Classification Problems for MDPs and Optimal Customer Admission to M/M/k/N Queues

A Dissertation Presented

by

Fenghsu Yang

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

December 2010

Abstract of the Dissertation

Classification Problems for MDPs and Optimal Customer Admission to M/M/k/N Queues

by

Fenghsu Yang

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2010

My dissertation addresses the unichain classification problem for any finite Markov decision process (MDP) with a recurrent or stopping state and the optimal admission problem for an M/M/k/N queue with holding costs. In the first chapter, we study the unichain classification problem for MDPs. The unichain classification problem is to detect whether an MDP with finite states and actions is unichain or not. This problem has been proven to be NP-hard. We study this problem while an MDP contains a state which is either recurrent under all deterministic policies or absorbing under some action. We introduce the definitions of avoidable and reachable sets and provide the corresponding polynomial algorithms that finds the states from which a given set is avoidable or reachable. We also provide a polynomial algorithm that detects whether a state is recurrent and solves the unichain classification problem for an MDP with a recurrent state and a polynomial algorithm for finding all recurrent and stopping states and solving the unichain classification problem with recurrent or stopping states. At the end of the first chapter, we discuss detecting all transient states in an MDP in polynomial time.

In the second chapter, we study optimal admission of arrivals to an M/M/k/N queue. The arriving customers are classified into m types, where $m \geq 1$. The rewards and holding costs depend on customer types. Upon admitting an arriving customer, the system collects the reward from the admitted customer and pays the holding cost to the admitted customer. We study average reward per unit time for the problem. We prove the existence of an optimal trunk reservation policy and describe the structures of stationary optimal, canonical, bias optimal, and Blackwell optimal policies. If there exist two or more stationary optimal policies, we apply more sensitive optimality criteria to detect the best policy among all stationary optimal policies. We show that bias optimal and Blackwell optimal policies are unique, coincide, and are the trunk reservation policies with the largest optimal control level for each customer type.

Key Words: Unichain classification, Markov decision process, recurrent state, polynomial algorithm, queueing system, trunk reservation policy, stationary optimal policy, canonical policy, bias optimal policy, Blackwell optimal policy.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to thank Professor Eugene Feinberg, my advisor, for his suggestions and support during the past few years. My understanding of the research topics and directions have been shaped to a large extent under his guidance. His profound knowledge and inside thoughts for the research have always encouraged me to achieve a higher goal. I am really thankful for having him as my advisor and I will always remember his patience and advice for me during the time I was at Stony Brook

Also, I would like to thank Professor Joseph Mitchell, who gave me ideas about the future directions in the field of queueing systems. I would like to memorize Professor Woo Jong Kim, who suggested me to study in the field of stochastic processes during my first year at Stony Brook, and extend my thanks to Professor Xiaolin Li, who encouraged me to study further in the Ph.D. program at Stony Brook.

In my family, I would like to thank my mother for her support, Weilin Chang, my wife, who has been taking nice care of and supporting me during the past two years, and KBB, my cat, who has always been the best company during the past seven years. I also want to memorize Silver, one of my best friends, whose unfortunate death led me to the local charities, where I had a chance to meet Steve and Patty Stampf, my sincerest friends, who have been sharing my ups and downs and kindly supporting my studies at Stony Brook and my local charity for animals at Manorville during the past few years.

These seven years I spent at Stony Brook have been an important part of my life. I want to thank Yufeng Wu, Xu Yan, Ji Cheng, and Peng Dai, who have been excellent friends and colleagues and have shared their ideas with me in the discussions.

This work was partially supported by NSF grants DMI-0600538, CMMI-0900206, and CMMI-0928460.

Chapter 1

Polynomial Classification Algorithms for Markov Decision Processes

1.1 Introduction

In the first chapter, we consider discrete-time Markov Decision Processes (MDPs) with finite state and action sets. The probability structure of an MDP is defined by a state space $S = \{1, \ldots, N\}$, finite sets of actions $A(i)$ for all $i \in S$, and transition probabilities $p(j|i,a)$, where $i, j \in S$ and $a \in A(i)$. A deterministic policy φ is defined as a function from S to $\bigcup_{i \in S} A(i)$ which assigns an action $\varphi(i) \in A(i)$ to each state $i \in S$. Each deterministic policy defines a stochastic matrix $P(\phi) = (p(j|i,\varphi(i))_{i,j=1,\ldots,N}$. The stochastic matrix defined by a deterministic policy is also known as a transition matrix of a homogeneous Markov chain. A transition matrix determines which states of the Markov chain are recurrent, transient, or equivalent under a deterministic policy.

A state $i \in S$ is called *transient* (*recurrent*) if it is transient (recurrent) under all deterministic policies. An MDP is called *multichain* if the transition matrix corresponding to at least one deterministic policy φ contains two or more recurrent classes. Otherwise, an MDP is called *unichain.* In other words, under any deterministic policy, the state space of a unichain MDP consists of only one recurrent class and a possible empty set of transient states.

The property of unichain is important for MDPs with average reward criterion because stronger results on the existence of optimal policies hold and better algorithms are available for unichain MDPs than for general MDPs; see [15] for detail. Unichain MDPs have been treated separately from general MDPs since Howard [12] introduced the policy iteration algorithms for MDPs; see e.g. [5, 6, 15, 22]. Kallenberg [14] studied irreducibility, communicating, weakly communicating, and unichain classification problems for MDPs. For the first three problems, Kallenberg [14] constructed polynomial algorithms. However, for the unichain classification problem, Kallenberg [14], [15, p. 41] posted a question whether a polynomial algorithm exists. Tsitsiklis [25] answered this question by showing that the unichain classification problem is NP-hard.

