(d_0)$ with $v_3 = 0$

$$g\begin{pmatrix}1\\1\\1\end{pmatrix} + \begin{pmatrix}v_1\\v_2\\v_3\end{pmatrix} = \begin{pmatrix}8\\16\\7\end{pmatrix} + \begin{pmatrix}\frac{1}{2} & \frac{1}{4} & \frac{1}{4}\\ \frac{1}{2} & 0 & \frac{1}{2}\\ \frac{1}{4} & \frac{1}{4} & \frac{1}{2}\end{pmatrix}\begin{pmatrix}v_1\\v_2\\v_3\end{pmatrix}$$

$$\Longrightarrow V(d_0) = \begin{pmatrix}1.33\\7.47\\0\end{pmatrix},\ g(d_0) = 9.20$$

Step 2 (**PIA**):

For each i we will identify the action k which maximizes $r_i^k + \sum_{j=1}^3 p_{ij}^k v_j(d_0)$.

(1) For $i = 1$:

$$\begin{pmatrix}8.00\\2.75\\4.25\end{pmatrix} + \begin{pmatrix}\frac{1}{2} & \frac{1}{4} & \frac{1}{4}\\ \frac{1}{16} & \frac{3}{4} & \frac{3}{16}\\ \frac{1}{4} & \frac{1}{8} & \frac{5}{8}\end{pmatrix}\begin{pmatrix}1.33\\7.47\\0\end{pmatrix} = \begin{pmatrix}10.53\\8.44\\5.52\end{pmatrix}$$

Hence $\tilde{k} = 1$ results in maximum for $i = 1$;

(2) For $i = 2$:

$$\begin{pmatrix} 16.00 \\ 15.00 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{16} & \frac{7}{8} & \frac{1}{16} \end{pmatrix} \begin{pmatrix} 1.33 \\ 7.47 \\ 0 \end{pmatrix} = \begin{pmatrix} 16.67 \\ 21.62 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 2$;

(3) For $i = 3$:

$$\begin{pmatrix} 7.00 \\ 4.00 \\ 4.50 \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ \frac{3}{4} & \frac{1}{16} & \frac{3}{16} \end{pmatrix} \begin{pmatrix} 1.33 \\ 7.47 \\ 0 \end{pmatrix} = \begin{pmatrix} 9.20 \\ 9.77 \\ 5.96 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 3$. Therefore d_1^∞ where $d_1 = (1, 2, 2)$ is an improved stationary strategy.

Step 3 (**VDO**):

Using the improved stationary strategy d_1^∞, solve the following linear system for $g(d_1)$, $V(d_1)$ with $v_3 = 0$

$$g \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 15 \\ 4 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{16} & \frac{7}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

$$\Longrightarrow V(d_1) = \begin{pmatrix} -3.88 \\ 12.85 \\ 0 \end{pmatrix}, \; g(d_1) = 13.15 > g(d_0)$$

Step 4 (**PIA**):

For each i we will identify the action k which maximizes $r_i^k + \sum_{j=1}^3 p_{ij}^k v_j(d_1)$.

(1) For $i = 1$:

$$\begin{pmatrix} 8.00 \\ 2.75 \\ 4.25 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{16} & \frac{3}{4} & \frac{3}{16} \\ \frac{1}{4} & \frac{1}{8} & \frac{5}{8} \end{pmatrix} \begin{pmatrix} -3.88 \\ 12.85 \\ 0 \end{pmatrix} = \begin{pmatrix} 9.27 \\ 12.15 \\ 4.89 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 1$;

(2) For $i = 2$:

$$\begin{pmatrix} 16.00 \\ 15.00 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{16} & \frac{7}{8} & \frac{1}{16} \end{pmatrix} \begin{pmatrix} -3.88 \\ 12.85 \\ 0 \end{pmatrix} = \begin{pmatrix} 14.06 \\ 26.00 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 2$;

(3) For $i = 3$:

$$\begin{pmatrix} 7.00 \\ 4.00 \\ 4.50 \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ \frac{3}{4} & \frac{1}{16} & \frac{3}{16} \end{pmatrix} \begin{pmatrix} -3.88 \\ 12.85 \\ 0 \end{pmatrix} = \begin{pmatrix} 9.24 \\ 13.15 \\ 2.39 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 3$. Therefore d_2^∞ where $d_2 = (2, 2, 2)$ is another improved stationary strategy.

Step 5 (**VDO**):

Using the improved stationary strategy d_2^∞, solve the following linear system for $g(d_2)$, $V(d_2)$ with $v_3 = 0$

$$g \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} 2.75 \\ 15 \\ 4 \end{pmatrix} + \begin{pmatrix} \frac{1}{16} & \frac{3}{4} & \frac{3}{16} \\ \frac{1}{16} & \frac{7}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

$$\Longrightarrow V(d_2) = \begin{pmatrix} -1.18 \\ 12.66 \\ 0 \end{pmatrix}, \; g(d_2) = 13.34 > g(d_1)$$

Step 6 (**PIA**):

For each i we will identify the action k which maximizes $r_i^k + \sum_{j=1}^3 p_{ij}^k v_j(d_2)$.

(1) For $i = 1$:

$$\begin{pmatrix} 8.00 \\ 2.75 \\ 4.25 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{16} & \frac{3}{4} & \frac{3}{16} \\ \frac{1}{4} & \frac{1}{8} & \frac{5}{8} \end{pmatrix} \begin{pmatrix} -1.18 \\ 12.66 \\ 0 \end{pmatrix} = \begin{pmatrix} 10.58 \\ 12.66 \\ 11.87 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 1$;

(2) For $i = 2$:

$$\begin{pmatrix} 16.00 \\ 15.00 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{16} & \frac{7}{8} & \frac{1}{16} \end{pmatrix} \begin{pmatrix} -1.18 \\ 12.66 \\ 0 \end{pmatrix} = \begin{pmatrix} 15.41 \\ 26.00 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 2$;

(3) For $i = 3$:

$$\begin{pmatrix} 7.00 \\ 4.00 \\ 4.50 \end{pmatrix} + \begin{pmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\ \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ \frac{3}{4} & \frac{1}{16} & \frac{3}{16} \end{pmatrix} \begin{pmatrix} -1.18 \\ 12.66 \\ 0 \end{pmatrix} = \begin{pmatrix} 9.87 \\ 13.34 \\ 4.41 \end{pmatrix}$$

Hence $\tilde{k} = 2$ results in maximum for $i = 3$. Therefore no improved stationary strategy can be found and hence d_2^∞ is the optimal stationary strategy and the corresponding optimized average return per unit time is $g(d_2)$=\$13.34.

1.4 Terminating Markovian Decision Processes

In this section, we discuss a discrete time MDP for the special case in which the system has a common absorbing state. Once the system makes a transition into the absorbing state, it will stay there forever. Further, we assume that the absorbing state is accessible from any other state in a finite number of transitions. In other words, all the other states are transient and the process will be terminated when the system enters the absorbing state.

Definition 1.38 : Terminating Assumption

The absorbing state, assumed to be state 1, will be reached from any transient state $i \in S_T = \{2, 3, \ldots, N\}$ *in a finite number of transitions with probability one, regardless of the decision made and initial state from which the process started.*

In order to ensure this terminating assumption, it suffices to assume that

$$\sum_{j=2}^{N} p_{ij}^k < 1 \text{ for at least one transient state } i \in S_T \text{ and for all } k \in K_i$$

i.e. that the absorbing state 1 will be reached from transient state i in a single transition with positive probability (Mine & Osaki [2]).

By the terminating assumption, we know that the total expected return up to the time the system enters the absorbing state is always finite, even when no discount factor is used. Hence, we can deal with a terminating MDP in either the discounting or non-discounting case by applying the theorems obtained earlier for a MDP with

discounting as well as the corresponding policy iteration method with some minor adjustments.

The total expected return up to the time the system enters the absorbing state is

$$V_\beta(\delta) = \sum_{n=0}^{\infty} \beta P_n(\delta_n) R(d_{n+1})$$

where $\delta = (d_1, d_2, \ldots, d_n, \ldots)$ with $d_n \in D$, $R(d_n)$ is the $(N-1)$ dimensional vector obtained by deleting the first component of the original system rewards vector R, and $P_n(\delta_n)$ is the $(N-1) \times (N-1)$ matrix obtained by eliminating the first row and the first column of the original Markov transition probability matrix. Notice that $p_{11}^k \equiv 1$ for all $k \in K_1$ since state 1 is the absorbing state. Furthermore, we simply let $\beta = 1$ in the non-discounting case.

Theorem 1.39 *If $V_\beta(\delta^*) \geq V_\beta(d, \delta^*)$ for all $d \in D$, then δ^* is an optimal strategy* (Mine & Osaki [2]).

We omit the proof since it is almost identical to that of Theorem 1.16.

Theorem 1.40 *Exactly one of the following has to occur:*

a. *$V_\beta(d^\infty) \geq V_\beta(\tilde{d}, d^\infty)$ for all $\tilde{d} \in D$, in which case d^∞ is optimal.*

b. *$V_\beta(d^\infty) < V_\beta(\tilde{d}, d^\infty) \leq V_\beta(\tilde{d}^\infty)$ for some $\tilde{d} \in D$, in which case d^∞ is an improved stationary strategy.*

Proof: If $V_\beta(d^\infty) \geq V_\beta(\tilde{d}, d^\infty)$ for all $\tilde{d} \in D$, then d^∞ is a stationary optimal strategy by Theorem 1.39. Otherwise, $V_\beta(d^\infty) < V_\beta(\tilde{d}, d^\infty)$ for some $\tilde{d} \in D$ in which case it follows from the argument in Theorem 1.17 that

$$V_\beta(\tilde{d}, d^\infty) \leq V_\beta(\overbrace{\tilde{d}, \tilde{d}, \ldots, \tilde{d}}^{n}, d^\infty) \quad n = 1, 2, \ldots$$

Therefore, by letting $n \to \infty$

$$V_\beta(d^\infty) < V_\beta(\tilde{d}, d^\infty) \leq V_\beta(\tilde{d}^\infty)$$

and d^∞ is an improved stationary strategy.

□

From Theorem 1.39 and Theorem 1.40 we can establish the following:

Theorem 1.41 *For a terminating MDP there is a stationary optimal strategy* (Mine & Osaki [1]).

1.4.1 Policy Improvement Method for an Infinite Horizon

The policy improvement method for a terminating MDP is very similar to that for a MDP with discounting except that the dimension of matrices and vectors used in the procedure are $N-1$ instead of N. Moreover, the discount factor can be set to 1 for the non-discounting case since the expected total return generated by the system for the set S_T of transient states is finite if the process satisfies the terminating assumption 1.38.

VDO: Value Determination Operation

Obtain the total expected return vector $V_\beta(d)$ for stationary strategy d^∞ where $d \in D$ by solving the linear system

$$v_i(\beta, d) = r_i^k + \beta \sum_{j=2}^{N} p_{ij}^k v_j(\beta, d)$$

with $d(i) = k$ for $i = 1, 2, \ldots, N$.

PIA: Policy Improvement Algorithm

For each state $i \in S$ use $V_\beta(d)$ obtained from the **VDO** to find an action $\tilde{k} \in K_i$ which maximizes

$$r_i^k + \beta \sum_{j=2}^{N} p_{ij}^k v_j(\beta, d) \qquad 0 < \beta \leq 1.$$

and let $\tilde{d}(i) = \tilde{k}$. If the maximum value exceeds $v_i(d)$ for any state i, then $\tilde{d}^\infty$ is an improved stationary strategy relative to d^∞.

The policy iteration method alternates the **VDO** and **PIA** steps until no improved stationary strategy is produced.

Remark 1.42 : Implementation of the policy improvement method

The above policy improvement method to find a stationary optimal strategy and the corresponding optimized total expected return up to the time the process enters the absorbing state, for a discrete time terminating MDP with orwithout discounting can be implemented by using the program **dt_MDP** *which uses the procedures* **PIA_mdp** *and* **VDO_mdp** *with appropriate inputs* (Appendices A.1, B.2 and B.3).

Chapter 2

Discrete Time Semi-Markovian Decision Process

Contrary to a discrete time Markov process in which a transition is made at every time point and the time between consecutive transitions is always one unit time-interval, we now deal with a more general class of Markov processes where the holding time may be any multiple of the unit time-interval and may depend on both the current state and the state to be visited.

2.1 Discrete Time Semi-Markov Process

In a discrete time semi-Markov process, each transition from a current state to a successor state is governed by the transition probabilities of a Markov process, but the occupancy time in the current state is an integer-valued random variable that can be any multiple of the unit time-interval. This is in contrast to a Markov process where transitions are always made at unit time-intervals. When the process enters into a state, the next state into which it will move is determined according to the Markov transition probabilities. However, before it actually makes the transition, the process will hold for a random period of time that depends on the current state and

the next state to be visited.

From the description above, we see that a semi-Markov process behaves just like a Markov process at the time of transition except that the holding times between transitions are determined by integer-valued random variables governed by a discrete holding time distribution. Thus such a process is referred to as an *embedded* Markov process.

In order to carry out analysis for a semi-Markov process, it is necessary to introduce some of the distributions used to describe the process. These include the distributions of holding-time and waiting-time, as well as their corresponding means, and the interval transition probabilities (Howard [4], Vol II).

Definition 2.1 : Distributions for a discrete time Markov Process

a. *Let* $H(n) = (h_{ij}(n))_{N\times N}$ *be the matrix of holding time probabilities for* $n = 1, 2, \ldots$. *Assume that* $h_{ij}(0) = 0$ *for all* $i, j \in S$.

b. *Let* $W(n) = diag(w_{ii}(n))_{N\times N}$ *and* $W^*(n) = diag(w^*_{ii}(n))_{N\times N}$ *be the matrices whose diagonal entries are the waiting time probabilities and cumulative waiting time probabilities respectively for* $n = 1, 2, \ldots$

$$w_{ii}(n) = \sum_{j=1}^{N} p_{ij}h_{ij}(n) \quad w^*_{ii}(n) = \sum_{m=0}^{n} w_{ii}(m) = \sum_{j=1}^{N} p_{ij} \sum_{m=0}^{n} h_{ij}(m) \;\; i, j \in S.$$

c. *Let* $\bar{H} = (\bar{h}_{ij})_{N\times N}$ *and* $\bar{W} = diag(\bar{w}_{ii})_{N\times N}$ *be the matrices of the means of the holding time and the waiting time distributions respectively*

$$\bar{h}_{ij} = \sum_{n=0}^{\infty} nh_{ij}(n) \qquad \bar{w}_{ii} = \sum_{n=0}^{\infty} nw_{ii}(n)dt = \sum_{j=1}^{N} p_{ij}\bar{h}_{ij} \quad i, j \in S.$$

d. *Let* $\Phi(n) = (\phi_{ij}(n))_{N\times N}$ *be the matrix whose entries are the interval transition probabilities for the embedded Markov process for* $n = 1, 2, \ldots$

$$\phi_{ij}(n) = \delta_{ij}(1 - w^*_{ii}(n)) + \sum_{k=1}^{N} p_{ik} \sum_{m=0}^{n} h_{ik}(m)\phi_{kj}(n - m) \quad i, j \in S.$$

where $\delta_{ij} = 1$ *if i=j and 0 otherwise.*

The first term represents the probability that the transition from state i occurs after time n, i.e. that the process remains in state i at least through time n. The second term represents the probability that the process makes its first transition from state i to another state, say state k, at some time $m < n$ and then enters state j in the remaining time $n - m$. We can express $\Phi(n)$ in matrix form

$$\Phi(n) = (\mathbf{I} - W^*(n)) + \sum_{m=0}^{n} (P \odot H(m))\Phi(n-m) \quad for\ n = 0, 1, 2, \ldots$$

where $\odot$ denotes the direct product of two matrices i.e. $P \odot H(m) = (p_{ij}h_{ij})_{N\times N}$ (Howard [4], Vol II).

For an ergodic Markov process the limiting interval transition probability matrix $\Phi = \lim_{n\to\infty} \Phi(n)$ is given by

$$\phi_{ij} = \frac{\pi_j \bar{w}_{jj}}{\sum_{l=1}^{N} \pi_l \bar{w}_{ll}} = \phi_j \quad for\ all\ i, j \in S$$

where $\pi_i = (\pi_1, \ldots, \pi_N)$ is the steady state probability vector for the embedded ergodic Markov chain. Note that for an ergodic Markov process the rows of Φ are identical (Howard [4], Vol II).

2.2 Discrete Time Semi-Markovian Decision Process with Discounting

Before we get into a formal discussion of the optimization procedures, we need to introduce the reward structure for the system rewards.

Definition 2.2 : Yields and bonus

a. *Let* $y_{ij}^k(l)$ *be the yield that the system earns at time l for the occupancy of state i*

over the unit time interval (l,l+1), when action k is chosen upon entry into state i and where a successor state j has been determined according to the transition probabilities p_{ij}^k.

b. *Let* $b_{ij}^k(n)$ *be the bonus earned when the transition from state i to state j is made at time n* (Howard [4], Vol II).

From the definition above, it is clear that a yield $y_{ij}^k(l)$ will be earned if the process still remains in state i at time l or a bonus $b_{ij}^k(l)$ will be earned if the process leaves the state i and enters state j at time l. The combination of yield and bonus structure provides a powerful modeling capability. In an inventory system in which the states represent the number of items in stock and transitions represent sales or orders, we can describe the yield in terms of the cost of maintaining the inventory and carrying charges, and describe the bonus in terms of the cost of ordering and the revenue associated with sales.

Lemma 2.3

If action k is taken when the system enters state i and the system then makes a transition to state j after holding for n time units in state i, then the system reward r_{ij}^k *earned as a result of the occupancy and transition is*

$$b_{ij}^k(n) + \sum_{l=0}^{n-1} y_{ij}^k(l) \qquad \textit{for the non-discounting case}$$

$$\beta^n b_{ij}^k(n) + \sum_{l=0}^{n-1} \beta^l y_{ij}^k(l) \quad \textit{for the discounting case}$$

The second expression tells us that if we have a discount factor β, then the contribution to the system reward generated from yields y_{ij}^k and bonus b_{ij}^k on a unit time scale are reduced to present values, by multiplying them by the successive terms of the geometric sequence in β i.e. $1, \beta, \beta^2, \ldots$ (Howard [4], Vol I & II).

2.2.1 Value Iteration Procedure for a Finite Horizon

Definition 2.4 : Total discounted expected return for a finite horizon

a. *Let* $V_\beta^n(\delta_n) = (v_1(n,\beta,\delta_n),\ldots,v_N(n,\beta,\delta_n))^T$ *be the vector of total discounted expected returns for* n *time periods associated with the policy* $\delta_n = (d_1,\ldots,d_n)$, *where* $d_m = (d_m(1),\ldots,d_m(N))$ *is the decision vector whose components specify the action to be taken in each state when the time remaining is* $l = n+1-m$ *for* $m = 1,\ldots,n$.

b. *Let* $V^0 = (v_1(0),\ldots,v_N(0))^T$ *be the terminal reward vector.*

The total expected returns $v_i(n,\beta,\delta_n)$ for horizon n are defined recursively as follows (Howard [4], Vol II):

$$\begin{aligned} v_i(n,\beta,\delta_n) &= \sum_{j=1}^{N} p_{ij}^k \sum_{m=n+1}^{\infty} h_{ij}^k(m)\left(\sum_{l=0}^{n-1}\beta^l y_{ij}^k(l) + \beta^n v_i(0)\right) \\ &+ \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m)\left(\sum_{l=0}^{m-1}\beta^l y_{ij}^k(l) + \beta^m b_{ij}^k(m) + \beta^m v_j(n-m,\beta,\delta_{n-m})\right) \end{aligned}$$

where $\delta_n = (d,\delta_{n-1})$ with $d(i) = k$ for $i = 1,2,\ldots,N$ and $n = 1,2,\ldots$

We give an intuitive argument for the above equation. When the process enters state i at time 0 and leaves i at time m to enter state j, then the discounted value of all yields for a chosen action in state i at time zero is

$$y_{ij}^k(m,\beta) = \sum_{l=0}^{m-1}\beta^l y_{ij}^k(l) \quad i,j \in S.$$

Now consider the time at which the transition occurs. If the first transition out of state i occurs after time n, then state i is occupied for the entire interval of length n. In this case, the process will generate discounted rewards according to the yields y_{ij}^k throughout the time interval n (see Definition 2.2), and it will also produce a terminal reward which has a present value $\beta^n v_i(0)$. Thus the first term in the equation is obtained by summing these discounted rewards multiplied by the

corresponding Markov transition probabilites over all possible successor states and over all holding times beyond time n.

On the other hand, if the first transition out of state i into state j occurs at time m before time n, then in addition to the yields y_{ij}^k produced throughout the time interval of length m, the bonus will produce a contribution at time m with the present value $\beta^m b_{ij}^k(m)$. The second term is obtained by summing these discounted rewards multiplied by the Markov transition probabilites over all possible successor states and over all holding times prior to the transition time m $(m = 1, 2, \ldots, n)$.

The expression for $v_i(n, \beta, d(n))$ can be rewritten in a way which provides us with a better understanding of the system rewards structure.

$$v_i(n, \beta, \delta_n) = \ddot{y}_i^k(n, \beta) + \beta^n v_i(0) \ddot{w}_{ii}^k(n) + r_i^k(n, \beta) + \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m) \beta^m v_j(n-m, \beta, \delta_{n-m})$$

The first term on the right hand side denoted by $\ddot{y}_i^k(n, \beta)$ is the contribution to the total expected return from the discounted yields $y_{ij}^k(n, \beta)$ for chosen action k when the first transition out of state i occurs after time n; the second term is the contribution from the terminal reward when the first transition is made after time n; the third term denoted by $r_i^k(n, \beta)$ is the contribution of both the yield and the bonus for the first transition when it occurs at or before time n; the last term is the expected total return from future transitions when the first transition occurs at or before time n. Mathematically, these contributions are given by

$$\begin{aligned} \ddot{y}_i^k(n, \beta) &= \sum_{j=1}^{N} p_{ij}^k \ddot{h}_{ij}^k(n) y_{ij}^k(n, \beta) \\ r_i^k(n, \beta) &= \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m) (y_{ij}^k(m, \beta) + \beta^m b_{ij}^k(m)) \end{aligned}$$

where $\ddot{h}_{ij}^k(n) = \sum_{m=n+1}^{\infty} h_{ij}^k(m)$ and $\ddot{w}_{ii}^k(n) = \sum_{m=n+1}^{\infty} w_{ii}^k(m) = \sum_{j=1}^{N} p_{ij}^k \ddot{h}_{ij}^k(n)$ are the complements of the holding time distribution and waiting time distribution respectively, associated with the action k taken in state i, and $y_{ij}^k(n, \beta) = \sum_{l=0}^{n-1} \beta^l y_{ij}^k(l)$ is

the contribution to the reward from the discounted yields during the holding time in state i prior to entering the successor state j.

We now introduce the value iteration procedure used to obtain the β-optimal policy δ_n^* for time horizon n (Howard [4], Vol II).

VIP: Value Iteration Procedure

The maximum total expected return vector $V_\beta^n = \max V_\beta^n$ and the associated β-optimal policy δ_n^* for time horizon n are obtained recursively using V_β^l and δ_l^* for horizon $l < n$ as follows: for each horizon $n = 1, 2, \ldots$, obtain

$$v_i(n,\beta) = \max_k \left\{ \ddot{y}_i^k(n,\beta) + \beta^n v_i(0)\ddot{w}_{ii}^k(n) + r_i^k(n,\beta) + \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m)\beta^m v_j(n-m,\beta) \right\}$$

and the action $\tilde{k} \in K_i$ which produces the maximum for $i = 1, \ldots, N$. Then the optimal policy for time horizon n is $\delta_n^* = (d, \delta_{n-1}^*)$ i.e. the optimal policy δ_{n-1}^* preceeded by another new decision vector $d = (\tilde{k}_1, \tilde{k}_2, \ldots, \tilde{k}_N)$. In this way, we recursively generate V_β^n and the optimal policy $\delta_n^* = (d_1, \ldots, d_n)$ where d_m is the optimal decision vector for the initial decision when the time remaining is $l = n + 1 - m$ for $m = 1, \ldots, n$.

Remark 2.5 Implementation of the value iteration procedure

The above value iteration procedure to find a time-dependent optimal policy and the corresponding total discounted expected return over a finite horizon for a discrete time semi-MDP with discounting can be implemented by using the program **dt_SMDP** *which uses the procedure* **VIP_smdp** *with appropriate inputs* (Appendices A.2 and B.4).

2.2.2 Policy Iteration Method for an Infinite Horizon

Lemma 2.6 Total discounted expected return in the long run

For each stationary strategy d^∞ where $d \in D$, the vector $V_\beta(d)$ of total discounted expected returns over an infinite horizon satisfies the linear system

$$v_i(\beta, d) = r_i^k(\beta) + \sum_{j=1}^{N} p_{ij}^k h_{ij}^{gk}(\beta) v_j(\beta, d)$$

where

$$v_i(\beta, d) = \lim_{n\to\infty} v_i(n, \beta, d) \quad r_i^k(\beta) = \lim_{n\to\infty} r_i^k(n, \beta) \quad \textit{and} \quad h_{ij}^{gk}(z) = \sum_{n=1}^{\infty} z^n h_{ij}^k(n)$$

for all $i, j \in S$. Note that $h_{ij}^g(z)$ is the geometric transform of the holding time distribution $h_{ij}(n)$ (Howard [4], Vol I). *Hence $V_\beta(d)$ can be solved in matrix form*

$$V_\beta(d) = (\mathbf{I} - P(d) \odot H^g(\beta, d))^{-1} R_\beta(d)$$

This follows from the fact that $\ddot{h}_{ij}^k(n)$ and β^n both approach 0 as $n \to \infty$ so that both $\ddot{y}_i^k(n, \beta)$ and $\beta^n v_i(0)\ddot{w}_i^k(n)$ approach 0 as well.

The following theorem provides the basis for the policy iteration method for a discrete time semi-MDP with discounting.

Theorem 2.7 : Existence of stationary β-optimal strategy

There exists a stationary β-optimal strategy for a semi-Markovian decision process with a finite state space S and a finite policy space K (Howard [4], Vol II).

For an infinite time horizon, we have the following policy iteration method consisting of the Value Determination Operation (**VDO**) and the Policy Improvement Algorithm (**PIA**) for a discrete-time semi-MDP with discounting.

VDO: Value Determination Operation

Obtain the total discounted expected return vector $V_\beta(d)$ for stationary strategy d^∞ where $d \in D$ by solving the linear system

$$v_i(\beta, d) = r_i^k(\beta) + \sum_{j=1}^{N} p_{ij}^k h_{ij}^{gk}(\beta) v_j(\beta, d)$$

where

$$r_i^k(\beta) = \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{\infty} h_{ij}^k(m) \left(\sum_{l=0}^{m-1} \beta^l y_{ij}^k(l) + \beta^m b_{ij}^k(m) \right)$$

and $d(i) = k$ for $i = 1, 2, \ldots, N$.

PIA: Policy Improvement Algorithm

For each state $i \in S$ use $V_\beta(d)$ obtained from the **VDO** to find an action $\tilde{k} \in K_i$ which maximizes

$$r_i^k(\beta) + \sum_{j=1}^{N} p_{ij}^k h_{ij}^{gk}(\beta) v_j(\beta, d)$$

and let $\tilde{d}(i) = \tilde{k}$. If the maximum value exceeds $v_i(\beta, d)$ for any state i, then $\tilde{d}^\infty$ is an improved stationary strategy relative to d^∞.

The policy iteration method alternates the **VDO** and **PIA** steps until no improved stationary strategy is produced.

Remark 2.8 : Implementation of the policy improvement method

The above policy improvement method to find a stationary β-optimal strategy and the corresponding total discounted expected return over an infinite horizon for a discrete time semi-MDP with discounting can be implemented by using the program **dt_SMDP** *which uses the procedures* **PIA_smdp** *and* **VDO_smdp** *with appropriate inputs* (Appendices A.2, B.5 and B.6).

2.3 Discrete Time Semi-Markovian Decision Process without Discounting

2.3.1 Value Iteration Procedure for a Finite Horizon

Definition 2.9 : Total expected return for a finite horizon

In the non-discounting case, let $V^n(\delta_n) = (v_1(n,\delta_n),\ldots,v_N(n,\delta_n))^T$ *be the vector of total expected returns for time horizon* n *that are generated by the process under a policy* $\delta_n = (d_1,\ldots,d_n)$, *where* $d_m = (d_m(1),\ldots,d_m(N))$ *is the decision vector whose components specify the action to be taken in each state when the time remaining is* $l = n+1-m$ *for* $m = 1,\ldots,n$.

If we do not want to discount the system rewards, we simply set β to 1 in our previous analysis for $V_\beta^n(\delta_n)$ defined in Definition 2.4. We then use the same recursive linear system for the total expected non-discounted return vector $V^n(\delta_n)$ for finite time horizon n, namely

$$v_i(n,\delta_n) = \ddot{y}_i^k(n) + v_i(0)\ddot{w}_{ii}^k(n) + r_i^k(n) + \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m) v_j(n-m,\delta_{n-m})$$

where δ_n is the time-dependent policy defined previously and

$$\begin{aligned} \ddot{y}_i^k(n) &= \sum_{j=1}^{N} p_{ij}^k \ddot{h}_{ij}^k(n) \sum_{l=0}^{n-1} y_{ij}^k(l) \\ r_i^k(n) &= \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m) \left(\sum_{l=0}^{m-1} y_{ij}^k(l) + b_{ij}^k(m) \right) \end{aligned}$$

where $\ddot{h}_{ij}^k(n)$ and $\ddot{w}_{ii}^k(n) = \sum_{j=1}^{N} p_{ij}^k \ddot{h}_{ij}^k(n)$ are the same as for the discounting case.

VIP: Value Iteration Procedure

As for the discounting case, we can obtain recursively the maximum total expected return vector V^n and the associated optimal policy δ_n^* for horizon n.

For each $n = 1, 2, \ldots$ obtain

$$v_i(n) = \max_k \left\{ \ddot{y}_i^k(n) + v_i(0)\ddot{w}_{ii}^k(n) + r_i^k(n) + \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{\dot{n}} h_{ij}^k(m) v_j(n-m) \right\}$$

and the action $\tilde{k} \in K_i$ which produces the maximum for $i = 1, \ldots, N$. In this way, we recursively generate V^n and the optimal policy $\delta_n^* = (d_1, \ldots, d_n)$ where d_m is the optimal decision vector for the initial decision when the time remaining is $l = n+1-m$ for $m = 1, \ldots, n$.

Remark 2.10 Implementation of the value iteration procedure

The above value iteration procedure to find a time-dependent optimal policy and the corresponding total expected return over a finite horizon for a discrete time semi-MDP without discounting can also be implemented by using the program **dt_SMDP** *which uses the procedure* **VIP_smdp** *with appropriate inputs* (Appendices A.2 and B.4).

2.3.2 Average Gain for a Completely Ergodic Semi-MDP

Since no discounting factor is present, the total expected return values will usually grow without bound as n increases. Thus we focus on the average gain value $g(d)$ per unit time associated with a stationary strategy d^∞ for a completely ergodic discrete time Semi-MDP.

For a completely ergodic semi-MDP the multi-step Markov transition probabilities converge to a steady distribution regardless of the actions taken, and hence the limiting interval transition probabilities have the form given in Section 2.1. Analogous to Theorem 1.33 for the completely ergodic MDP, there is a version of the approximation equation for the completely ergodic Semi-MDP, namely for stationary strategy d^∞ with $d \in D$ and n sufficiently large

$$V^n(d) \approx ng(d)\mathbf{1} + V(d)$$

where the average gain per unit time $g(d)$ is the expected system-reward for each transition divided by the mean of waiting-time between transitions in the steady state (Howard [4], Vol II).

Since $\ddot{h}_{ij}^k(n) \to 0$ as $n \to \infty$, it follows that $\ddot{y}_i^k(n) \to 0$ and $\ddot{w}_{ii}^k(n) \to 0$ for any action k taken in state i. It then follows that for any stationary strategy d^∞ and sufficiently large n

$$v_i(n,d) \approx r_i^k + \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m) v_j(n-m,d) \quad \textit{for } i = 1,\ldots,N$$

where $r_i^k = \lim_{n\to\infty} r_i^k(n)$ (Howard [4], Vol II). By the approximation equation, it follows that for sufficiently large n

$$\begin{aligned} g(d)n + v_i(d) &\approx r_i^k + \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m)[g(d)(n-m) + v_j(d)] \\ &\approx r_i^k + g(d)n - g(d)\eta_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(d) \end{aligned}$$

where $\eta(d) = (\bar{w}_{11}^k, \ldots, \bar{w}_{NN}^k)^T$ is the vector of mean waiting times for decision vector d. Therefore, as $n \to \infty$, we obtain

$$g(d)\eta_i^k + v_i(d) = r_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(d) \qquad \textit{for all } i \in S$$

which in matrix form is expressed

$$V(d) + g(d)\eta(d) = R(d) + P(d)V(d)$$

Remark 2.11 : Relative value

The components $v_i(d)$ of $V(d)$ are called relative values since the linear system is not affected when the same constant is added to each component $v_i(d)$, $i = 1, 2, \ldots, N$. Hence, we can solve the linear system for all $v_i(d)$ and $g(d)$ for any decision vector d by setting $v_N(d)$ to zero.