Even though the unichain classification problem for MDPs is NP-hard, many applications are modelled as unichain MDPs. Moreover, many applications of MDPs contain the states which are recurrent under all stationary policies. For instance, for a queueing or inventory control problem, a recurrent state is typically either the state when the buffer is empty or the state when the buffer is full. In this chapter we show that the problem of detecting whether an MDP has a recurrent state is polynomial. We also show that the unichain classification problem for an MDP with at least one recurrent state is polynomial. In this chapter, we call a state $i \in S$ *stopping* if $p(i|i,a) = 1$ for some $a \in A(i)$. The problem of detecting stopping states is polynomial and we also show that the unichain classification problem for an MDP with a stopping state is polynomial. We provide the polynomial algorithms and complexities to the corresponding problems.

Kallenberg [14] solved Some of classification problems in terms of graphs G^1 and G^2 whose arcs respectively represent that there are one-step transitions between two states under all actions and under some actions. According to the definition in [14], these graphs contain no loops. We slightly modify the definition of graph G^2 by adding loops (i,i) if and only if $i \in S$ is a stopping state.

An MDP is called deterministic if $p(j|i,a) \in \{0,1\}$ for all $i,j \in S$ and for all $a \in A(i)$. For deterministic MDPs, the unichain classification problem is equivalent to the question whether the corresponding graphs G^2 have two node-disjoint cycles. This problem has been proven to be polynomial by McCuaig [18] and, therefore, the unichain classification problem for deterministic MDPs is polynomial.

The first chapter is organized in the following. In Section 1.2, we show that the unichain classification problem cannot be solved in terms of the graphs G^1 and G^2. In Section 1.3, we introduce the definitions of avoidable and reachable sets and provide the corresponding polynomial algorithms which find the states from which a given set is avoidable and reachable. In section 1.4, we provide a polynomial algorithm that detects whether a state is recurrent and solves the unichain classification problem for an MDP with a recurrent state and a polynomial algorithm for detecting recurrent and stopping states and for the unichain classification problem with either recurrent or stopping states. Section 1.5 deals with detecting transient states in polynomial time and it discusses the implications of this capability for alleviating the complexity of the unichain classification problem. In Section 1.6, we conclude our results and discuss possible future research for the unichain classification problem.

1.2 Insufficiency of Graphs G^1 and G^2

Following Kallenberg [14], we define a directed graph G^1 to be a graph with the set of nodes S, no loops, and arcs (i,j), $i \neq j$, if and only if $\min\{p(j|i,a) |\, a \in A(i)\} > 0$. We also define a directed graph G^2 to be a graph with the set of nodes S such that:

i. an arc (i,j), $i \neq j$, belongs to G^2 if and only if $\max\{p(j|i,a) |\, a \in A(i)\} > 0$.

ii. a loop (i,i) belongs to G^2 if and only if $p(i|i,a) = 1$ for some $a \in A(i)$.

For a graph G, we also denote by G its incident matrix, i.e., $G(i,j) = 1$ if the arc (i,j) belongs to the graph and $G(i,j) = 0$ otherwise. The reason why we allow loops in graphs G^2 is because, as the following example shows, in the modified form the loops help us detect stopping states, while in the original form they do not.

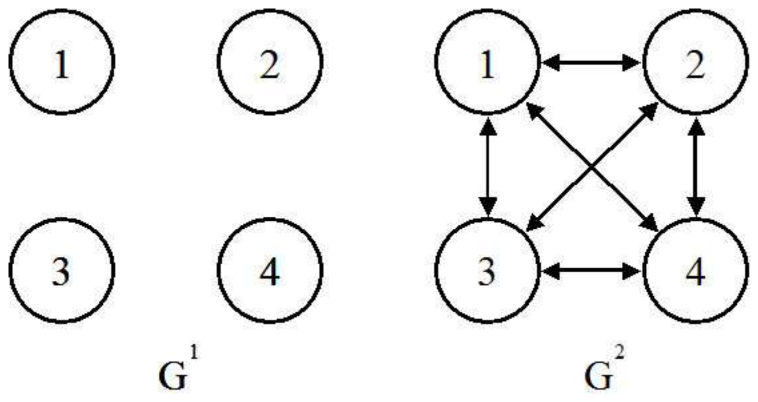

Figure 1.1: Graphs G^1 and G^2 in Example 1.2.

Example 1.1 Let $S = \{1, 2, 3\}$, $A(1) = \{a, b\}$, $A(2) = A(3) = \{a\}$, $p(2|1, a) = p(3|1, b) = p(3|2, a) = p(2|3, a) = 1$. Observe that state 1 is not stopping. If we add an action c to $A(1)$ with $p(1|1, c) = 1$ then state 1 becomes stopping. The graph G^1 does not change. If we follow the definition in [14] that $G^2(i, i)$ is always 0, the graph G^2 does not change either. According to the above definition, $G^2(1, 1)$ becomes equal to 1 and this detects that state 1 is stopping, if the action c is added.

The following example shows that a unichain MDP and a multichain MDP have the same G^1 and G^2 graphs.