Analogous to Theorem 2.7 for a Semi-MDP with discounting, the following theorem provides the basis for the policy iteration method for a discrete time Semi-MDP without discounting.

Theorem 2.12 : Existence of stationary optimal strategy

There exists a stationary optimal strategy for a semi-Markovian decision process with a finite state space S and a finite policy space K.

2.3.3 Policy Iteration Method for an Infinite Horizon

For an infinite time horizon, we have the following policy iteration method consisting of the Value Determination Operation (**VDO**) and the Policy Improvement Algorithm (**PIA**) for a discrete-time semi-MDP without discounting:

VDO: Value Determination Operation

Obtain the average return $g(d)$ per unit time and the relative value vector $V(d)$ for stationary strategy d^∞ where $d \in D$ by solving the linear system

$$g(d)\eta_i^k + v_i(d) = r_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(d)$$

where

$$r_i^k = \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{\infty} h_{ij}^k(m) \left(\sum_{l=0}^{m-1} y_{ij}^k(l) + b_{ij}^k(m) \right)$$

with $v_N(d) = 0$ and $d(i) = k$ for $i = 1, 2, \ldots, N$.

PIA: Policy Improvement Algorithm

For each state $i \in S$ use $g(d)$ and $V(d)$ obtained from the **VDO** to find an action $\tilde{k} \in K_i$ which maximizes

$$\frac{1}{\eta_i^k} \left[r_i^k + \sum_{j=1}^{N-1} p_{ij}^k v_j(d) - v_i(d) \right]$$

and let $\tilde{d}(i) = \tilde{k}$. If the maximum value exceeds $g(d)$ for any state i, then $\tilde{d}^\infty$ is an improved stationary strategy relative to d^∞.

The policy iteration method alternates the **VDO** and **PIA** steps until no improved stationary strategy is produced.

Remark 2.13 : Implementation of the policy improvement method
The above policy improvement method to find a stationary optimal strategy and the corresponding average return per unit time over an infinite horizon for a discrete time completely ergodic semi-MDP without discounting can be implemented by using the program **dt_SMDP** *which uses the procedures* **PIA_smdp** *and* **VDO_smdp** *with appropriate inputs* (Appendices A.2, B.5 and B.6).

2.4 Example: Car Rental Policy

The experience of a car rental company shows that under normal circumstances, when a car is rented in town **1** (town **2**) there is a 0.8 (0.3) probability that it will be returned to town **1** and a 0.2 (0.7) probability that it will be return to town **2**. Moreover there are always customers available in each town so that cars are rented right away at the towns to which they are returned. Due to the nature of the trips involved, the length of time a car will be rented depends both on where it is rented and where it is returned. Therefore, in the discrete time Semi-Markov process to be used for this situation, the holding time corresponds to the length of time a car will be rented when it is rented at town i and returned to town j. It will be assumed,that all car rental times are multiples of some unit time period, say one day, and have a geometric distribution (see Table 2.1).

Let us now examine the reward structure corresponding to the income of the car rental company under this model. We assume that the yields $y_{ij}^k(m)$ are all zero for all $i, j = 1, 2,\ m = 0, 1, 2, \ldots$ since the company receives no income during the

period that a car is rented. However, when a car is rented in town **1** (town **2**), the company charges a rental fee of \$10.00 (\$5.00) per unit time and a drop charge of \$45.00 (\$40.00) if the car is returned in town **2** (town **1**). The company will receive the total payment for each rental when the car is returned.

The company is considering the introduction of an alternative policy that requires each rental car to be returned to the other town, but reduces the drop charge to \$30.00 if a car rented in town **1** is returned to town **2** and eliminates the drop charge completely if a car is returned to town **1**. The distributions of car rental times (holding times) are assumed to be the same for both the original policy and the alternative policy. The assumptions above are summarized in Table 2.1 in which i and k represent the states and the policies respectively (Howard [4], Vol II).

Table 2.1: Data of Car Rental Decision Problem (Discrete-Time Version)

		Transition Probabilities		Car Rental Time (Holding Time) pmf		Daily Charge	Drop Charge	
i	k	p_{i1}^k	p_{i2}^k	$h_{i1}^k(n)$	$h_{i2}^k(n)$	f_i^k	f_{i1}^k	f_{i2}^k
1	1	0.8	0.2	$(1/3)(2/3)^{n-1}$	$(1/6)(5/6)^{n-1}$	10	0	45
	2	0	1	$(1/3)(2/3)^{n-1}$	$(1/6)(5/6)^{n-1}$	10	0	30
2	1	0.3	0.7	$(1/4)(3/4)^{n-1}$	$(1/12)(11/12)^{n-1}$	5	40	0
	2	1	0	$(1/4)(3/4)^{n-1}$	$(1/12)(11/12)^{n-1}$	5	0	0

From the data provided above, we obtain the mean holding times $\bar{h}_{ij}^k$, mean waiting times $\bar{w}_{ii}^k$ and bonus $b_{ij}^k(n) = f_i^k n + f_{ij}^k$ which corresponds to the total payment received when a rental car is returned.

Table 2.2: Means of Holding and Waiting Times and Reward Structure

State i	Policy k	Holding Time $\bar{h}_{i1}^k$	Holding Time $\bar{h}_{i2}^k$	Waiting Time $\bar{w}_{ii}^k$	Total Payment $b_{i1}^k(n)$	Total Payment $b_{i2}^k(n)$
1	1	3	6	3.6	$10n$	$10n$+45
	2	3	6	6	$10n$	$10n$+30
2	1	4	12	9.6	$5n$+40	$5n$
	2	4	12	4	$5n$	$5n$

2.4.1 Optimal Policies over a Finite Horizon

First, let us find the time-dependent policy that maximizes the expected total return generated by the process during a time interval of length n ($n = 1, 2, \ldots$). From the value iteration procedure discussed earlier in this chapter, the maximum total expected return vector V_β^n and the associated β-optimal policy $\delta_n^* = (d_1, \ldots, d_n)$ when using discount factor β can be obtained recursively for $n = 1, 2, \ldots$ using

$$v_i(n, \beta) = \max_k \left\{ \sum_{j=1}^{N} p_{ij}^k \sum_{m=1}^{n} h_{ij}^k(m) \beta^m [b_{ij}^k(m) + v_j(n-m, \beta)] \right\}$$

since it is assumed that all yields $y_{ij}^k(m) = 0$ and the initial return values $v_i(0) = 0$.

The results of this analysis are presented in Table 2.3 for time horizons $n = 1, 2, \ldots, 15$ and discount factors $\beta = 0.2, 0.5(0.1)1.0$. The entries in the body of the table correspond to $d^n(i)$, the optimal initial decision taken in state i, and $v_i(n, \beta)$ which is the maximum expected total return for time horizon n, when the system is in state i ($i = 1, 2$).

Table 2.3: Optimal Policy & Maximum Expected Return for Finite Horizon

		Discount Factor β						
		0.2	0.5	0.6	0.7	0.8	0.9	1.0
n	i	Optimal Decision $d^n(i)$ and Maximum Expected Return $v_i(n,\beta)$						
1	1	**2:** 1.33	**2:** 3.33	**2:** 4.00	**2:** 4.67	**2:** 5.33	**2:** 6.00	**2:** 6.67
	2	**1:** 0.73	**1:** 1.83	**1:** 2.20	**1:** 2.57	**1:** 2.93	**1:** 3.30	**1:** 3.67
2	1	**2:** 1.64	**2:** 5.22	**2:** 6.72	**2:** 8.37	**2:** 10.17	**2:** 12.12	**2:** 14.22
	2	**1:** 0.90	**1:** 2.85	**1:** 3.66	**1:** 4.56	**1:** 5.53	**1:** 6.59	**1:** 7.73
3	1	**2:** 1.70	**2:** 6.24	**2:** 8.48	**2:** 11.16	**2:** 14.33	**2:** 18.05	**2:** 22.35
	2	**1:** 0.93	**1:** 3.40	**1:** 4.62	**1:** 6.08	**1:** 7.80	**1:** 9.82	**1:** 12.16
4	1	**2:** 1.71	**2:** 6.77	**2:** 9.58	**2:** 13.19	**2:** 17.80	**2:** 23.61	**2:** 30.83
	2	**1:** 0.94	**1:** 3.70	**1:** 5.24	**1:** 7.22	**1:** 9.75	**1:** 12.95	**1:** 16.93
5	1	**2:** 1.72	**2:** 7.04	**2:** 10.25	**2:** 14.65	**2:** 20.64	**2:** 28.73	**1:** 39.67
	2	**1:** 0.94	**1:** 3.86	**1:** 5.63	**1:** 8.07	**1:** 11.42	**2:** 16.44	**2:** 23.92
6	1	**2:** 1.72	**2:** 7.18	**2:** 10.66	**2:** 15.68	**2:** 22.93	**1:** 33.46	**1:** 49.57
	2	**1:** 0.94	**1:** 3.94	**1:** 5.88	**1:** 8.07	**2:** 13.13	**2:** 20.70	**2:** 31.97
7	1	**2:** 1.72	**2:** 7.24	**2:** 10.91	**2:** 16.40	**2:** 24.81	**1:** 38.17	**1:** 59.76
	2	**1:** 0.94	**1:** 3.99	**1:** 6.04	**1:** 9.16	**2:** 14.90	**2:** 24.74	**2:** 40.70
8	1	**2:** 1.72	**2:** 7.28	**2:** 11.05	**2:** 16.90	**2:** 26.37	**1:** 42.51	**1:** 70.14
	2	**1:** 0.94	**1:** 4.01	**1:** 6.14	**1:** 9.50	**2:** 16.36	**2:** 28.60	**2:** 49.97
9	1	**2:** 1.72	**2:** 7.30	**2:** 11.13	**2:** 17.24	**2:** 27.66	**1:** 46.47	**1:** 80.68
	2	**1:** 0.94	**1:** 4.02	**1:** 6.20	**1:** 9.74	**2:** 17.56	**2:** 32.23	**2:** 59.64
10	1	**2:** 1.72	**2:** 7.30	**2:** 11.18	**2:** 17.48	**2:** 28.71	**1:** 50.07	**1:** 91.33
	2	**1:** 0.94	**1:** 4.03	**1:** 6.23	**2:** 9.95	**2:** 18.55	**2:** 35.60	**2:** 69.62
11	1	**2:** 1.72	**2:** 7.31	**2:** 11.21	**2:** 17.65	**2:** 29.57	**1:** 53.34	**1:** 102.06
	2	**1:** 0.94	**1:** 4.03	**1:** 6.26	**2:** 10.12	**2:** 19.37	**2:** 38.71	**2:** 79.83
12	1	**2:** 1.72	**2:** 7.31	**2:** 11.23	**2:** 17.77	**2:** 30.26	**1:** 56.30	**1:** 112.86
	2	**1:** 0.94	**1:** 4.03	**1:** 6.27	**2:** 10.25	**2:** 20.03	**2:** 41.56	**2:** 90.23
13	1	**2:** 1.72	**2:** 7.31	**2:** 11.24	**2:** 17.86	**2:** 30.83	**1:** 58.99	**1:** 123.70
	2	**1:** 0.94	**1:** 4.03	**1:** 6.28	**2:** 10.33	**2:** 20.57	**2:** 44.17	**2:** 100.77
14	1	**2:** 1.72	**2:** 7.31	**2:** 11.25	**2:** 17.92	**2:** 31.28	**1:** 61.41	**1:** 134.60
	2	**1:** 0.94	**1:** 4.03	**1:** 6.28	**2:** 10.39	**2:** 21.00	**2:** 46.53	**2:** 111.41
15	1	**2:** 1.72	**2:** 7.31	**2:** 11.25	**2:** 17.97	**2:** 31.64	**1:** 63.60	**1:** 145.53
	2	**1:** 0.94	**1:** 4.03	**1:** 6.29	**2:** 10.43	**2:** 21.36	**2:** 48.68	**2:** 122.14

Examining the results shown in Table 2.3, we can observe the following:

a. For $\beta \leq 0.6$, the optimal policy uses (2,1), namely the alternative car rental policy in town **1** and the normal car rental policy in town, **2** for all time horizons from 1 to 15;

b. For β=0.7, the optimal policy uses (2,1) for time horizons up to 9 but uses (2,2), namely the alternative car rental policy in both towns, for time horizon 10 and beyond;

c. For β=0.8, the optimal policy uses (2,1) for time horizons up to 5 but uses (2,2) for time horizon 6 and beyond;

d. For β=0.9, the optimal policy uses (2,1) for time horizons up to 4 and uses (2,2) for time horizon 5 only, and then uses (1,2), namely the normal car rental policy in town **1** and the alternative car rental policy in town **2**, for time horizons 6 and beyond;

e. For β=1.0 corresponding to the non-discounting case, the optimal policy uses (2,1) for time horizons up to 4 and then uses (2,2) for time horizon 5 and beyond.

From the observations made above we conclude that

a. If the discount factor is small ($\beta \geq 0.9$ or no discounting) and the number of days remaining is at least 6, then a customer who rents a car in town **1** should be free to return it to either town, whereas a customer who rents a car in town **2** should be required to return it to town **1** in order for the company to gain more income.

b. If the discount factor is high ($\beta \leq 0.6$), then the opposite policy should be chosen regardless the time remaining. In this case, town **2** customers should be free to return their rented cars to either town, while town **1** customers should be required to return their cars to town **2** in order to maximize the company's income.

In general, the best policy not only depends upon the discount factor, but also

upon the number of days remaining. The company can use these computational results as a guideline for decision making.

2.4.2 Stationary Strategies over an Infinite Horizon

Now let us turn our attention to the behavior of the total expected return over an infinite horizon. In this case, we find the β-optimal stationary strategy for the company and the corresponding maximum total expected return vector using the policy iteration method (**VDO** and **PIA**). The results are presented in Table 2.4 for discount factors $\beta = 0.2, 0.5(0.1)0.9$. The entries in the body of the table correspond to $d(i)$, the optimal decision when the system is in state i, and $v_i(\beta, d)$, the corresponding maximum expected total return over an infinite horizon.

Table 2.4: Optimal Decision & Maximum Expected Return for Infinite Horizon

	Discount Factor β					
	0.2	0.5	0.6	0.7	0.8	0.9
i	Optimal Decision $d(i)$ and Max. Expected Return $v_i(\beta, d)$					
1	**2:** 1.72	**2:** 7.31	**2:** 11.26	**2:** 18.07	**2:** 33.13	**1:** 83.55
2	**1:** 0.94	**1:** 4.03	**1:** 6.29	**2:** 10.54	**2:** 22.81	**2:** 68.49

If the company does not wish to use a discount factor, then we find the stationary strategy d^∞ that maximizies the expected average future gain value $g(d)$ per unit time (i.e. expected rental income per day from either town) using the policy iteration method (**VDO** and **PIA**).

Using d_0^∞ with $d_0 = (2, 1)$ as an initial stationary strategy, we obtain $g(d_0) = 7.63$ and the relative value $v_1(d_0) = 44.21$ from the **VDO**. A new improved stationary strategy d_1^∞ with $d_1 = (1, 2)$ is then found using **PIA** for which $g(d_1) = 11.14$ and the relative value $v_1(d_1)) = 24.55$. We are not able to obtain another improved

stationary strategy so that we conclude that d_1^∞ is the optimal stationary strategy for the company in the non-discounting case.

Examining the numerical results in Table 2.3, we see that the maximum expected returns $v_i(n, \beta)$ increase monotonically in n for each discount factor β and approach the corresponding maximum expected return for an infinite time period given in Table 2.4. However the rate of convergence decreases as the discount factor increases.

When no discounting is provided ($\beta = 1.0$), then the successive differences $v_i(n) - v_i(n-1)$ of the maximum expected returns increase with n and approach the average gain value g under an optimal stationary strategy for the process over an infinite horizon. This convergence to $g = 11.14$ can be observed in the following table for the car rental decision problem:

Table 2.5: Successive Differences in Maximum Expected Returns

$$v_i(n) - v_i(n-1)$$

State	Horizon n							
	1	2	3	4	5	6	7	8
i	Successive Difference: $v_i(n) - v_i(n-1)$							
1	6.67	7.56	8.13	8.48	8.84	9.90	10.19	10.39
2	3.67	4.06	4.43	4.77	7.00	8.05	8.73	9.26

State	Horizon n						
	9	10	11	12	13	14	15
i	Successive Difference: $v_i(n) - v_i(n-1)$						
1	10.54	10.65	10.73	10.80	10.85	10.89	10.93
2	9.67	9.98	10.22	10.40	10.54	10.65	10.73

Chapter 3

Continuous Time Semi-Markovian Decision Process

With some knowledge of discrete-time MDP and Semi-MDP, we now transfer our attention to the theoretical as well as computational analysis for continuous-time MDP and Semi-MDP in which transitions among the states can occur at arbitrary positive times, not necessarily integer multiples of a unit time. All the models presented in earlier chapters can be adapted to the corresponding process models for continuous-time.