Example 1.2 Let $S = \{1, 2, 3, 4\}$ and $A(i) = \{a, b, c\}$, $i = 1, 2, 3, 4$. The first MDP is deterministic. Each action moves the process to a different state, and there are no stopping states. For example, in state 1, the action a moves the process to state 2, the action b moves the process to state 3, and the action c moves the process to state 4. This MDP is multichain. Indeed, if from state 1 (3) the process moves to state 2 (4) and from state 2 (4) the process moves

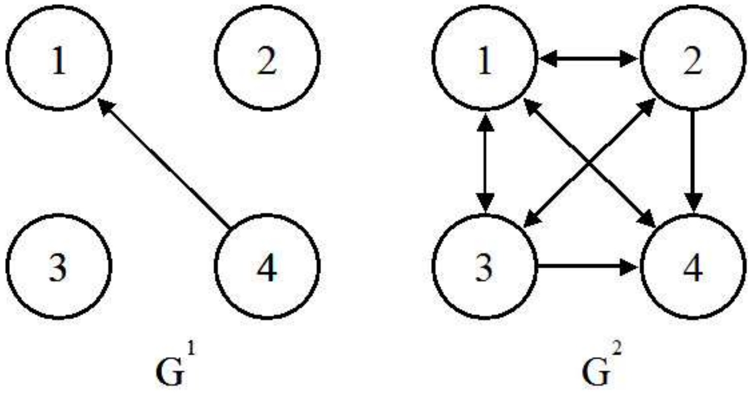

Figure 1.2: Graphs G^1 and G^2 in Example 1.3.

to state 1 (3), then there are two recurrent classes $\{1, 2\}$ and $\{3, 4\}$.

The second MDP has the same state and action sets as the first one. However, all three actions define different transition probability vectors. For each action, the probability to stay in the same state is 0 and the probability to move to each of two of remaining three states is 0.5. So, for state 1, we have $p(2|1, a) = p(3|1, a) = p(2|1, b) = p(4|1, b) = p(3|1, c) = p(4|1, c) = 0.5$. This MDP is unichain because the minimal possible number of states in a recurrent class is 3 and under all policies there are no absorbing states. For the both graphs, we have that $G^1 = 0$ and $G^2(i, j) = 1$ if and only if $i \neq j$.

In the following example, we provide two MDPs with identical corresponding graphs G^1 and G^2 such that one of these MDPs has no recurrent states and the other one has a recurrent state. Therefore, the information provided by graphs G^1 and G^2 is insufficient to detect whether a state is recurrent.

Example 1.3 Consider two MDPs with $S = \{1, 2, 3, 4\}$ and $A(i) = \{a, b, c\}$, $i = 1, 2, 3$, and $A(4) = \{a\}$. For both MDPs $p(1|4, a) = 1$. In states 1,2, and

3 the first MDP has the same transition probabilities as the first MDP in Example 1.2 and the second MDP has the same transition probabilities as the second MDP in Example 1.2. Since transition probabilities are the same at state 4, the corresponding graphs G^1 and G^2 coincide for these MDPs.

The first MDP does not have a recurrent state. Indeed, if we select actions in states 1,2, and 3 that move the process to state 4 then states 2 and 3 are transient and $\{1,4\}$ is a recurrent class. If we select in states 1 and 2 the actions that move the process to state 3 and in state 3 we select the action that moves the process to state 2 then 1 and 4 are transient states and $\{2,3\}$ is a recurrent class.

For the second MDP, state 1 is always recurrent. Indeed, for any deterministic policy any recurrent class contains at least three states. However, the process always moves from state 4 to state 1. Therefore, the set $\{2,3,4\}$ cannot be a recurrent class under any deterministic policy.

In the following example, we provide two MDPs with identical corresponding graphs G^1 and G^2 such that one of these MDPs has no transient states and the other one has transient states. Therefore, the information provided by graphs G^1 and G^2 is insufficient to detect whether a state is transient.

Example 1.4 Consider two MDPs with $S = \{1,2,3,4\}$ and $A(i) = \{a,b,c\}$, $i = 1,2$, and $A(3) = A(4) = \{a\}$. The first MDP is deterministic. From states 1 and 2 it is possible to move to any other state. States 3 and 4 are absorbing. So, for the first MDP $p(2|1,a) = p(3|1,b) = p(4|1,c) = p(1|2,a) = p(3|2,b) = p(4|2,c) = p(3|3,a) = p(4|4,a) = 1$. Consider a policy that always selects the action a. Then the Markov chain has three recurrent classes: $\{1,2\}$, $\{3\}$, and

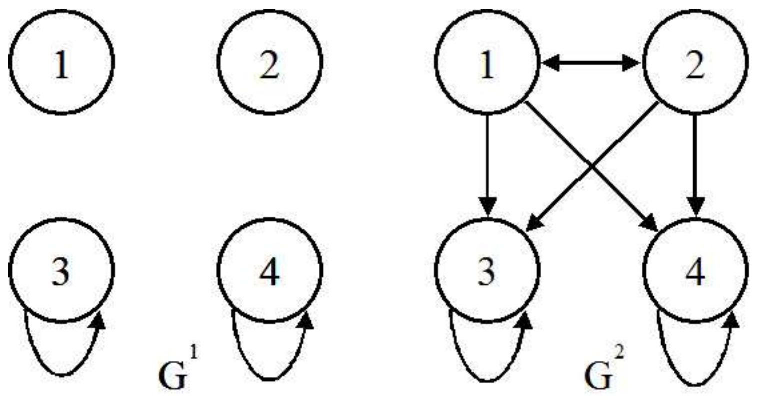

Figure 1.3: Graphs G^1 and G^2 in Example 1.4.

$\{4\}$. Thus, this MDP does not have a transient state.

The second MDP has the same state and action sets as the first one with $p(2|1,a) = p(3|1,a) = p(2|1,b) = p(4|1,b) = p(3|1,c) = p(4|1,c) = p(1|2,a) = p(3|2,a) = p(1|2,b) = p(4|2,b) = p(3|2,c) = p(4|2,c) = 0.5$ and $p(3|3,a) = p(4|4,a) = 1$. This MDP has two transient states 1 and 2. In the both cases $G^1 = 0$. In the both cases: (a) $G^2(i,j) = 1$ when $i = 1,2$ and $j \neq i$, (b) $G^2(3,3) = G^2(4,4) = 1$, and (c) $G^2(i,j) = 0$ for other i and j.