The continuous-time Semi-MDP is like the discrete-time Semi-MDP except for the domain of the holding times. All successive transitions are still governed by the transition probabilities p_{ij}^k of the embedded Markov process associated with the action taken when the system is in state i. However, the holding time h_{ij}^k, which depends on the state i being occupied, the next state j to be visited and the action k taken once the process enters state i, is a continuous random variable that can take on positive values.

The continuous-time MDP is a special case of the continuous-time Semi-MDP with a much simpler structure. In the continuous-time MDP, the distribution of holding time in each state is the same for all successor states and has the property that

the time remaining in the current state does not depend on the time since the state was entered. Such process is an *independent* Semi-MDP in which the holding time distribution for all transitions out of a state are identical and therefore must equal the waiting time distributions for that state. Of course, the waiting time distribution may depend on the current state.

3.1 Continuous Time Semi-Markov Process

In a continuous time semi-Markov process, each transition from a current state to a successor state is governed by the transition probabilities of a Markov process, but the occupancy time in the current state is a real-valued continuous random variable. When the process enters into a state, the successor state is determined by the embedded Markov process and the time of transition is determined by the corresponding continuous holding time distributions.

Definition 3.1 Distributions for a continuous time Markov process

a. *Let* $H(t) = (h_{ij}(t))_{N\times N}$ *be the matrix of holding time densities for* $t \geq 0$.

b. *Let* $W(t) = diag(w_{ii}(t))_{N\times N}$ *and* $W^*(t) = diag(w^*_{ii}(t))_{N\times N}$ *be the matrices whose diagonal entries are the waiting time densities and cumulative waiting time probabilities respectively for* $t \geq 0$

$$w_{ii}(t) = \sum_{j=1}^{N} p_{ij}h_{ij}(t) \quad w^*_{ii}(t) = \int_0^t w_{ii}(\tau)d\tau = \sum_{j=1}^{N} p_{ij}\int_0^t h_{ij}(\tau)d\tau \quad i,j \in S.$$

c. *Let* $\bar{H} = (\bar{h}_{ij})_{N\times N}$ *and* $\bar{W} = diag(\bar{w}_{ii})_{N\times N}$ *be the matrices of the means of the holding time and the waiting time distributions respectively*

$$\bar{h}_{ij} = \int_0^\infty t h_{ij}(t)dt \qquad \bar{w}_{ii} = \int_0^\infty t w_{ii}(t)dt = \sum_{j=1}^{N} p_{ij}\bar{h}_{ij} \quad i,j \in S.$$

d. *Let* $\Phi(t) = (\phi_{ij}(t))_{N\times N}$ *be the matrix whose entries are the interval transition probabilities for the embedded Markov process for* $t \geq 0$

$$\phi_{ij}(t) = \delta_{ij}(1 - w^*_{ii}(t)) + \sum_{k=1}^{N} p_{ik} \int_0^t h_{ik}(\tau)\phi_{kj}(t-\tau)d\tau \quad i,j \in S.$$

Using the symbol $\odot$ for the direct multiplication of matrices, we can express $\Phi(t)$ in matrix form

$$\Phi(t) = (\mathbf{I} - W^*(t)) + \int_0^t (P \odot H(\tau))\Phi(t-\tau)d\tau \quad for\ t \geq 0.$$

The limiting behavior of $\Phi(t)$ is analogous to that of $\Phi(n)$ for the discrete-time case since the embedded Markov chains governing all transitions are the same for both discrete-time and continuous-time semi-Markov processes. In addition, when the embedded Markov chain is ergodic, the limiting interval transition probabilities are independent of the initial state of the process.

For an ergodic Markov process the limiting interval transition probability matrices $\Phi = \lim_{t\to\infty} \Phi(t)$ is given by

$$\phi_{ij} = \frac{\pi_j \bar{w}_{jj}}{\sum_{l=1}^{N} \pi_l \bar{w}_{ll}} = \phi_j \quad for\ all\ i,j \in S$$

where $\pi_i = (\pi_1, \ldots, \pi_N)$ is the steady state probability vector for the embedded ergodic Markov chain. Note that for an ergodic Markov process the rows of Φ are identical (Howard [4], Vol II).

3.2 Continuous Time Semi-Markovian Decision Process with Discounting

Using the same approach as for discrete-time, we now introduce the system reward structure for a semi-MDP with continuous time parameter.

3.2.1 System Return Structure for a Finite Horizon

Definition 3.2 : Yield-rate and bonus

a. *Let $y_{ij}^k(t)$ be the yield rate at time t during the occupancy of state i when action k is chosen upon the entry into state i and where a successor state j has been determined according to the transition probabilities p_{ij}^k. Thus yield is earned continuously during the occupancy of state i.*

b. *Let $b_{ij}^k(t)$ be the bonus earned when the transition from state i to state j is actually made at time t. In contrast to the yield rate, the bonus is a fixed amount.*

Definition 3.3 Discounted accumulated yield

For a discount rate α, the discounted accumulated yield $y_{ij}^k(t,\alpha)$ generated from the yield rate function over a time interval (0,t) is

$$y_{ij}^k(t,\alpha) = \int_0^t e^{-\alpha\tau} y_{ij}^k(\tau) d\tau \quad 0 \leq t < \infty.$$

We assume that all yield rate functions grow no faster than exponential rate α in order that the discounted accumulated yield be finite (Howard [4], Vol II).

Definition 3.4 Total discounted expected return for a stationary strategy

Let $V_\alpha(t,d) = (v_1(t,\alpha,d), v_2(t,\alpha,d), \ldots, v_N(t,\alpha,d))^T$ be the total discounted expected returns over a time interval of length t, associated with a stationary strategy d^∞. Further, let $V(0) = (v_1(0), \ldots, v_N(0))^T$ be the terminal reward vector.

The total discounted expected return generated by a continuous-time semi-MDP over a finite time interval, when using stationary strategy d^∞ can be represented as described below (Howard [4], Vol II):

$$
\begin{aligned}
& v_i(t,\alpha,d) \\
&= \sum_{j=1}^{N} p_{ij}^k \int_t^\infty h_{ij}^k(\tau)d\tau \left(\int_0^t e^{-\alpha\sigma} y_{ij}^k(\sigma)d\sigma + e^{-\alpha t} v_i(0) \right) \\
&\quad + \sum_{j=1}^{N} p_{ij}^k \int_0^t h_{ij}^k(\tau) \left(\int_0^\tau e^{-\alpha\sigma} y_{ij}^k(\sigma)d\sigma + e^{-\alpha\tau} b_{ij}^k(\tau) + e^{-\alpha\tau} v_j(t-\tau,\alpha,d) \right) d\tau \\
&= \ddot{y}_i^k(t,\alpha) + e^{-\alpha t} v_i(0) \ddot{w}_{ii}^k(t) + r_i^k(t,\alpha) + \sum_{j=1}^{N} p_{ij}^k \int_0^t e^{-\alpha\tau} h_{ij}^k(\tau) v_j(t-\tau,\alpha,d)d\tau
\end{aligned}
$$

for $i = 1, 2, \ldots, N$ *and* $t > 0$, where

a:

$$
\begin{aligned}
\ddot{y}_i^k(t,\alpha) &= \sum_{j=1}^{N} p_{ij}^k \int_t^\infty h_{ij}^k(\tau)d\tau \int_0^t e^{-\alpha\sigma} y_{ij}^k(\sigma)d\sigma \\
&= \sum_{j=1}^{N} p_{ij}^k \ddot{h}_{ij}^k(t) y_{ij}^k(t,\alpha)
\end{aligned}
$$

is the contribution to the total expected return from the discounted accumulated yields $y_{ij}^k(t,\alpha)$ for chosen action k when the first transition out of state i occurs after time t, where $\ddot{h}_{ij}^k(t) = \int_t^\infty h_{ij}^k(\tau)d\tau$.

b:

$$
\ddot{w}_{ii}^k(t) = \int_t^\infty w_{ii}^k(\tau)d\tau = \sum_{j=1}^{N} p_{ij}^k \ddot{h}_{ij}^k(t)
$$

and $e^{-\alpha t} v_i(0)$ multiplied by $\ddot{w}_{ii}^k(t)$ is the contribution of the terminal reward to the total expected return once the holding time has expired.

c:

$$
\begin{aligned}
r_i^k(t,\alpha) &= \sum_{j=1}^{N} p_{ij}^k \int_0^t h_{ij}^k(\tau) \left(\int_0^\tau e^{-\alpha\sigma} y_{ij}^k(\sigma)d\sigma + e^{-\alpha\tau} b_{ij}^k(\tau) \right) d\tau \\
&= \sum_{j=1}^{N} p_{ij}^k \int_0^t h_{ij}^k(\tau) \left(y_{ij}^k(\tau,\alpha) + e^{-\alpha\tau} b_{ij}^k(\tau) \right) d\tau
\end{aligned}
$$

is the contribution of both the accumulated yield and the bonus from the first transition when it occurs before time t.

d:

$$\sum_{j=1}^{N} p_{ij}^k \int_0^t e^{-\alpha\tau} h_{ij}^k(\tau) v_j(t-\tau, \alpha, d) d\tau$$

is the total expected return from future transitions when the first transition occurs before time t.

3.2.2 Policy Iteration Method for an Infinite Horizon

Lemma 3.5 : Total discounted expected return for infinite horizon

For each stationary strategy d^∞ where $d \in D$, the vector $V_\alpha(d)$ of total discounted expected returns over an infinite horizon satisfies the following linear system (Howard [4], Vol II)*:*

$$v_i(\alpha, d) = r_i^k(\alpha) + \sum_{j=1}^{N} p_{ij}^k h_{ij}^{ek}(\alpha) v_j(\alpha, d)$$

where

$$v_i(\alpha, d) = \lim_{t\to\infty} v_i(t, \alpha, d) \quad r_i^k(\alpha) = \lim_{t\to\infty} r_i^k(t, \alpha) \quad h_{ij}^{ek}(s) = \int_0^\infty e^{-st} h_{ij}^k(t) dt$$

for all $i, j \in S$. Note that $h_{ij}^e(s)$ is the Laplace transform of holding-time distribution $h_{ij}(t)$. Hence $V_\alpha(d)$ can be solved in matrix form

$$V_\alpha(d) = [\mathbf{I} - P(d) \odot H^e(\alpha, d)]^{-1} R_\alpha(d)$$

The lemma above follows from the fact that $\ddot{h}_{ij}^k(t) \to 0$ and $e^{-\alpha t} \to 0$ when $t \to \infty$ so that $\ddot{y}_i^k(t, \alpha) \to 0$ and $e^{-\alpha t} v_i(0) \ddot{w}_i^k(t) \to 0$.

Definition 3.6 : α-optimal strategy

A strategy δ^* is said to be α-optimal if $V_\alpha(\delta^*) \geq V_\alpha(\delta)$ for all strategies δ, where α ($\alpha > 0$) is a discount rate ($e^{-\alpha t}$ corresponds to the discount factor β for the discrete time case).

As for the discrete-time case with discounting, we have the following result on the existence of an α-optimal stationary strategy.

Theorem 3.7 : Existence of α-optimal stationary strategy

There exists a stationary α-optimal strategy for a continuous-time semi-Markovian decision process with finite state space and finite policy space (Mine & Osaki [1]).

For an infinite time horizon, we have the following policy iteration method consisting of the Value Determination Operation (**VDO**) and the Policy Improvement Algorithm (**PIA**) for continuous-time semi-MDP with discounting:

VDO: Value Determination Operation

Obtain the total discounted expected return vector $V_\alpha(d)$ for stationary strategy d^∞ where $d \in D$ by solving the linear system

$$v_i(\alpha, d) = r_i^k(\alpha) + \sum_{j=1}^{N} p_{ij}^k h_{ij}^{ek}(\alpha) v_j(\alpha, d)$$

where

$$r_i^k(\alpha) = \sum_{j=1}^{N} p_{ij}^k \int_0^\infty h_{ij}^k(\tau) \left(\int_0^\tau e^{-\alpha\sigma} y_{ij}^k(\sigma) + e^{-\alpha\tau} b_{ij}^k(\tau) \right) d\tau$$

and $d(i) = k$ for $i = 1, 2, \ldots, N$.

PIA: Policy Improvement Algorithm

For each state $i \in S$ use $V_\alpha(d)$ obtained from the **VDO** to find an action $\tilde{k} \in K_i$ which maximizes

$$r_i^k(\alpha) + \sum_{j=1}^{N} p_{ij}^k h_{ij}^{ek}(\alpha) v_j(\alpha, d)$$

and let $\tilde{d}(i) = \tilde{k}$. If the maximum value exceeds $v_i(\alpha, d)$ for any state i, then $\tilde{d}^\infty$ is an improved stationary strategy relative to d^∞.

The policy iteration method alternates the **VDO** and **PIA** steps until no improved stationary strategy is produced.

Remark 3.8 : Implementation of the policy improvement method

The above policy improvement method to find a stationary α-optimal strategy and the corresponding total discounted expected return over an infinite horizon for a continuous time semi-MDP with discounting can be implemented by using the program **ct_SMDP** *which uses the procedures* **PIA_smdp** *and* **VDO_smdp** *with appropriate inputs* (Appendices A.3, B.5 and B.6).

3.3 Continuous Time Semi-Markovian Decision Process without Discounting

Similar to the discounting case, let $V(t,d) = (v_1(t,d), \ldots, v_N(t,d))^T$ be the vector of the total expected returns over a time interval of length t for a stationary strategy d^∞ in the non-discounting case. It has the following representation:

$$v_i(t,d) = \ddot{y}_i^k(t) + v_i(0)\ddot{w}_{ii}^k(t) + r_i^k(t) + \sum_{j=1}^{N} p_{ij}^k \int_0^t h_{ij}^k(\tau) v_j(t-\tau,d)d\tau$$

for $i = 1, 2, \ldots, N$ and $t > 0$, where

$$\begin{aligned} \ddot{y}_i^k(t) &= \sum_{j=1}^{N} p_{ij}^k \ddot{h}_{ij}^k(t) y_{ij}^k(t) \\ r_i^k(t) &= \sum_{j=1}^{N} p_{ij}^k \int_0^t h_{ij}^k(\tau) \left(y_{ij}^k(\tau) + b_{ij}^k(\tau) \right) d\tau \end{aligned}$$

with $y_{ij}^k(\tau) = \int_0^\tau y_{ij}^k(\sigma)d\sigma$ for all $i, j \in S$.

3.3.1 Average Gain for a Completely Ergodic Semi-MDP

As for the discrete time semi-MDP without discounting, the choice of a stationary strategy d^∞ for an infinite horizon in the continuous time case will be based on the average gain per unit time $g(d)$ rather than the total expected return values $V(t,d)$

since the latter will usually grow without bound as time t increases. If the process is completely ergodic, then analogous to the discrete time case, we have the following approximation for t sufficiently large

$$V(t,d) \approx tg(d)\mathbf{1} + V(d)$$

where the average gain per unit time $g(d)$ is the expected system-reward for each transition divided by the mean of waiting-time between transitions in the steady state (Howard [4], Vol II).

Since $\ddot{h}_{ij}^k(t) \to 0$ when $t \to \infty$, $\ddot{y}_i^k(t,0) \to 0$ and $\ddot{w}_{ii}^k(t) \to 0$ for any action k taken in state i. It then follows that for any stationary strategy d^∞ and sufficiently large t

$$v_i(t,d) \approx r_i^k + \sum_{j=1}^{N} p_{ij}^k \int_0^t h_{ij}^k(\tau) v_j(t-\tau,d)d\tau \quad for\ all\ i \in S$$

where $r_i^k = \lim_{t\to\infty} r_i^k(t)$. Using the approximation equation, it follows that for sufficiently large t

$$\begin{aligned} g(d)t + v_i(d) &\approx r_i^k + \sum_{j=1}^{N} p_{ij}^k \int_0^\infty h_{ij}^k(\tau)[g(d)(t-\tau) + v_j(d)]d\tau \\ &\approx r_i^k + g(d)t - g(d)\eta_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(d) \end{aligned}$$

where $\eta(d) = (\bar{w}_{11}^k, \ldots, \bar{w}_{NN}^k)^T$ is the vector of mean waiting times associated with the decision vector d. Therefore, as $t \to \infty$, we obtain

$$g(d)\eta_i^k + v_i(d) = r_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(d) \quad for\ all\ i \in S$$

Theorem 3.9 : Existence of stationary optimal strategy

There exists a stationary optimal strategy for a continuous time Semi-Markovian decision process with finite state space and finite policy space (Mine & Osaki [1]).