1.3 Avoidable and Reachable Sets

In this section, we define the avoidable and reachable sets and provide the polynomial algorithms that find the states from which a given set is avoidable and reachable respectively.

Definition 1.1 *Let $i \in S$ and $Y \subset S$. The set Y is called avoidable from i if there exists a deterministic policy φ such that $P_i^\varphi(x_t \in Y) = 0$ for all $t = 0, 1, \ldots$.*

A subset $Z \subseteq S$ is called *closed* under a deterministic policy φ if $p(i|j, \varphi(j)) = 0$ for any $j \in Z$ and for any $i \in S \setminus Z$. It is clear that a set Y is avoidable from i if and only if there exists $Z \subseteq S \setminus Y$ such that:

i. Z is closed under some deterministic policy and

ii. $i \in Z$.

For $Y \subseteq S$ we let $Z^A(Y)$ denote the set of states in S from which Y is avoidable. Algorithm 1 below finds the set $Z^A(Y)$ for $Y \subseteq S$. Its convergence is based on the necessary and sufficient condition formulated in the previous paragraph.

Algorithm 1 *Finding $Z^A(Y)$ for a given $Y \subseteq S$.*

1. Set $Z := Y$, $\tilde{Z} := Y$.

2. Do while $\tilde{Z} \neq \emptyset$: for $j \in S \setminus Z$ set

$$A(j) := A(j) \setminus \{a \in A(j) | \sum_{l \in \tilde{Z}} p(l|j, a) > 0\}, \tag{1.1}$$

   set $\tilde{Z} := \{j \in S \setminus Z : A(j) = \emptyset\}$, and set $Z := Z \cup \tilde{Z}$; end do.

3. Set $Z^A(Y) := S \setminus Z$. Stop.

The complexity of Algorithm 1 is $O(A \cdot N)$. Indeed, let $\tilde{Z}_t$, $t = 0, 1, \ldots, m$, be the set $\tilde{Z}$ at the t^{th} iteration of Step 2, where $\tilde{Z}_0 = Y$, $\tilde{Z}_m = \emptyset$, and $m \leq N - |Y| + 1$, where $|E|$ denotes the number of elements in the finite set E. Observe that $\tilde{Z}_t \cap \tilde{Z}_s = \emptyset$ for $t \neq s$ and $\cup_{t=0}^{m-1} \tilde{Z}_t \subseteq S \setminus Y$. The complexity of computations in (1.1) at t^{th} iteration is $O(A \cdot |\tilde{Z}_t|)$, $t = 0, \ldots, m-1$. This implies that the complexity of Algorithm 1 is $O(A \cdot N)$.

Note that there are some similarities between the definition of an avoidable set above and the node that should be avoided in the Optimal Node Visitation (ONV) problem in stochastic graphs studied by Bountourelis and Reveliotis [4]. In particular, Algorithm 1 uses the same node elimination procedure as the independently formulated algorithm [4, Figure 3] for the reduction of the ONV problem.

Definition 1.2 *Let $i \in S$, $Y \subset S$, and $i \notin Y$. The set Y is called reachable from i if there exists a deterministic policy φ such that $P_i^{\varphi}(x_t \in Y) > 0$ for some $t = 1, 2, \ldots$.*

Note that the definition of a reachable set is slightly different than the standard definition of an accessible set since the former requires $i \notin Y$ and only considers $t > 0$.

For $Y \subset S$, we denote by $Z^R(Y)$ the set of states in S from which Y is reachable. Finding $Z^R(Y)$ is equivalent to finding all the states from $S \setminus Y$ from which there is a path to Y in the graph G^2. The following algorithm finds the set $Z^R(Y)$ based on this concept.

Algorithm 2 *Finding $Z^R(Y)$ for a given $Y \subseteq S$.*

1. Construct the graph G^2. If Y is a singleton, let $Y = \{y\}$. If Y is not a singleton, reduce the set of nodes S by replacing the set Y with a single node $y \in Y$. Set $S^* := \{y\} \cup (S \setminus Y)$.

2. For all $i \in S \setminus Y$ set

$$G^2(i,y) := \begin{cases} 1 & if\ G^2(i,l) > 0\ for\ some\ l \in Y; \\ 0 & otherwise; \end{cases}$$

   and reverse all the arcs in the reduced graph G^2.

3. For the starting node y, apply the breadth-first search algorithm [1, p.73-76]. $Z^R(Y)$ is the set of nodes, except y, in the breadth-first search tree.

The complexity of constructing the graph G^2 is $O(A \cdot N)$; see Kallenberg [14]. The complexities of Steps 2 is $O(N^2) \leq O(A \cdot N)$. The complexity of the breadth-first search algorithm is $O(N^2)$ [1, p.73-76]. Thus, the complexity of Algorithm 2 is $O(A \cdot N)$.

1.4 The Polynomial Classification Algorithms

In this section, we use the concepts of avoidable and reachable sets to formulate a polynomial algorithm that detects whether a particular state i is recurrent. Moreover, if the state is recurrent, the polynomial algorithm also detects whether the MDP is unichain. Example 1.3 indicates that finding a recurrent state in an MDP cannot be done by using only matrices G^1 and G^2. Later in this section, we show our main result of the polynomial algorithm for detecting recurrent and stopping states and for the unichain classification problem with either recurrent or stopping states. We provide the complexities of both algorithms.