Notice that the set of linear equations for the average gain and the relative values are exactly the same as for discrete-time completely ergodic semi-MDP over an infinite horizon. Therefore the policy iteration procedure for continuous-time semi-MDP will not be presented here.

Remark 3.10 : Implementation of the policy improvement method

The above policy improvement method to find a stationary optimal strategy and the corresponding average return per unit time over an infinite horizon for a continuous time and completely ergodic semi-MDP without discounting can be implemented by using the program **ct_SMDP** *which uses the procedures* **PIA_smdp** *and* **VDO_smdp** *with appropriate inputs* (Appendices A.3, B.5 and B.6).

3.3.2 Example: Car Rental Problem Revisited

The car rental example, which was introduced in the previous chapter, is presented here again with the same set of actions, transition probabilities, yield-rate and bonus structure but where the holding times are assumed to be exponentially distributed. This corresponds to the assumption that the rental locations in both towns have 24-hour check-in so that the duration of a rental can be measured on a continuous time scale. For numerical convenience, we consider our time unit to be 12 hours which corresponds to 12 of the unit time periods used in the discrete-time version (Howard [4], Vol II).

All the input data is given in the Table 3.1 where $t \geq 0$ and i and k represent the states and the policies respectively. The mean holding-times $\bar{h}_{ij}^k$, mean waiting-times $\bar{w}_{ij}^k$ and bonus $b_{ij}^k(t) = f_i^k t + f_{ij}^k$ are presented in Table 3.2.

Table 3.1: Data of Car Rental Decision Problem (Continuous-Time Version)

i	k	Transition Probabilities p_{i1}^k	p_{i2}^k	Car Rental Time (Holding Time) pdf $h_{i1}^k(t)$	$h_{i2}^k(t)$	Daily Charge f_i^k	Drop Charge f_{i1}^k	f_{i2}^k
1	1	0.8	0.2	$4e^{-4t}$	$2e^{-2t}$	10	0	45
	2	0	1	$4e^{-4t}$	$2e^{-2t}$	10	0	30
2	1	0.3	0.7	$3e^{-3t}$	e^{-t}	5	40	0
	2	1	0	$3e^{-3t}$	e^{-t}	5	0	0

Table 3.2: Means of Holding and Waiting Times and Reward Structure

State i	Policy k	Holding Time $\bar{h}_{i1}^k$	$\bar{h}_{i1}^k$	Waiting Time $\bar{w}_{ii}^k$	Total Payment $b_{i1}^k(t)$	$b_{i2}^k(t)$
1	1	0.25	0.50	0.30	$10t$	$10t$+45
	2	0.25	0.50	0.50	$10t$	$10t$+30
2	1	$0.\overline{33}$	1.00	0.80	$5t$+40	$5t$
	2	$0.\overline{33}$	1.00	$0.\overline{33}$	$5t$	$5t$

First consider the discounting case. The α-optimal stationary strategies obtained by using the policy iteration method for the continuous-time semi-MDP are presented in Table 3.3 for discount rates $\alpha = 0.1, 0.2, 0.3, 0.5, 0.7, 0.8, 0.9$.

The entries in the body of the table correspond to $d(i)$, the optimal decision when the system is in state i, and $v_i(d)$, the corresponding maximum expected return over an infinite horizon.

If no discounting is used, then we find the stationary strategy d^∞ that maximizes the average gain per unit time (i.e. expected rental income per day from each town) using the policy iteration method. Using d_0^∞ as an initial stationary strategy where $d_0 = (2, 1)$ in accordance with the limiting values of the non-discounted system re-

Table 3.3: Optimal Decision and Maximum Expected Return for Infinite Horizon

	Discount Rate α						
	0.1	0.2	0.3	0.5	0.7	0.8	0.9
i	Optimal Decision $d(i)$ and Maximum Expected Return v_i						
1	**2:** 441.57	**2:** 221.60	**2:** 148.29	**2:** 89.66	**2:** 64.54	**2:** 56.69	**2:** 50.58
2	**2:** 428.89	**2:** 209.22	**2:** 136.19	**2:** 78.08	**2:** 53.43	**2:** 45.79	**2:** 39.90

wards ($r_1^1 = 12.00,\ r_1^2 = 35.00,\ r_2^1 = 16.00,\ r_2^2 = 1.67$), we then obtain $g(d_0) = 27.89$ and the relative value $v_1(d_0) = 21.05$ from the **VDO**. A new improved stationary strategy d_1^∞ where $d_1 = (2, 2)$ is then found through the **PIA** for which $g(d_1) = 44.00$ and the relative value vector $v_1(d_1) = 13.00$. No further improved stationary strategy can be found and hence d_2^∞ is an optimal stationary strategy. Compared with the result for the discrete-time case, there is a different optimal policy, namely that customers should be allowed to return their rented cars to either town regardless of where they are rented.

3.4 Continuous Time Markovian Decision Process

As mentioned at the beginning of this chapter, continuous-time Markov processes are a special case of continuous-time semi-Markov processes. We now discuss such processes and the determination of optimal strategies.

3.4.1 Continuous Time Markov Process

The continuous-time Markov process has a simpler theoretical structure than the continuous-time semi-Markov process. In a continuous-time Markov process, the

time remaining in the current state does not depend on the length of time that the state has been occupied and is also irrelevant to determining the next state to be entered. Furthermore, the holding time distribution for each state is the same for all successor states and hence must equal the waiting time distribution for that state.

Lemma 3.11 Exponential waiting-times for continuous time Markov process

For a continuous-time Markov process, the time between transitions must be exponentially distributed, that is with pdf $w_{ii}(t) = \lambda_i e^{-\lambda_i t}$ *for* $t \geq 0$ *and mean* $\bar{w}_{ii} = 1/\lambda_i$ *where* $\lambda_i > 0$ *for* $i = 1, 2, \ldots, N$ (Howard [4], Vol II).

Definition 3.12 Transition rates of continuous time Markov process

Let $A = \Lambda(P - \mathbf{I})$ *be the matrix of transition-rates for a continuous-time Markov process where* $\Lambda = diag(\lambda_1, \ldots, \lambda_N)$ *i.e.*

$$a_{ii} = -\lambda_i \sum_{j \neq i} p_{ij} \text{ for all } i \in S; \quad a_{ij} = \lambda_i p_{ij} \text{ for all } i \neq j$$

The rows of this transition-rate matrix sum to zero rather than 1. It plays a central role for continuous-time Markov processes as does the transition probability matrix for discrete-time Markov processes (Howard [4], Vol II).

Lemma 3.13 Interval transition probabilities and state probabilities

The interval transition probability matrix $\Phi(t)$ *as well as the state probability vector* $\pi(t)$ *for a continuous-time Markov process are exponential functions of the transition-rate matrix A of the process, namely*

$$\Phi(t) = \mathbf{e}^{At} \quad \text{and } \pi(t) = \pi(0)\Phi(t) = \pi(0)\mathbf{e}^{At} \quad t \geq 0$$

where $\Phi(0) = \mathbf{I}$, *and* $\pi(0)$ *is the initial state probability vector and the matrix function* $\mathbf{e}^{At}$ *is defined by*

$$\mathbf{e}^{At} = \sum_{n=0}^{\infty} \frac{A^n t^n}{n!} \qquad for\ t \geq 0.$$

The above relationship corresponds to $\Phi(n) = \mathbf{P}^n$ *for* $n = 1, 2, \ldots$ *in the discrete-time case* (Howard [4], Vol II).

3.4.2 Continuous Time MDP with Discounting

Consider a continuous-time Markovian process with transition-rate matrix $A(d) = (a_{ij}^k)_{N\times N}$ and the system reward rate matrix $R(d) = (r_{ij}^k)_{N\times N}$ associated with decision vector $d \in D$. For a stationary strategy d^∞, let $V_\alpha(t, d) = (v_1(t, \alpha, d), \ldots, v_N(t, \alpha, d))^T$ be the vector of discounted total expected returns earned over a time interval of length of t.

Suppose the system earns a reward $r_{ii}^k dt$ when action k is taken upon entry into state i if it remains in the state i for time dt, plus the expected return $v_i(t, \alpha, d)$ that it will earn in the remaining time t. On the other hand, the system earns a reward $r_{ij}^k dt$ for a transition from state i to state j during time dt, plus the expected return $v_j(t, \alpha, d)$ when it enters state j with time t remaining. Each of the above components is subject to a discount factor $(1 - \alpha dt)$ when the discount rate is α. These assumptions lead to the following expression for $v_i(t + dt, \alpha, d)$ (Howard [3]).

$$\begin{aligned} & v_i(t + dt, \alpha, d) \\ &= (1 - \alpha dt)\left\{\left(1 - \sum_{i\neq j} a_{ij}^k dt\right)[r_{ii}^k dt + v_i(t, \alpha, d)] + \sum_{i\neq j} a_{ij}^k dt[r_{ij}^k + v_j(t, \alpha, d)]\right\} \\ &= q_i^k dt + v_i(t, \alpha, d) + \sum_{j=1}^{N} a_{ij}^k v_j(t, \alpha, d)dt - \alpha v_i(t, \alpha, d)dt + o(dt) \end{aligned}$$

where $q_i = r_{ii} + \sum_{i\neq j} a_{ij} r_{ij}$ is defined to be the system earning rates associated with

state i. Thus

$$v_i(t+dt,\alpha,d) - v_i(t,\alpha,d) = \left(q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(t,\alpha,d) - \alpha v_i(t,\alpha,d) \right) dt + o(dt)$$

for each $i = 1, 2, \ldots, N$ and $t \geq 0$. Forming the difference quotient and letting $dt \to 0$, we obtain the following:

$$\frac{dv_i(t,\alpha,d)}{dt} = q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(t,\alpha,d) - \alpha v_i(t,\alpha,d) \quad for\ t \geq 0$$

It is obvious that

$$\lim_{t\to\infty} \frac{dv_i(t,\alpha,d)}{dt} = 0$$

since $v_i(t,\alpha,d)$ is bounded and non-decreasing in t for all $i \in S$. If we let $v_i(\alpha,d) = \lim_{t\to 0} v_i(t,\alpha,d)$, then we obtain the following linear system for $v_i(\alpha,d)$.

$$\alpha v_i(\alpha,d) = q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(\alpha,d) \quad i = 1, 2, \ldots, N.$$

Analogous to Theorem 3.7 for a continuous time semi-MDP with discounting, the following theorem provides the basis for the policy iteration method for a continuous time MDP with discounting.

Theorem 3.14 : Existence of α-optimal stationary strategy

There exists a stationary α-optimal strategy for a continuous-time, Markovian decision process, with a finite state space and a finite policy space.

For an infinite time horizon, we have the following policy iteration method consisting of the Value Determination Operation (**VDO**) and the Policy Improvement Algorithm (**PIA**) for continuous-time MDP with discounting:

VDO: Value Determination Algorithm

Obtain the total discounted expected return vector $V_\alpha(d)$ for stationary strategy d^∞ where $d \in D$ by solving the linear system

$$\alpha v_i(\alpha, d) = q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(\alpha, d)$$

where $q_i = r_{ii} + \sum_{i \neq j} a_{ij} r_{ij}$ and $d(i) = k$ for $i = 1, 2, \ldots, N$.

PIA: Policy Improvement Algorithm

For each state $i \in S$ use $V_\alpha(d)$ obtained from the **VDO** to find an action $\tilde{k} \in K_i$ which maximizes

$$\frac{1}{\alpha}\left[q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(\alpha, d)\right]$$

and let $\tilde{d}(i) = \tilde{k}$. If the maximum value exceeds $v_i(\alpha, d)$ for any state i, then $\tilde{d}^\infty$ is an improved stationary strategy relative to d^∞.

The policy iteration method alternates the **VDO** and **PIA** steps until no improved stationary strategy is produced.

Remark 3.15 : Implementation of the policy improvement method

The above policy improvement method to find a stationary α-optimal strategy and the corresponding total discounted expected return over an infinite horizon for a continuous time MDP with discounting can be implemented by using the program **ct_MDP** *which uses the procedures* **PIA_mdp** *and* **VDO_mdp** *with appropriate inputs* (Appendices A.4, B.2 and B.3).

3.4.3 Continuous Time MDP without Discounting

Let $V(t, d) = (v_1(t, d), \ldots, v_N(t, d))^T$ be the vector of non-discounted total expected return earned over a time interval of length of t. Analogous to the discounting case,

one obtains the following expression for $v_i(t+dt,d)$ (Howard [4], Vol II):

$$v_i(t+dt,d) = q_i^k dt + v_i(t,d) + \sum_{j=1}^{N} a_{ij}^k v_j(t,d)dt \quad i = 1,2,\ldots,N \text{ and } t \geq 0$$

Hence

$$v_i(t+dt,d) - v_i(t,d) = \left(q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(t,d) \right) dt + o(dt)$$

for $i = 1,2,\ldots,N$ and $t \geq 0$. Forming the difference quotient and letting $dt \to 0$, we then obtain the following linear system

$$\frac{dv_i(t,d)}{dt} = q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(t,d) \quad for\ t \geq 0$$

For a completely ergodic MDP the total expected return up to time t can be approximated by the following for a stationary strategy d^∞ for t sufficiently large

$$V(t,d) \approx tg(d)\mathbf{1} + V(d)$$

where $g(d)$ is the average gain per unit time and $v_i(d)$ is the relative value associated with the stationary strategy d^∞. Using this approximation and letting $v_i(d) = \lim_{t\to 0} v_i(t,d)$, we obtain the following linear system for $v_i(d)$ for t sufficiently large (Howard [3]):

$$g(d) = q_i^k + g(d)t\sum_{j=1}^{N} a_{ij}^k + \sum_{j=1}^{N} a_{ij}^k v_j(d) = q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(d)$$

Theorem 3.16 : Existence of stationary optimal strategy

There exists a stationary optimal strategy for a continuous time Markovian decision process with finite state space and finite policy space.

For an infinite time horizon, we have the following policy iteration method consisting of the Value Determination Operation (**VDO**) and the Policy Improvement

Algorithm (**PIA**) for a continuous-time MDP without discounting:

VDO: Value Determination Algorithm

Obtain the average return $g(d)$ per unit time and the relative return vector $V(d)$ for stationary strategy d^∞ by setting $v_N(d) = 0$ and then solving the linear system

$$g(d) = q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(d) \quad i = 1, 2, \ldots, N.$$

where $v_i(d)$ is the i^{th} component of $V(d)$ and $d(i) = k$.

PIA: Policy Improvement Algorithm

For each state $i \in S$, use $g(d)$ and $V(d)$ obtained from the **VDO** to find an action $\tilde{k} \in K_i$ which maximizes

$$q_i^k + \sum_{j=1}^{N} a_{ij}^k v_j(d)$$

and let $\tilde{d}(i) = \tilde{k}$. If the maximum value exceeds $g(d)$ for any state i, then $\tilde{d}^\infty$ is an improved stationary strategy relative to d^∞.

The policy iteration method alternates the **VDO** and **PIA** steps until no improved stationary strategy is produced.

Remark 3.17 : Implementation of the policy improvement method

The above policy improvement method to find a stationary optimal strategy and the corresponding average return per unit time over an infinite horizon for a continuous time MDP without discounting can be implemented by using the program **ct_MDP** *which uses the procedures* **PIA_mdp** *and* **VDO_mdp** *with appropriate inputs* (Appendix A [4]; Appendix B: [2] & [3]).

3.4.4 Example: Another Machine Maintenance Problem

Let us recast the machine maintenance example with a different set of data but the same set of policies:

For state 1: k=1 represents no intervention; k=2 represents preventive maintenance.
For state 2: k=1 represents normal repair; k=2 represents extended repair.

Table 3.4: Data of Machine Maintenance Problem

State i	Action k	Probability p_{i1}^k	Probability p_{i2}^k	Reward Rate r_{i1}^k	Reward Rate r_{i2}^k
1	1	0.5	0.5	6.0	0.0
	2	0.8	0.2	4.0	0.0
2	1	0.4	0.6	-0.5	-1.0
	2	0.7	0.3	-0.5	-1.5

Assume that the time for each transition follows an exponential distribution with mean 10 regardless of the actions taken in each state i.e. $w_{ii} = e^{-0.1t}$ for $t > 0$. Computing the corresponding transition-rate matrix A and the earning-rate vector Q associated with the different policies, we have the following table:

Table 3.5: Transition Rate Matrix A & Earning Rate Vector Q

State i	Action k	Transition-rates a_{i1}^k	Transition-rates a_{i2}^k	Earning-rates q_i^k
1	1	-5	5	6
	2	-2	2	4
2	1	4	-4	-3
	2	7	-7	-5

a. Discounting Case

Let us first consider discounting case using $\alpha = 1/9$. As is the custom, we choose a initial stationary strategy d_0^∞ where $d_0 = (1, 1)$ that maximizes the earning rate. Then we obtain the following results from **VDO** and **PIA** iteration procedure:

Step 1 (**VDO**):

$$\frac{1}{9}V_\alpha(d_0) = Q(d_0) + A(d_0)V_\alpha(d_0) \Longrightarrow V_\alpha(d_0) = (9.55, 8.56)^T$$

Step 2 (**PIA**):

Comparing the following test quantities

$$q_1^1 + \sum_{j=1}^{2} a_{1j}^1 v_j(\alpha, d_0) = 1.06, \quad q_1^2 + \sum_{j=1}^{2} a_{1j}^2 v_j(\alpha, d_0) = 2.02$$

$$q_2^1 + \sum_{j=1}^{2} a_{2j}^1 v_j(\alpha, d_0) = 0.95, \quad q_2^2 + \sum_{j=1}^{2} a_{2j}^2 v_j(\alpha, d_0) = 1.91$$

Hence the first improved stationary strategy d_1^∞ is obtained, where $d_1 = (2, 2)$.