If Y contains only one state, $Y = \{i\}$, we shall write $Z^A(i)$ and $Z^R(i)$ instead of $Z^A(Y)$ and $Z^R(Y)$ respectively. We apply Algorithm 1 to state i and find the set $Z^A(i)$. If $Z^A(i) = \emptyset$, it is obvious that state i is a recurrent state and the MDP is unichain. If $Z^A(i) \neq \emptyset$, we apply Algorithm 2 to $Y = Z^A(i)$ and find the set $Z^R(Z^A(i))$. If $i \in Z^R(Z^A(i))$ then $Z^A(i)$ is reachable from state i and state i is avoidable from any $j \in Z^A(i)$. Therefore, state i is not recurrent and we do not know whether the MDP is unichain or multichain. On the other hand, if $i \notin Z^R(Z^A(i))$, then, starting from state i, the process will never reach $Z^A(i)$ and will travel only through the states from which state i is not avoidable. In this case, we know state i is a recurrent state and the MDP is multichain because there is a subset of $Z^A(i)$ which forms a recurrent class for a Markov chain defined by some deterministic policy. The following algorithm detects whether a state i is a recurrent state. If the state is a recurrent state, the algorithm also detects whether an MDP is unichain.

Algorithm 3 *Detecting whether a state i is recurrent and, if i is recurrent, whether the MDP is unichain.*

1. Apply Algorithm 1 to find $Z^A(i)$. If $Z^A(i) = \emptyset$ then the state i is a recurrent state and the MDP is unichain, and stop.

2. Apply Algorithm 2 to find $Z^R(Z^A(i))$. If $i \in Z^R(Z^A(i))$ then the state i is not a recurrent state. Else, the state i is a recurrent state and the MDP is multichain. Stop.

The complexity of Algorithm 3 is $O(A \cdot N)$ because Algorithms 1 and 2 have this complexity. We may have to apply Algorithm 3 to each state in order to detect if there exists a recurrent state in an MDP. This procedure leads to construct the set of recurrent states and its complexity is $O(A \cdot N^2)$. Repeating Algorithm 3 at most N times until a recurrent state is found also leads to the solution of the unichain classification problem for an MDP with a recurrent state. Thus, the complexity of this algorithm is $O(A \cdot N^2)$ too. In the following, we provide an algorithm for solving a unichain classification problem for an MDP with either a recurrent or stopping state.

If a state i is either a recurrent or stopping state then the MDP is unichain if and only if under all deterministic policies there is no recurrent class that does not contain state i. Let a state i be either stopping or recurrent. If $Z^A(i) = \emptyset$ then the state i is unavoidable from all other states. In this case, under any deterministic policy any recurrent class contains i. Therefore, the MDP is unichain. On the other hand, if $Z^A(i) \neq \emptyset$ then under some deterministic policy the corresponding Markov chain contains a recurrent class that does not contain the state i. Obviously, the MDP is multichain.

If an MDP contains more than one stopping state, it is multichain obviously. Even though there may be two or more recurrent states in the MDP, we only need to apply Algorithm 1 to one recurrent or stopping state in order to know whether the MDP is unichain. Thus, we can formulate the following algorithm.

Algorithm 4 *Polynomial Algorithm to Detect whether an MDP has a stopping or recurrent state and, if it does, whether an MDP is unichain.*

1. For $i = 1, \ldots, N$ and for $a \in A(i)$ check the condition $p(i,i) = 1$ until

two stopping states are found.

2. If two stopping states are found, the MDP is multichain and stop.

3. If one stopping state i is found then apply Algorithm 1 with $Y = \{i\}$ and

   (a) if $Z^A(i) \neq \emptyset$, the MDP is multichain and stop;

   (b) if $Z^A(i) = \emptyset$, the MDP is unichain and stop.

4. For $i = 1, \ldots, N$ apply Algorithm 3 as long as a recurrent state is not found. Stop after a recurrent state is found and the MDP is classified.

5. Conclude that the MDP contains neither stopping nor recurrent states and stop.

The complexity of Algorithm 4 is $O(A \cdot N^2)$ since it requires running Algorithm 3 at most N times.

1.5 Finding Transient States and Remarks

Let T be the set of transient states. This set can be found by apply Bather's decomposition algorithm [2] This algorithm is formalized in [14, Algorithm 7] and its complexity is $O(A \cdot N^2)$ [14]. In terms of [14, Algorithm 7], the set of transient states T is the union of the sets $T_1, \ldots, T_m$ computed by that algorithm.

After the set of transient states T is found, we can delete T from the state space S and reduce the action sets $A(j)$, $j \in S \setminus T$, to

$$A(j) := A(j) \setminus \{a \in A(j) : \sum_{i \in T} p(i|j,a) > 0\}. \tag{1.2}$$

Any deterministic policy φ in the original MDP defines a deterministic policy in the reduced MDP as a function on $S \setminus T$. Since the states in T are always transient in the original MDP, the recurrent classes for these two Markov chains coincide. Thus, it is easy to see that the original MDP is unichain if and only if the reduced MDP is unichain. Thus, if $T \neq \emptyset$, by removing the set T and reducing the actions, we can reduce the unichan classification problem to a smaller problem.

An MDP is called *communicating* if for each two states $i, j \in S$ there exists a deterministic policy φ, which may depend on i and j, such that j accessible from i in the Markov chain defined by φ. An MDP is called *weakly communicating* if, after the set T is deleted and the action sets in $E := S \setminus T$ are reduced following (1.2), the MDP with the state space E is communicating. If an MDP is not weakly communicating, it is multichain. This follows from Bather's [2] decomposition.

Algorithm 4 in [14] detects whether an MDP is weakly communicating and its complexity is $O(A \cdot N^2)$. If an MDP is weakly communicating, it can be reduced in polynomial $O(A \cdot N^2)$ time to a communicating MDP; see (1.2). Thus, the unichain classification problem for a weakly communicating MDP can be reduced in polynomial $(O(A \cdot N^2))$ time to an NP-hard unichain classification problem for a communicating MDP.