Step 3 (**VDO**):

$$\frac{1}{9}V_\alpha(d_1) = Q(d_1) + A(d_1)V_\alpha(d_1) \Longrightarrow V_\alpha(d_1) = (18.22; 17.23)^T$$

Step 4 (**PIA**):

$$q_1^1 + \sum_{j=1}^{2} a_{1j}^1 v_j(\alpha, d_1) = 1.06, \quad q_1^2 + \sum_{j=1}^{2} a_{1j}^2 v_j(\alpha, d_1) = 2.02$$

$$q_2^1 + \sum_{j=1}^{2} a_{2j}^1 v_j(\alpha, d_1) = 0.95, \quad q_2^2 + \sum_{j=1}^{2} a_{2j}^2 v_j(\alpha, d_1) = 1.91$$

No further improved stationary strategy can be obtained, so that d_1^∞ with $d_1 = (2, 2)$ is a stationary α optimal strategy in the discounting case. This suggests that the rapid repair would be the best chose in a long run regardless of the machine's condition.

b. Non-discounting Case

For the initial stationary strategy d_0^∞, we set $v_2(d_0) = 0$ and then solve

$$g(d_0) = Q(d_0) + A(d_0)V(d_0) \Longrightarrow V(d_0) = (1.00, 0)^T \ \ and \ g(d_0) = 1.00$$

The same approach leads to an improved stationary strategy d_1^∞, with $d_1 = (2, 2)$. For this improved stationary strategy, we set $v_2(d_1) = 0$ and solve

$$g(d_1) = Q(d_1) + A(d_1)V(d_1) \Longrightarrow V(d_1) = (1.00, 0)^T \quad and \; g(d_0) = 2.00$$

No further improved stationary strategy can be found, and hence d_1^∞ is the optimal stationary strategy in the non-discounting case, which yields the same suggestion for this machine maintenance problem in the discounting case.

Chapter 4

Summary and Conclusion

In discussions of Markovian or semi-Markovian decision processes we have presented the optimization techniques appropriate for a variety of different cases: discrete-time or continuous-time, finite or infinite horizon, discounting or non-discounting, completely ergodic or terminating. In the case of an infinite horizon, the appropriate policy iteration methods involving the value determination operation (**VDO**) and the policy improvement algorithm (**PIA**) can be used to find a stationary optimal strategy which maximizes either the total discounted expected return in the discounting case or the average gain per unit time in the non-discounting case. In order to obtain a strategy that maximizes total expected return, either discounted or non-discounted, for a finite number of discrete-time stages, one can use the value iteration method (**VIP**) for a finite horizon.

For each type of decision process over an infinite horizon, the existence of the corresponding stationary optimal strategy is noted in theorems presented throughout the thesis. All variations on the policy improvement method for infinite horizon and the value iteration procedures for finite horizon are formulated in the thesis and implemented in the **Maple** programs and procedures given in the appendices to the thesis.

In the discounting case, the discount factor does not change the basic nature of the decision-making problem. However, the time-dependent optimal policy for a finite horizon and the stationary optimal strategy for an infinite horizon is affected by changes in the discount factor changes, as was illustrated in the car rental example.

In the non-discounting case, we know that the total expected return values usually grow without bound as a process continues for an indefinite duration. Thus the average return or gain per unit time generated by the system in the long run is used as the criterion. Moreover, we have for the most part restricted our attention to the case of completely ergodic decision processes i.e. the Markov chain regulating the transition probabilities is ergodic regardless of the action taken in each state at any time in the process. In this case, we have the approximation equations for the total expected return values for sufficiently large time. The different approximation equations lead to the appropriate linear system for the policy iteration method to obtain the average gain over an infinite horizon.

For a terminating decision process, the terminating assumption is made so that the common absorbing state can be reached with probability 1 from any transient state in a finite number of transitions, regardless of the decision made in each transient state. We then consider the total expected return generated by the system before the process enters the absorbing state and terminates. The policy iteration method for a terminating process is similar to that for a non-terminating decision process in the discounting case.

As we have seen, the analysis of optimization in terms of policy iteration method for discrete-time and continuous-time semi-MDP is very similar. We have a similar computational structure in the policy iteration method except that we need to use either the geometric transform of the holding-time distribution in the discrete-time

case or the Laplace transform of the holding-time distribution in the continuous-time case in order to implement the policy iteration method. For a semi-MDP without discounting, the mean of the waiting time distribution is needed for the implementation of the policy iteration method.

In the continuous time case, we have a more complex situation since transitions occur on a continuous time scale and hence a strategy is a function defined over an interval of time. If we are interested in the total expected return for a finite horizon, then we have the option of approximating the continuous-time process by a discrete-time process, which has the same transition probabilities, action spaces, initial system reward values, and then using an approximation to the holding-time distribution for a sequence of discrete times over the horizon specified. On the other hand, in the case of an infinite horizon the policy iteration method is applicable as it was in the discrete-time case.

A continuous-time MDP is a special case of a continuous-time semi-MDP in which the times between transitions are exponentially distributed and independent of the successor states. The transition-rate matrix, which is determined from the Markov transition probabilities and the mean waiting-times, is used in the policy improvement method for both the discounting and non-discounting case.

For a decision process in which the embedded Markov chain has more than one communicating class, the system reward structure is usually often more complicated. However, if a discount factor is used, then the policy iteration method may be adapted to obtain the maximum total discounted expected return.

An investigation of Markovian or semi-Markovian reward processes, where the structure of rewards is specified but where no decision structure is used, was carried out at the early stages and some of the results in terms of system reward structures

have been included in this thesis. Transformational methods are useful not only for calculating the total expected rewards as a function of time remaining before the process reaches a finite horizon, but also for determining the asymptotic forms of the reward expressions. Furthermore, the transform analysis is heavily used in a computation of some important probabilities and distributions which arise in a Markov or semi-Markov process, such as the interval transition probabilities, the entrance and destination probabilities, the distribution of first passage times, and the distribution of state occupancies. Since these measures are not directly related to the decision structure and the optimization for a decision process, little attention has been devoted to them in this thesis.

Bibliography

[1] Sheldon M. Ross, *Introduction to Probability Models*, Academic Press, 1993.

[2] Hisashi Mine and Shunji Osaki, *Markovian Decision Processes*, Modern Analytic and Computational Methods in Science and Mathematics, American Elsevier Publishing Company, 1970.

[3] Ronald A. Howard, *Dynamic Programming and Markov Processes*, Massachusetts Institute of Technology Press and John Wiley and Sons, 1960.

[4] Ronald A. Howard, *Dynamic Probabilistic Systems Volumes I & II*, John Wiley and Sons, 1971.

[5] Cyrus Derman, *Finite State Markovian Decision Processes*, Mathematics in Science and Engineering, Volume 67, Academic Press, 1970.

[6] D.J. White, *Finite Dynamic Programming - an Approach to Finite Markov Decision Processes*, John Wiley and Sons, 1978.

[7] John G. Kemeny and J. Laurie Snell, *Finite Markov Chains*, Van Nostrand, 1960.

[8] Richard E. Barlow and Frank Proschan, *Mathematical Theory of Reliability*, John Wiley and Sons, 1965.

[9] David V. Widder, *The Laplace Transform*, Princeton University Press, 1946.

[10] Wade Ellis Jr, et al, *Maple V Flight Manual*, Brooks/Cole, 1992.

[11] Bruce W. Char, et al, *First Leaves - a Tutorial Introduction to MAPLE V*, Springer-Verlag, 1992.

[12] Ronald Pyke, Markov renewal processes and semi-Markov processes preliminary properties, *Ann. Math. Stat.*, Vol. 32, pp.1231-1242, May 1961.

[13] Ronald Pyke, Markovian renewal processes with finitely many states, *Ann. Math. Stat.*, Vol. 32, pp.1243-1259, May 1961.

[14] David Blackwell, Discrete dynamic programming, *Ann. Math. Stat.*, Vol. 33, pp. 719-726, 1962.

[15] Jeremy F. Shapiro, Turnpike planning horizons for a Markovian decision model, *Management Science*, Vol. 14, pp. 292-300, 1968.

Appendix A

Maple Program Listings

1. **Discrete-time Markovian Decision Processes:** *dt_MDP*

   Cases included: discounting & non-discounting; finite & infinite horizons.

   Procedures called: *VIP_mdp*, *PIA_mdp*, *VDO_mdp*, *best_row_index*.

2. **Discrete-time Semi-Markovian Decision Processes:** *dt_SMDP*

   Cases included: discounting & non-discounting; finite & infinite horizons.

   Procedures called: *VIP_smdp*, *PIA_smdp*, *VDO_smdp*, *best_row_index*.

3. **Continuous-time Semi-Markovian Decision Processes:** *ct_SMDP*

   Cases included: discounting & non-discounting; infinite horizon.

   Procedures called: *PIA_smdp*, *VDO_smdp*, *best_row_index*.

4. **Continuous-time Markovian Decision Processes:** *ct_MDP*

   Cases included: discounting & non-discounting; infinite horizon.

   Procedures called: *PIA_mdp*, *VDO_mdp*, *best_row_index*.

A.1 Program *dt_MDP*

```
#########################################################################
#                                                                       #
#              DISCRETE-TIME MARKOVIAN DECISION PROCESS                 #
#                (for both finite & infinite horizon)                   #
#                                                                       #
# Input:                                                                #
#  1. Process indicator specifying the type of MDP:          Process    #
#                                                                       #
#  2. Discounting indicator specifying whether is discounting MDP       #
#     or non-discounting MDP:                              Discounting #
#                                                                       #
#  3. Maximizing or Minimizing Indicator (either MAX or MIN):    key    #
#                                                                       #
#  4. Markov transition-probability matrix of dimension N*K by N        #
#     (N:# of states; K:# of actions), associated with the actions      #
#     (for a terminating MDP the first state is an absorbing state      #
#     and the others are all Transient states):                   P     #
#                                                                       #
#  5. System reward-rate matrix of dimension N*K by N, associated       #
#     with the actions:                                           Rr    #
#                                                                       #
#  6. Finite horizon:                                             FH    #
#                                                                       #
# Output:                                                               #
#  1. Time-independent optimal decision vector making the               #
#     stationary optimal strategy (for infinite horizon):         OD    #
#                                                                       #
#  2. Time-dependent optimal decision matrix making the time-           #
#     dependent optimal policy at each time-remaining n from 1          #
#     to the given finite horizon FH (for finite horizon):        TOD   #
#                                                                       #
#  3. Optimized total expected return vector in discounting case        #
#     (for infinite horizon) or before the process enters in the        #
#     absorbing state in terminating MDP:                         OV    #
#                                                                       #
#  4. Optimized average return per unit time in non-discounting         #
#     case (for infinite horizon):                                Og    #
#                                                                       #
#  5. Optimized total expected return matrix at each time-              #
#     remaining n from 1 to FH (for finite horizon):              VFH   #
#                                                                       #
# Procedures to be loaded in order to run this main program:            #
#                                                                       #
#  1. Policy improvement Algorithm  (for infinite horizon):  PIA_mdp    #
#                                                                       #
#  2. Value determination operation (for infinite horizon):  VDO_mdp    #
#                                                                       #
```

```
#  3. Value iteration method (for finite horizon):            VIP_mdp #
#                                                                      #
#  4. Indices vector of dimension N of maximum or minimum row          #
#     elements of a matrix:                                best_row_index #
#                                                                      #
########################################################################

dt_MDP:=proc() local i,j;

M:=rowdim(P); N:=coldim(P); K:=M/N; m:='m'; n:='n';

# Input display #
print('Markov transition-probability matrix' );
print('associated with the actions:'): print(P );
print('System reward-rate matrix'):
print('associated with the actions:'): print(Rr);
print('Finite horizon:'):              print(FH);
print(' ');

print('You have chosen:');
print('DISCRETE-TIME');

if Process=E or Process=e
   then Process_ID:=COMPLETELY_ERGODIC;
        print('COMPLETE-ERGODIC MARKOVIAN DECISION PROCESS')
   else Process_ID:=TERMINATING;
        print('TERMINATING MARKOVIAN DECISION PROCESS')
fi;

if Discounting<>NO and Discounting<>no and Discounting<>No
   then Discounting_ID:=DISCOUNTING;
        print('with DISCOUNTING & Discount rate is:');
        print(Discounting);
        dr:=Discounting;              # used in "PIA_mdp" & "VIP_mdp" #
   else Discounting_ID:=NO_DISCOUNTING;
        print('without DISCOUNTING!');
        dr:=1.0                       # used in "PIA_mdp" & "VIP_mdp" #
fi; print(' ');

if key<>MIN and key<>min and key<>Min
   then key_ID:=MAX; print('You are doing MAXIMIZATION!'): print(' ')
   else key_ID:=MIN; print('You are doing MINIMIZATION!'): print(' ')
fi;

R:=vector(M,(i)->sum(P[i,j]*Rr[i,j], j=1..N));
print('System-rewards vector for a single transition in this state');
print(R): print(' ');

Indx:=vector(N); V:=vector(M); MP:=P; MR:=R;     # used in "VIP_mdp" #
```

```
PIA_mdp();        # Call the procedure "PIA_mdp" involving "VDO_mdp" #

TOD:=matrix(N,FH); VFH:=matrix(N,FH);
R_M:=convert(R, matrix);                          # used in "VIP_mdp" #

# Call the procedure "VIP_mdp" to obtain TOD & VFH (finite horizon)  #
VIP_mdp();

print('Time-dependent optimal decision matrix');
print('making the time-dependent optimal policy for this MDP');
print('at each time-remaining n from 1 to FH:');
print(TOD): print(' ');
print('Optimized total expected return matrix for this MDP');
print('at each time-remaining n from 1 to FH:');
print(VFH): print(' ');

end;

##################### End of main program: dt_MDP ######################
```

A.2 Program *dt_SMDP*

```
#########################################################################
#                                                                       #
#          DISCRETE-TIME SEMI-MARKOVIAN DECISION PROCESS                #
#               (for both finite & infinite horizon)                    #
#                                                                       #
# Input:                                                                #
#  1. Process indicator specifying the type of semi-MDP:     Process    #
#                                                                       #
#  2. Discounting indicator specifying whether is discounting           #
#     semi-MDP or non-discounting MDP:                    Discounting   #
#                                                                       #
#  3. Maximizing or Minimizing Indicator (either MAX or MIN):    key    #
#                                                                       #
#  4. Markov transition-probability matrix of dimension N*K by N        #
#     (N:# of states; K:# of actions), associated with the actions      #
#     (for a terminating semi-MDP the first state is an absorbing       #
#     state and the others are all Transient states):              P    #
#                                                                       #
#  5. Arbitrary holding-time pmf matrix of dimension N*K by N           #
#     (function of time n), associated with the actions, with an        #
#     initial value of zero at time n=0:                        HT(n)   #
#                                                                       #
#  6. Yield matrix of dimension N*K by N (function of time n),          #
#     associated with the actions:                              YR(n)   #
#                                                                       #
#  7. Bonus matrix of dimension N*K by N (function of time n),          #
#     associated with the actions:                               B(n)   #
#                                                                       #
#  8. Initial system-reward vector:                              V0      #
#                                                                       #
#  9. Finite horizon:                                            FH      #
#                                                                       #
# Output:                                                               #
#  1. Time-independent optimal decision vector making the               #
#     stationary optimal strategy (for infinite horizon):        OD      #
#                                                                       #
#  2. Time-dependent optimal decision matrix making the time-           #
#     dependent optimal policy at each time remaining n from 1          #
#     to the given finite horizon FH (for finite horizon):       TOD     #
#                                                                       #
#  3. Optimized total expected return vector in discounting case        #
#     (for infinite horizon) or before the process enters in the        #
#     absorbing state in terminating semi-MDP:                   OV      #
#                                                                       #
#  4. Optimized average return per unit time in non-discounting         #
#     case (for infinite horizon):                               Og      #
#                                                                       #
```

```
#  5. Optimized total expected return matrix at each time-          #
#     remaining n from 1 to FH (for finite horizon):          VFH #
#                                                                  #
# Procedures to be loaded in order to run this main program:       #
#                                                                  #
#  1. Policy improvement Algorithm  (for infinite horizon): PIA_smdp #
#                                                                  #
#  2. Value determination operation (for infinite horizon): VDO_smdp #
#                                                                  #
#  3. Value iteration method (for finite horizon):          VIP_smdp #
#                                                                  #
#  4. Indices vector of dimension N of maximum or minimum row      #
#     elements of a matrix:                          best_row_index #
#                                                                  #
####################################################################

dt_SMDP:=proc() local i,j;

M:=rowdim(P); N:=coldim(P); K:=M/N; m:='m'; n:='n';

# Input display #
print(`Markov transition-probability matrix` );
print(`associated with the actions:`): print(P);
print(`Holding-time pmf matrix (function of time n)`);
print(`associated with the actions:`): print(HT(n));
print(`Yield matrix (function of time n)`);
print(`associated with the actions:`): print(YR(n));
print(`Bonus matrix (function of time n)`);
print(`associated with the actions:`): print(B(n));
print(`Initial reward vector:`):       print(V0);
print(`Finite horizon:`):              print(FH);
print(` `);

print(`You have chosen:`);
print(`DISCRETE-TIME`);
if Process=E or Process=e
   then Process_ID:=COMPLETELY_ERGODIC;
        print(`COMPLETE-ERGODIC SEMI-MARKOVIAN DECISION PROCESS`)
   else if Process=T or Process=t
           then Process_ID:=TERMINATING;
                print(`TERMINATING MARKOVIAN DECISION PROCESS`)
           else print(`neither COMPLETELY-ERGODIC nor TERMINATING`);
                print(`MARKOVIAN DECISION PROCESS`)
        fi
fi;

if Discounting<>NO and Discounting<>no and Discounting<>No
   then Discounting_ID:=DISCOUNTING;
        print(`with DISCOUNTING & Discounting-rate is:`);
        print(Discounting);
```

```
        dr   :=Discounting;        # used in "PIA_smdp" & "VIP_smdp" #
        z:=dr;
        HTg:=matrix(M,N,(i,j)->sum(HT(n)[i,j]*z^n, n=1..infinity));
        P_HTg:=matrix(M,N,(i,j)->P[i,j]*HTg[i,j]);
        z:='z'; n:='n';
#HTg is geometric transform of HT; P_HTg is direct product of P & HTg#

   else Discounting_ID:=NO_DISCOUNTING;
        print(`without DISCOUNTING!`);
        dr  :=1.0;                                  # used in "VIP_smdp" #
        MHTD:=matrix(M,N,(i,j)->sum(n*HT(n)[i,j], n=1..infinity));
        MWTD:=vector(M,(i)->sum(P[i,j]*MHTD[i,j], j=1..N));
        MWTD_M:=convert(MWTD, matrix);
        print(MWTD); print(MWTD_M);
fi; print(` `);

if key<>MIN and key<>min and key<>Min
   then key_ID:=MAX; print(`You are doing MAXIMIZATION!`): print(` `)
   else key_ID:=MIN; print(`You are doing MINIMIZATION!`): print(` `)
fi;

# The following elements will be used in the procedure "PIA_smdp":   #
#    RR: Contribution of both yield & bonus (system reward for each  #
#        transition) associated with the actions;                    #
#    RS: System reward vector of dimension N*K (function of n);      #
#    RF: System future reward vector of dimension N*K in a long run; #
# P_HTt: The direct product between the transition probability       #
#        matrix P and either geometric transform matrix of holding-  #
#        time for discrete-time case or Laplace transform matrix of  #
#        holding-time for continuous-time case.                      #

Y    :=matrix(M,N,(i,j)->sum(YR(k)[i,j]*dr^k,k=0..m-1));
RR   :=matrix(M,N,(i,j)->sum(HT(m)[i,j]
           *(Y[i,j] + B(m)[i,j]*dr^m), m=1..n));
RS   :=vector(M,(i)->sum(P[i,j]*RR[i,j], j=1..N));
RF   :=vector(M,(i)->limit(RS[i], n=infinity)); n:='n';
P_HTt:=P_HTg;

print(`Expected system reward vector for each transition at each n:`);
print(RS): print(` `);
print(`Expected system future reward vector in a long run:`);
print(RF): print(` `);

Indx:=vector(N); V:=vector(M);

PIA_smdp();     # Call the procedure "PIA_smdp" involving "VDO_smdp" #


# The following elements will be used in the procedure "VIP_smdp":   #
#  Y: Present yield matrix of the rewards generated   #
```

```
#     by the yield-rate YR during an state occupancy; #
# Y1: Discounted yield-rates contribution where the   #
#     process makes no transition before time n;      #
# Wc: The complement of waiting-time distribution;    #

Y   :=matrix(M,N,(i,j)->sum(YR(m)[i,j]*dr^m, m=0..n-1));
HTDc:=matrix(M,N,(i,j)->sum(HT(m)[i,j], m=n+1..infinity));
Y1  :=vector(M,(i)->sum(P[i,j]*Y[i,j]*HTDc[i,j], j=1..N));
Wc  :=vector(M,(i)->sum(P[i,j]*HTDc[i,j], j=1..N));

TOD:=matrix(N,FH); VFH:=matrix(N,FH);

# Call the procedure "VIP_smdp" to obtain TOD & VFH (finite horizon) #
VIP_smdp();

print('Time-dependent optimal decision matrix');
print('making the time-dependent optimal policy for this Semi-MDP');
print('at each time-remaining n from 1 to FH:');
print(TOD): print(' ');
print('Optimized total expected return matrix for this Semi-MDP');
print('at each time-remaining n from 1 to FH:');
print(VFH): print(' ');

end;

#################### End of main program "dt_SMDP" ####################
```

A.3 Program *ct_SMDP*

```
#########################################################################
#                                                                       #
#          CONTINUOUS-TIME SEMI-MARKOVIAN DECISION PROCESS              #
#                    (for finiteinfinite horizon)                       #
#                                                                       #
# Input:                                                                #
#  1. Process indicator specifying the type of semi-MDP:     Process    #
#                                                                       #
#  2. Discounting indicator specifying whether is discounting           #
#     semi-MDP or non-discounting MDP:                    Discounting   #
#                                                                       #
#  3. Maximizing or Minimizing Indicator (either MAX or MIN):    key    #
#                                                                       #
#  4. Markov transition-probability matrix of dimension N*K by N        #
#     (N:# of states; K:# of actions), associated with the actions      #
#     (for a terminating semi-MDP the first state is an absorbing       #
#     state and the others are all Transient states):              P    #
#                                                                       #
#  5. Arbitrary holding-time pdf matrix of dimension N*K by N           #
#     (function of time t), associated with the actions:        HT(t)   #
#                                                                       #
#  6. Yield-rate matrix of dimension N*K by N (function of              #
#     time t), associated with the actions:                     YR(t)   #
#                                                                       #
#  7. Bonus matrix of dimension N*K by N (function of time t),          #
#     associated with the actions:                               B(t)   #
#                                                                       #
# Output:                                                               #
#  1. Optimal decision vector making the stationary optimal             #
#     strategy (for infinite horizon):                            OD    #
#                                                                       #
#  2. Optimized total expected return vector in discounting case        #
#     (for infinite horizon) or before the process enters in the        #
#     absorbing state in terminating semi-MDP:                    OV    #
#                                                                       #
#  3. Optimized average return per unit time in non-discounting         #
#     case (for infinite horizon):                                Og    #
#                                                                       #
# Procedures to be loaded in order to run this main program:            #
#  1. Policy improvement Algorithm  (for infinite horizon): PIA_smdp    #
#                                                                       #
#  2. Value determination operation (for infinite horizon): VDO_smdp    #
#                                                                       #
#  3. Indices vector of dimension N of maximum or minimum row           #
#     elements of a matrix:                          best_row_index     #
#                                                                       #
#########################################################################
```

```
ct_SMDP:=proc() local i,j;

M:=rowdim(P); N:=coldim(P); K:=M/N; t:='t'; s:='s';

# Input display #
print('Markov transition-probability matrix' );
print('associated with the actions:'): print(P);
print('Holding-time pdf matrix (function of time t)');
print('associated with the actions:'): print(HT(t));
print('Yield matrix (function of time t)');
print('associated with the actions:'): print(YR(t));
print('Bonus matrix (function of time n)');
print('associated with the actions:'): print(B(t));
print(' ');

print('You have chosen:');
print('CONTINUOUS-TIME');
if Process=E or Process=e
   then Process_ID:=COMPLETELY_ERGODIC;
        print('COMPLETE-ERGODIC SEMI-MARKOVIAN DECISION PROCESS')
   else if Process=T or Process=t
           then Process_ID:=TERMINATING;
                print('TERMINATING MARKOVIAN DECISION PROCESS')
           else print('neither COMPLETELY-ERGODIC nor TERMINATING');
                print('MARKOVIAN DECISION PROCESS')
        fi
fi;

if Discounting<>NO and Discounting<>no and Discounting<>No
   then Discounting_ID:=DISCOUNTING;
        print('with DISCOUNTING & Discounting-rate is:');
        print(Discounting);
        dr :=Discounting;                          # used in "PIA_smdp" #
        HTe:=matrix(M,N,(i,j)->laplace(HT(t)[i,j], t,s));
        s:=Discounting;   #Laplace transform of HT is evalued at dr #
        P_HTe:=matrix(M,N,(i,j)->P[i,j]*HTe[i,j]); s:='s'
   else Discounting_ID:=NO_DISCOUNTING;
        print('without DISCOUNTING!');
        dr :=0;                                    # used in "PIA_smdp" #
        MHTD:=matrix(M,N,(i,j)->int(t*HT(t)[i,j], t=0..infinity));
        MWTD:=vector(M,(i)->sum(P[i,k]*MHTD[i,k], k=1..N));
        MWTD_M:=convert(MWTD, matrix)
fi; print(' ');

if key<>MIN and key<>min and key<>Min
   then key_ID:=MAX; print('You are doing MAXIMIZATION!'): print(' ')
   else key_ID:=MIN; print('You are doing MINIMIZATION!'): print(' ')
fi;
```

```
# The following elements will be used in the procedure "PIA_smdp":   #
#     Y: Present value matrix of the rewards generated by the        #
#        yield-rates YR during an state occupancy;                   #
#    RR: Contribution of both yield-rate & bonus (system reward      #
#        for each transition) associated with the actions;           #
#    RS: System reward vector of dimension N*K (function of t);      #
#    RF: System future reward vector of dimension N*K in a long run; #
# P_HTt: The direct product between the transition probability       #
#        matrix P and either geometric transform matrix of holding-  #
#        time for discrete-time case or Laplace transform matrix of  #
#        holding-time for continuous-time case.                      #

Y    :=matrix(M,N,(i,j)->int(YR(t)[i,j]*exp(-dr*t), t=0..time));
RR   :=matrix(M,N,(i,j)->int(HT(time)[i,j]
           *(Y[i,j] + exp(-dr*time)*B(time)[i,j]), time=0..t));
RS   :=vector(M,(i)->sum(P[i,j]*RR[i,j], j=1..N));
RF   :=vector(M,(i)->limit(RS[i], t=infinity));
P_HTt:=P_HTe;

print('Expected system reward vector for each transition at time t:');
print(RS): print(' ');
print('Expected system future reward vector in the long run:');
print(RF): print(' ');

Indx:=vector(N); V:=vector(M); MP:=P;

PIA_smdp();     # Call the procedure "PIA_smdp" involving "VDO_smdp" #

end;

#################### End of main program "ct_SMDP" ####################
```

A.4 Program *ct_MDP*

```
#########################################################################
#                                                                       #
#             CONTINUOUS-TIME MARKOVIAN DECISION PROCESS                #
#                    (for infinite horizon only)                        #
#                                                                       #
# Input:                                                                #
#  1. Process indicator specifying the type of MDP:          Process    #
#                                                                       #
#  2. Discounting indicator specifying whether is discounting MDP       #
#     or non-discounting MDP:                                Discounting #
#                                                                       #
#  3. Maximizing or Minimizing Indicator (either MAX or MIN):    key     #
#                                                                       #
#  4. Markov transition-probability matrix of dimension N*K by N        #
#     (N:# of states; K:# of actions), associated with the actions      #
#     (for a terminating MDP the first state is an absorbing state      #
#     and the others are all Transient states):                  P      #
#                                                                       #
#  5. System reward-rate matrix of dimension N*K by N, associated       #
#     with the actions:                                          Rr     #
#                                                                       #
#  6. Exponential waiting-rate vector of dimension N*K, associated      #
#     with the actions:                                          EWR    #
#                                                                       #
# Output:                                                               #
#  1. Optimal decision vector making the stationary optimal             #
#     strategy (for infinite horizon):                           OD     #
#                                                                       #
#  2. Optimized total expected return vector in discounting case        #
#     (for infinite horizon) or before the process enters in the        #
#     absorbing state in terminating MDP:                        OV     #
#                                                                       #
#  3. Optimized average return per unit time in non-discounting         #
#     case (for infinite horizon):                               Og     #
#                                                                       #
# Procedures to be loaded in order to run this main program:            #
#                                                                       #
#  1. Policy improvement Algorithm  (for infinite horizon):  PIA_mdp    #
#                                                                       #
#  2. Value determination operation (for infinite horizon):  VDO_mdp    #
#                                                                       #
#  3. Indices vector of dimension N of maximum or minimum row           #
#     elements of a matrix:                              best_row_index #
#                                                                       #
#########################################################################
```

```
ct_MDP:=proc() local i,j;

M:=rowdim(P); N:=coldim(P); K:=M/N; t:='t'; s:='s';

# Input display #
print('Markov transition-probability matrix' );
print('associated with the actions:'): print(P );
print('System reward-rate matrix');
print('associated with the actions:'): print(Rr);
print('Exponential waiting-rate matrix');
print('associated with the actions:'): print(EWR);
print(' ');

# Generate the corresponding TRANSITION-RATE matrix A of dimension #
# N*K by N and the system EARNING-RATE vector Q of dimension N*K.  #
B:=copy(P);
for i from 1 to N do
  for j from (i-1)*K+1 to i*K do B[j,i]:=P[j,i]-1 od
                    od;
A:=matrix(M,N,(i,j)->B[i,j]/EWR[i]); MP:=A;

print('The corresponding TRANSITION-RATE matrix');
print('associated with the actions:'): print(A): print(' ');

Q:=vector(M);
for i from 1 to N do
  for j from (i-1)*K+1 to i*K do
    Q[j]:=(1-A[j,i])*Rr[j,i] + sum(A[j,k]*Rr[j,k], k=1..N)
                              od
                    od;
print('The corresponding expected EARNING-RATE vector');
print('associated with the actions:'): print(Q): print(' ');

print('You have chosen:');
print('CONTINUOUS-TIME');
if Process=E or Process=e
   then Process_ID:=COMPLETELY_ERGODIC;
        print('COMPLETE-ERGODIC MARKOVIAN DECISION PROCESS')
   else if Process=T or Process=t
           then Process_ID:=TERMINATING;
                print('TERMINATING MARKOVIAN DECISION PROCESS')
           else print('neither COMPLETELY-ERGODIC nor TERMINATING');
                print('MARKOVIAN DECISION PROCESS')
        fi
fi;

if Discounting<>NO and Discounting<>no and Discounting<>No
   then Discounting_ID:=DISCOUNTING;
        print('with DISCOUNTING & Discounting-rate is:');
        print(Discounting);
```

```
        dr:=1/Discounting;          # used in "PIA_mdp" & "VIP_mdp" #
   else Discounting_ID:=NO_DISCOUNTING;
        print('without DISCOUNTING!');
        dr:=1;                      # used in "PIA_mdp" & "VIP_mdp" #
        if Process_ID<>TERMINATING
           then A1:=copy(A);
                for i from 1 to N do
                  for j from (i-1)*K+1 to i*K do A1[j,i]:=A[j,i]+1 od
                                    od;
                MP:=A1
        fi
fi; print(' ');

if key<>MIN and key<>min and key<>Min
   then key_ID:=MAX; print('You are doing MAXIMIZATION!'): print(' ')
   else key_ID:=MIN; print('You are doing MINIMIZATION!'): print(' ')
fi;

Indx:=vector(N); V:=vector(M); MR:=evalm(Q*dr); # used in "PIA_mdp" #

PIA_mdp();       # Call the procedure "PIA_mdp" involving "VDO_mdp" #

end;

##################### End of main program: ct_MDP ####################
```

Appendix B

Maple Procedure Listings

1. *VIP_mdp*
   Value Iteration Method for discrete-time **MDP** with finite horizon.
2. *PIA_mdp*
   Policy Improvement Algorithm for **MDP** with infinite horizon.
3. *VDO_mdp*
   Value Determination Operation for **MDP** with infinite horizon.
4. *VIP_smdp*
   Value Iteration Method for discrete-time **Semi-MDP** with finite horizon.
5. *PIA_smdp*
   Policy Improvement Algorithm for **Semi-MDP** with infinite horizon.
6. *VDO_smdp*
   Value Determination Operation for **Semi-MDP** with infinite horizon.
7. *best_row_index*
   Indices of maximum or minimum row elements in a matrix.