Algorithm 4 solves the unichain classification problem for MDPs with recurrent and stopping states. Algorithm 5 in Kallenberg [14] also solves the unichain classification problem for some MDPs. Both algorithms have complexity $O(A \cdot N^2)$. [14, Algorithm 5] finds strongly connected components (maximal connected subsets) of the graph G^1. Then it compresses G^1 by replacing each strongly connected component in G^1 with a single node. In the compressed graph, there exists an arc (i^*, j^*) if in the strongly connected component corresponding to i^* there is a state i such that $\sum_{j \in X} p(j|i,a) > 0$ for all $a \in A(i)$, where X is the strongly connected component compressed into j^*. Then [14, Algorithm 5] conducts additional compressions by merging nodes i^* with the nodes j^* if the arc (i^*, j^*) exists and i^* is with outgoing rank 1. These procedures are repeated recursively until the graph cannot be compressed anymore. Let $(G^1)^+$ be the graph that is eventually obtained and cannot be compressed and k^+ be the number of strongly connected components in $(G^1)^+$. If $k^+ = 1$ then the MDP is unichain, if $k^+ = 2$ then the MDP is multiichain, and if $k^+ = 3$ then the MDP is either unichain or multichain; [14, Theorem 3.6].

At the end of this section, we give two examples to show that Algorithm 4 in this paper and [14, Algorithm 5] solve different classes of problems. Of course, these two classes overlap. Algorithm 4 always classifies an MDP with a recurrent state. It is clear that, if $k^+ = 1$, [14, Algorithm 5] compresses the graph around a recurrent state. Thus, if [14, Algorithm 5] detects that the MDP is unichain, this MDP has a recurrent state. Example 1.5 provides a unichain MDP with a recurrent states and this MDP cannot be classified by [14, Algorithm 5]. Example 1.6 shows that [14, Algorithm 5] can classify some MDPs without recurrent states.

Example 1.5 Consider an MDP with the same state and action sets as in the second MDP in Example 1.3. In states 1,2, and 3, the transition probabilities are the same as in the second MDP in Example 1.3. In addition, $p(1|4,a) = p(2|4,a) = p(3|4,a) = \frac{1}{3}$. In this MDP, states 1,2, and 3 are recurrent. For this MDP, $G^1(4,j) = 1$, $j = 1,2,3$, and $G^1(i,j) = 0$ in all other cases. This graph cannot be compressed and therefore $k^+ = 4$.

Example 1.6 Let $S = \{1,2,3,4\}$, $A(1) = A(3) = \{a\}$, and $A(2) = A(4) = \{a,b\}$. In addition, $p(2|1,a) = p(1|2,a) = p(4|3,a) = p(3|4,a) = 1$ and $p(1|2,b) = p(3|2,b) = p(3|4,b) = p(1|4,b) = \frac{1}{2}$. This MDP has no recurrent states. The graph G^1 has two strongly connected components $\{1,2\}$ and $\{3,4\}$ and they contract to a graph consisting of two isolated nodes. Thus $k^+ = 2$ and [14, Algorithm 5] detects that this MDP is multichain.

1.6 Conclusion

In this chapter, we studied the unichain classification problem for the MDPs which contain recurrent or stopping states. We introduced the definitions of avoidable and reachable sets and provided the polynomial algorithms to find the avoidable and reachable sets for a given set. We applied the ideas of avoidable and reachable sets to detect whether a given state in an MDP is recurrent or not and, if it is recurrent, whether the MDP is unichain. Since detecting stopping states in an MDP is polynomial, we provided the polynomial algorithm to solve the unichain classification problem for the MDPs with recurrent or stopping states.

There are still many spacial cases of MDPs where the unichain classification problem is polynomial. For example, it is interesting to look into the unichain classification problem for the MDPs without recurrent and stopping states. In the future, it is possible to construct the polynomial algorithm of solving the unichain classification problem for more general MDPs.

Chapter 2

Optimal Trunk Reservation for an M/M/k/N controlled queue with holding costs

2.1 Introduction

In this chapter, we consider an M/M/k/N controlled queueing system with m customer types, where $m \geq 1$. Customers of type j arrive at the system according to an independent Poisson process with rate λ_j, $j = 1, 2, \ldots, m$, where $0 < \lambda_j < \infty$. When a customer arrives at the system, its type becomes known. There are k identical servers in the system, where $k \geq 1$. The service times are independent, do not depend on the customer types, and are exponentially distributed with rate μ, where $0 < \mu < \infty$. When there are n customers in the system, the total service rate of the system is μ_n, where $\mu_n = n\mu$ for $n = 0, 1, \ldots, k-1$, and $\mu_n = k\mu$ for $n = k, k+1, \ldots, N$. Moreover, there is no preemption for customers. The queue follows the first-in-first-out (FIFO) rule.

At the arrival epochs, the system manager decides whether an arrival can enter the system or not. If a customer sees less than k customers in the system and is admitted, the customer goes to a free server immediately. If a customer sees at least k customers in the system and is admitted, the customer waits in the queue for service. If there are N customers in the system, the system is full and all the arrivals are rejected. Upon admitting a customer, the system collects a positive reward which depends on the customer type and incurs a nonnegative random holding cost which depends on the customer type and the number of customers in the system the admitted customer sees. Let $r_j(n)$ be the net reward collected by the system if a customer of type j sees n customers in the system and is admitted. The net rewards $r_j(n)$ are positive constants for $n = 0, 1, \dots, k-1$, and are nonincreasing in $n = k-1, k, \dots, N-1$. The dependence of $r_j(n)$ on n reflects the fact that net rewards depend on waiting times. Furthermore, we assume that different customer types have different reward functions . When an arrival is rejected, the system does not collect any reward. The objective is to maximize the long-run average reward. We call a policy, that maximizes the long-run average reward, an optimal policy. In addition to average reward optimality, we also consider three more selective policies: canonical, bias optimal, and Blackwell optimal policies. Below is the definition of a trunk reservation policy.