B.1 Procedure *VIP_mdp*

```
#########################################################################
#                                                                       #
#                  Value Iteration Method (VIP)                         #
#        (for discrete-time MDP over a chosen finite horizon)           #
#                                                                       #
#    To find a time-dependent optimal decision matrix TOD giving        #
#    the time-dependent optimal policy which maximizes the total        #
#    expected return matrix VFH at each time remaining n from 1 to      #
#    given finite horizon FH in discounting/non-discounting case        #
#                                                                       #
#########################################################################

VIP_mdp:=proc() local i;

for n from 1 to FH do
    for i from 1 to M do
        if n=1 then V[i]:= R[i];
          else V[i]:=R[i] + sum(P[i,j]*VFH[j,n-1]*dr, j=1..N);
       fi;
    od;

# Reshape V for using the procedure "best_row_index"                    #
    Shaped:=matrix(N,K,(i,j)->V[(i-1)*K + j]);

# Call the procedure "best_row_index" producing a new decision vector#
    best_row_index();

    Indx_V:=evalm(Indx + vector([seq(i*K, i=0..N-1)]));
    Indx_L:=convert(Indx_V,  list);
    V_M    :=convert(V, matrix);
    V_M_sub:=submatrix(V_M, Indx_L, 1..1);
    V_sub  :=convert(V_M_sub,vector);

    for i from 1 to N do VFH[i,n]:=V_sub[i]; TOD[i,n]:=Indx[i] od;
od;

end;

################## End of the procedure "VIP_mdp " ##################
```

B.2 Procedure *PIA_mdp*

```
#########################################################################
#                                                                       #
#                POLICY IMPROVEMENT ALGORITHM (PIA)                     #
#    (for discrete or continuous-time MDP over infinite horizon)        #
#                                                                       #
#   To find a time-independent optimal decision vector OD giving        #
#   the stationary optimal strategy over an infinite horizon that       #
#   maximizes either the total discounted expected return vector        #
#   V in a discounting case or the average return g per unit time       #
#   in a non-discounting case. For a TERMINATING MDP to maximize        #
#   the total expected return vector V before the process enters        #
#   the absorbing state (state 1).                                      #
#                                                                       #
#########################################################################

PIA_mdp:=proc() local i,j;

# Reshape MR for using the procedure "best_row_index"                   #
Shaped:=matrix(N,K,(i,j)->MR[(i-1)*K + j]);

# Call procedure "best_row_index" producing an initial decision         #
best_row_index();
Decision:=vector(N,0); iteration:=0; MR_M:=convert(MR, matrix);

while not(equal(Decision, Indx))
  do iteration:=iteration +1;
     print('VDO Iteration No.'): print(iteration);
     Decision:=copy(Indx);

     # Call the procedure "VDO_mdp" proceduring either OV or Og & VR #
     VDO_mdp();

     if Process_ID=TERMINATING or Discounting_ID=DISCOUNTING
        then V:=evalm(MR + dr*MP&*VF);
# The above is for Terminating or discounted Completely-Ergodic MDP     #
        else V:=evalm(MR + MP&*VR - g)
     # The above is for non-discounted, Completely-Ergodic MDP only     #
     fi;

     # Reshape V for using "best_row_index" #
     Shaped:=matrix(N,K,(i,j)->V[(i-1)*K + j]);

     # Call the procedure "best_row_index" producing a new decision     #
     best_row_index();
  od;

OD:=Decision;
```

```
print('Time-independent optimal decision vector OD');
print('making the stationary optimal strategy over infinite horizon');
print(OD): print(' ');

if Process_ID=TERMINATING or Discounting_ID=DISCOUNTING
   then OV:=VF;
        print('Optimized total expected return vector under OD:');
        print(OV): print(' ')
   else Og:=g;
        print('Optimized average return per unit time under OD:');
        print(Og): print(' ')
   fi;

end;

################### End of the procedure "PIA_mdp" ####################
```

B.3 Procedure *VDO_mdp*

```
#########################################################################
#                                                                       #
#                VALUE DETERMINATION OPERATION  (VDO)                   #
#  To compute either the total expected future return value vector      #
#  VF for an infinite horizon in discounting case or average future     #
#  gain value g per unit time in non-discounting case according to      #
#  the decision chosen in PIA for discrete or continuous-time MDP.      #
#                                                                       #
#########################################################################

VDO_mdp:=proc() local i;

print('Decision given in the PIA and VDO iteration procedure:');
print(Decision);

# Generate an index-list: Indx_L, for getting sub-matrix& sub-vector #
# according to the decision vector OD chosen in PIA for using VDO.   #

Indx_V  :=evalm(Decision + vector([seq(i*K, i=0..N-1)]));
Indx_L  :=convert(Indx_V, list);
MP_sub  :=submatrix(MP, Indx_L, 1..N);
MR_M_sub:=submatrix(MR_M, Indx_L, 1..1);

if Process_ID=COMPLETELY_ERGODIC
   then MR_sub:=convert(MR_M_sub, vector);
        if Discounting_ID=DISCOUNTING
           then # In this case, it is a C.E. MDP with discounting    #
                VF:=evalm(inverse(1 - dr*MP_sub)&*MR_sub);
                print('Total expected return  vector VF');
                print('in this iteration is:');
                print(VF): print(' ')

           else # In this case, it is a C.E. MDP without discounting #
                MPP:=evalm(1 - MP_sub);
                for i from 1 to N do MPP[i,N]:=1 od;
                VR:=evalm(inverse(MPP)&*MR_sub);
                g :=VR[N]; VR[N]:=0;
                print('The relative return vector VR');
                print('in this iteration is:');
                print(VR): print(' ');
                print('and the average return g per unit time:');
                print(g):  print(' ')
        fi

   else #In this case, it is a Terminating MDP w/ or w/o discounting #
        MP_sub2  :=submatrix(MP_sub, 2..N, 2..N);
        MR_M_sub2:=submatrix(MR_M_sub, 2..N, 1..1);
```

```
        MR_sub2  :=convert  (MR_M_sub2, vector);
        VV  :=evalm  (inverse(1-dr*MP_sub2)&*MR_sub2);
        VV_M:=convert(VV, matrix);
        VF_M:=stack  (vector([0]), VV_M);
        VF  :=convert(VF_M, vector);
        print('In this iteration, total expected return vector VF');
        print('before the process enters the absorbing state is:');
        print(VF)
fi;

end;

################## End of the procedure "VDO_mdp"  ####################
```

B.4 Procedure *VIP_smdp*

```
#########################################################################
#                                                                       #
#                  Value Iteration Method (VIP)                         #
#     (for discrete-time Semi-MDP over a chosen finite horizon)         #
#                                                                       #
#    To find a time-dependent optimal decision matrix TOD giving        #
#    the time-dependent optimal policy which maximizes the total        #
#    expected return matrix VFH at each time remaining n from 1 to      #
#    given finite horizon FH in discounting/non-discounting case.       #
#                                                                       #
#########################################################################

VIP_smdp:=proc() local i,j;

for n from 1 to FH do
   RS_M:=convert(RS, matrix); # RS is evaluated for time remaining n #
   Y1_M:=convert(Y1, matrix); # Y1 is evaluated for time remaining n #
   Wc_M:=convert(Wc, matrix); # Wc is evaluated for time remaining n #

   for i from 1 to M do
       if n=1 then s:=sum(P[i,j]*HT(n)[i,j]*V0[j]*dr, j=1..N);
          else s:=sum(P[i,j]*(HT(n)[i,j]*V0[j]*dr^n
                 +sum(HT(m)[i,j]*VFH[j,n-m]*dr^m, m=1..n-1)), j=1..N)
       fi;
       V[i]:=Y1[i] + V0[i]*Wc[i]*dr^n + RS[i] + s
   od;

# Reshape V for using the procedure "best_row_index"                    #
   Shaped:=matrix(N,K,(i,j)->V[(i-1)*K + j]);

# Call the procedure "best_row_index" producing a new decision          #
   best_row_index();

   Indx_V :=evalm(Indx + vector([seq(i*K, i=0..N-1)]));
   Indx_L :=convert(Indx_V, list);
   V_M    :=convert(V, matrix);
   V_M_sub:=submatrix(V_M, Indx_L, 1..1);
   V_sub  :=convert(V_M_sub,vector);

   for i from 1 to N do VFH[i,n]:=V_sub[i]; TOD[i,n]:=Indx[i] od;
od;

end;

################### End of the procedure "VIP_smdp " ###################
```

B.5 Procedure *PIA_smdp*

```
#########################################################################
#                                                                       #
#                POLICY IMPROVEMENT ALGORITHM (PIA)                     #
#  (for discrete or continuous-time Semi-MDP over infinite horizon)     #
#                                                                       #
#   To find a time-independent optimal decision vector OD giving        #
#   the stationary optimal strategy over an infinite horizon that       #
#   maximizes either the total discounted expected return vector        #
#   V in a discounting case or the average return g per unit time       #
#   in a non-discounting case. For a TERMINATING MDP to maximize        #
#   the total expected return vector V before the process enters        #
#   the absorbing state (state 1).                                      #
#                                                                       #
#########################################################################

PIA_smdp:=proc() local i,j;

# Reshape RF for using the procedure "best_row_index"                   #
Shaped:=matrix(N,K,(i,j)->RF[(i-1)*K + j]);

# Call procedure "best_row_index" producing an initial decision         #
best_row_index();
Decision:=vector(N,0); iteration:=0; RF_M:=convert(RF, matrix);

while not(equal(Decision, Indx))
  do iteration:=iteration +1;
     print('VDO Iteration No.'): print(iteration);
     Decision:=copy(Indx);

   # Call the procedure "VDO_smdp" proceduring either OV or Og & VR     #
     VDO_smdp();

     if Discounting_ID=DISCOUNTING
       then V:=evalm(RF + P_HTt&*VF)
     # The above is for all types of semi-MDP with discounting          #
        else V:=evalm(RF + P&*VR - g*MWTD)
     # The above is for all types of semi-MDP without discounting       #
     fi;

     # Reshape V for using "best_row_index" #
     Shaped:=matrix(N,K,(i,j)->V[(i-1)*K + j]);

     # Call the procedure "best_row_index" producing a new decision     #
     best_row_index();
  od;

OD:=Decision;
```

```
print('Time-independent optimal decision vector OD');
print('making the stationary optimal strategy over infinite horizon');
print(OD): print(' ');

if Discounting_ID=DISCOUNTING
   then OV:=VF;
        print('Optimized total expected return vector under OD:');
        print(OV): print(' ')
   else Og:=g;
        print('Optimized average return per unit time under OD:');
        print(Og): print(' ')
   fi;

end;

################### End of the procedure "PIA_smdp" ##################
```

B.6 Procedure *VDO_smdp*

```
#########################################################################
#                                                                       #
#                VALUE DETERMINATION OPERATION  (VDO)                   #
#  To compute either the total expected future return value vector      #
#  VF for an infinite horizon in a discounting case or the average      #
#  gain value g per unit time in a non-discounting case according to    #
#  the decision chosen in PIA for discrete/continuous-time Semi-MDP.    #
#                                                                       #
#########################################################################

VDO_smdp:=proc() local i;

print('Decision given in the PIA and VDO iteration procedure:');
print(Decision);

# Generate an index-list: Indx_L, for getting sub-matrix& sub-vector    #
# according to the decision vector OD chosen in PIA for using VDO.      #

Indx_V  :=evalm(Decision + vector([seq(i*K, i=0..N-1)]));
Indx_L  :=convert(Indx_V, list);
RF_M_sub:=submatrix(RF_M, Indx_L, 1..1);

if Process_ID=COMPLETELY_ERGODIC
   then RF_sub:=convert(RF_M_sub, vector);
        if Discounting_ID=DISCOUNTING
           then P_HTt_sub:=submatrix(P_HTt, Indx_L, 1..N);
                VF:=evalm(inverse(1 - P_HTt_sub)&*RF_sub);
                print('The total expected return vector VF');
                print('in this iteration is:'):
                print(VF): print(' ')
           else P_sub    :=submatrix(P, Indx_L, 1..N);
                MWTD_M_sub:=submatrix(MWTD_M, Indx_L, 1..1);
                MWTD_sub  :=convert  (MWTD_M_sub, vector);
                PP        :=evalm    (1 - P_sub);
                for i from 1 to N do PP[i,N]:=MWTD_sub[i] od;
                VR:=evalm(inverse(PP)&*RF_sub);
                g :=VR[N]; VR[N]:=0;
                print('The relative return vector VR');
                print('in this iteration is:'):
                print(VR): print(' ');
                print('and the average return g per unit time:');
                print(g):  print(' ')
        fi

   else RF_M_sub2 :=submatrix(RF_M_sub, 2..N, 1..1);
        RF_sub2   :=convert  (RF_M_sub2, vector);
```

```
        if Discounting_ID=DISCOUNTING
           then P_HTt_sub :=submatrix(P_HTt, Indx_L, 1..N);
                P_HTt_sub2:=submatrix(P_HTt_sub, 2..N, 2..N);
                VV  :=evalm  (inverse(1-P_HTt_sub2)&*RF_sub2);
                VV_M:=convert(VV, matrix);
                VF_M:=stack  (vector([0]), VV_M);
                VF  :=convert(VF_M, vector);
                print('The total expected return vector VF');
                print('in this iteration is:');
                print(VF): print(' ')
           else P_sub     :=submatrix(P, Indx_L, 1..N);
                MWTD_M_sub :=submatrix(MWTD_M,Indx_L, 1..1);
                P_sub2     :=submatrix(P_sub, 2..N, 2..N);
                MWTD_M_sub2:=submatrix(MWTD_M_sub, 2..N, 1..1);
                MWTD_sub2  :=convert  (MWTD_M_sub2, vector);
                PP2:=evalm(1 - P_sub2);
                for i from 1 to N-1 do PP2[i,N-1]:=MWTD_sub2[i] od;
                VV  :=evalm(inverse(PP2)&*RF_sub2);
                g   :=VV[N-1]; VV[N-1]:=0;
                VV_M:=convert(VV, matrix);
                VR_M:=stack  (vector([0]), VV_M);
                VR  :=convert(VR_M, vector);
                print('The relative return vector VR');
                print('in this iteration is:');
                print(VR): print(' ');
                print('and the average return g per unit time:');
                print(g):  print(' ')
        fi
fi;

end;

################## End of the procedure "VDO_smdp"  ###################
```

B.7 Procedure *best_row_index*

```
#########################################################################
#                                                                       #
#     THE INDICES of MAXIMUM or MINIMUM ELEMENTS OF MATRIX ROWS         #
#        (Used as an external-procedure in other procedures)            #
#                                                                       #
# Input :                                                               #
#  1. Shaped matrix of dimention N by K:                        Shaped  #
#  2. Maximizing or Minimizing Indicator (either MAX or MIN):key_ID     #
#                                                                       #
# Output:                                                               #
#  1. N dimensional Vector of Indices of maximum or minimum             #
#     elements of the input matrix (used to produce an improved         #
#     decision vector or matrix in PIA):                          Indx  #
#                                                                       #
#########################################################################

best_row_index:= proc() local i,j,k, small, big;

if key_ID=MAX
  then for i from 1 to N
         do big:=Shaped[i,1]; k:= 1;
           for j from 2 to K
             do if big<Shaped[i,j] then big:=Shaped[i,j]; k:=j fi
             od;
           Indx[i] :=k
         od
  else for i from 1 to N
         do small:=Shaped[i,1]; k:= 1;
           for j from 2 to K
             do if small>Shaped[i,j] then small:=Shaped[i,j]; k:=j fi
             od;
           Indx[i] :=k
         od
fi;
end;

############ End of external-procedure "best_row_index" #############
```

VITA

Surname: Ren
Given Names: Zhi-Zhong Oscar
Place of Birth: P. R. China
Date of Birth: October 13, 1961

Educational Institutions Attended:

University of Victoria	September 1992 to September 1994
Fudan University	September 1979 to May 1983

Degree Awarded:

B.Sc.	Fudan University	1983

Honours and Awards:

University of Victoria Fellowship	1992-1994

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis: **OPTIMIZATION PROCEDURES FOR MARKOVIAN AND SEMI-MARKOVIAN DECISION PROCESSES**

Author [redacted]

ZHI-ZHONG OSCAR REN

(Name in Block Letter)

Sep 30, 94

(Date)