Definition 2.1 *A policy ϕ is called a trunk reservation policy (TRP) if there are m control levels M_j^{ϕ}, $j = 1, \dots, m$, such that a type j arrival is admitted* *to the system if and only if the customer sees less than M_j^ϕ customers in the system and a type j arrival is rejected if the customer sees at least M_j^ϕ customers in the system.*

This definition is weaker than the definition of a TRP in Feinberg and Reiman [9], which was introduced in Miller [19] without using the term "trunk reservation." In [9, 19], it is required that $M_l^\phi = N$ if $r_l = \max_{j=1,2,\ldots,m} r_j$, where only the case $r_j(n) = r_j$ was considered. The stronger definition is possible when there are no holding costs in [9, 17, 19].

Optimal admission control problem has been extensively studied in the literatures, but the results are limited to the models without holding costs or with holding costs that do not depend on customer types. Miller [19] considered m types of customers for an M/M/k/loss queue without holding costs and ordered rewards $r_1 > r_2 > \cdots > r_m > 0$. He proved the existence of an optimal TRP f with $k = M_1^f \geq M_2^f \geq \cdots \geq M_{m-1}^f \geq M_m^f$. Feinberg and Reiman [9] considered general birth and death processes and studied a problem with a constraint. In particular, they extended Miller's [19] result to an M/M/k/N queue without holding costs. In addition, Feinberg and Reiman [9] showed that any randomized stationary optimal policy for a problem without holding costs is a randomized TRP. Lewis et al. [17] provided a simpler proof of this fact for (nonrandomized) stationary policies. Problems with more general constraints than in [9] were studied by Fan-Orzechowski and Feinberg [7, 8].

Several optimal admission control problems with holding costs have been studied. Naor [21] proved the existence of an optimal TRP for an M/M/1 queue with one customer type and linear holding costs. Knudsen [16] extended

Naor's result to an M/M/k queue. Yechiali [27, 28] extended Naor's and Knudsen's results to GI/M/1 and GI/M/k queues with one customer type and linear holding costs. Stidham [23] proved the existence of an optimal TRP for a GI/M/1 queue with one customer type and convex nondecreasing holding costs; see also [24]. Johansen [13] extended Stidham's result to a GI/M/k queue. Helm and Waldman [11] extended Stidham's [23] and Johansen's [13] results to GI/M/k queues with batch arrivals, convex nondecreasing holding costs, and changing environment.

For a Markov Decision Process (MDP) with finite state and action sets, the optimal policies found by Howard's policy iteration algorithm satisfy optimality equations and such policies are called canonical. Blackwell [3] proved the existence of a stationary policy that optimizes the expected total discounted reward for all discount factors $\beta \in [\beta^*, 1)$ for some $\beta^* \in [0, 1)$. Such policies are called Blackwell optimal. A policy is called bias optimal if the difference of the expected total discounted rewards between this policy and a Blackwell optimal policy tends to 0 as $\beta \to 1$. Veinott [26] modified Howard's [12] policy iteration algorithm to find a bias optimal policy. For continuous time problems, the values of the discount factor β close to 1 correspond to the values of the discount rate α close to 0, because essentially $\beta = e^{-\alpha}$.

For an M/M/1 queue with one customer type and convex increasing holding costs, Haviv and Puterman [10] showed that there is either one optimal TRP or two optimal TRPs. If there are two optimal TRPs, the difference between their control levels is 1. Haviv and Puterman [10] also showed that the optimal TRP with the larger optimal control level is the unique bias optimal policy and, therefore, it is also Blackwell optimal. Lewis et al. [17] proved that a similar result holds for an M/M/k/N queue with several customer types and without holding costs.

We organize this chapter as follows. In Section 2.2, we formulate the net reward functions $r_j(n)$, for $j = 1, \ldots, m$ and $n = 0, 1, \ldots, N$, and then we follow Miller's approach to formulate the Continuous-Time Markov Decision Process (CMDP) for our problem. In Section 2.3, we prove the existence of an optimal TRP for our problem. In addition, we provide complete descriptions of the classes of stationary optimal policies, optimal TRPs, canonical policies. In Section 2.4, we show that the bias optimal policy for our problem is unique and it is also Blackwell optimal. Similar to the cases of one customer type [10] and several customer types without holding costs [17], the bias optimal policy selects the largest optimal control level for each customer type. For all customer types, except at most one, there exist at most two optimal control levels. Under a certain condition, there is one customer type, for whom there are more than two optimal control levels. For this customer type, the largest optimal control level is N. In Section 2.5, we conclude our results and discuss the possible future research for queueing systems.

2.2 Problem Formulation

Consider the M/M/k/N queue defined in the introduction. Following the definitions, we formulate $r_j(n)$ and the continuous-time Markov decision process (CTMDP) for our problem. Let $X(t) = 0, 1, \ldots, N-k$ be the position of a customer in the queue at time t after its arrival. In particular, if the customer goes to a server, then $X(t) = 0$, and, if the customer is in the queue and there are $n-1$ customers in front in the queue, then $X(t) = n$. Observe that $X(t)$ does not depend on future arrivals and decisions. Moreover, let R_j be the reward an admitted type j customer pays for the service and $h_j(X(t), t) \geq 0$ be the rate the system incurs holding cost to the admitted type j customer when the position of this customer is $X(t)$ at time t.

Also, we let D_n, $n = 0, 1, \ldots, N-1$, be the delay for an admitted customer seeing n customers in the system at the arrival time, and W_n be the waiting time for such a customer. In other words, if a customer is admitted when there are n customers in the system, the admitted customer spends W_n units of time in the system and D_n units of time in the queue waiting until the service begins.

If type j arrival is admitted when there are n customers in the system, the expected holding cost $H_j^d(n)$ incurred during the time the customer waits for service in the queue is

$$H_j^d(n) = E \int_0^{D_n} h_j(X(t), t)dt,$$

and the expected holding cost $H_j^s(n)$ incurred during the time the customer spends in the system is

$$H_j^s(n) = E \int_0^{W_n} h_j(X(t), t)dt.$$

In particular, Haviv and Puterman [10] and Stidham [23] considered that the holding cost per unit time at time t depends only on the number of customers in the system for M/M/1 and GI/M/1 systems respectively. Let c_n

be the holding cost per unit time if there are n customers in the system. We define $c_{-1} = 0$ and $\triangle c_n = c_n - c_{n-1}$. Then, by selecting $h_j(x,t) = \triangle c_x$, we have that the holding cost per unit time for our problem is $\sum_{i=0}^{n} h_j(i,t) = c_n$.

The reward functions $r_j(n) = R_j - H_j(n)$, $n = 0, 1, \ldots, N-1$, where either $H_j(n) = H_j^d(n)$ or $H_j(n) = H_j^s(n)$ depending on a particular problem. In the either case, the functions $r_j(n)$ possess some natural properties. For example, $r_j(n) = r_j(0)$ for $n = 1, \ldots, k-1$. If $r_j(n) = R_j - H_j^d(n)$, then $r_j(0) = R_j$. If $r_j(n) = R_j - H_j^s(n)$, then $r_j(0) = R_j - H_j^s(0)$. In either case, it is natural to assume that $r_j(0) > 0$. Otherwise, type j customers should be always rejected. If $h_j(t) = 0$ for all $t \geq 0$, then $r_j(n) = r_j(0) = R_j$ for all n. If h_j are measurable functions such that $a \leq h_j(t) \leq b$ for two finite positive constants a and b, then $\frac{a}{k\mu} \leq r_j(n-1) - r_j(n) \leq \frac{b}{k\mu}$ for $n \geq k$. We assume the following broad and natural condition throughout this paper.

Condition 2.1 *For each $j = 1, 2, \ldots, m$, the reward function $r_j(n)$ satisfies the following conditions: (a) $r_j(n) = r_j(0) > 0$ for all $n = 0, 1, \ldots, k-1$, (b) either $r_j(n) = r_j(0)$ for all $n = k, k+1, \ldots, N-1$ or $r_j(n) < r_j(n-1)$ for all $n = k, k+1, \ldots, N-1$, and (c) different customer types have different reward functions, that is, if $j \neq l$, then $r_j(n) \neq r_l(n)$ for some $n = 0, 1, \ldots,$ where $j, l = 1, \ldots, m$.*

We formulate our problem as a CTMDP with the state space $S = \{0, 1, \ldots N\}$. State n means there are n customers in the system. $A(n)$ is the set of actions available at state n and $A = \bigcup_{n=0}^{N} A(n)$. An action $a \in A(n)$ is the set of the customer types to be admitted at state n and $a = \emptyset$ means no customer is accepted. Thus, $A(n)$, $n = 0, 1, \ldots, N-1$, contains 2^m actions, which are all the possible subsets of $\{1, 2, \ldots, m\}$, and $A(N) = \{\emptyset\}$. Each action $a \in A(n)$ defines a transition intensity $\lambda_n^a = \sum_{j\in a} \lambda_j$ and an expected reward per unit time $R(n, a) = \sum_{j\in a} \lambda_j r_j(n)$. Since S and A are finite, there exists a (nonrandomized) stationary optimal policy for our problem. Thus, in the rest of this paper, we focus on the set of all stationary policies and let F be this set. For any $f \in F$, we denote by f_n the action that f chooses from $A(n)$ at state n. We also define $f(n, j)$ as follows:

$$f(n, j) = \begin{cases} \{accept\}, & \text{if } j \in f_n; \\ \{reject\}, & \text{otherwise.} \end{cases}$$

Moreover, the policy f defines a vector of expected reward per unit time $R(f)$ whose $(n+1)$-st component is $R(n, f_n)$ and an infinitesimal matrix $Q(f)$ whose (n, l) element is $q(l|n, f_n)$, where $q(n + 1|n, f_n) = \lambda_n^f$ for $n = 0, 1, \ldots, N - 1$, $q(n - 1|n, f_n) = \mu_n$ for $n = 1, 2, \ldots, N$, $q(n|n, f_n) = -\mu_n - \lambda_n^f$ for $n = 0, 1, \ldots, N$, and $q(l|n, f_n) = 0$, otherwise. Furthermore, let $P_{nl}(t, f)$ be the probability that the process is at state l at time t, given that the process started at state n at time 0 under policy f. We let $P(t, f)$ be the corresponding matrix. Then $P(t, f)$ converges to the steady probability matrix $P^*(f)$ as $t \to \infty$. We denote v_n^f the average reward per unit time, given that the process starts at state n under f, and V^f the vector whose $(n + 1)$-st element is v_n^f, $n = 0, 1, \ldots, N$. Then

$$v_n^f = \liminf_{T\to\infty} \; T^{-1} E_n^f \int_0^T R(x_t, a_t) dt,$$

where $R(x_t, a_t)$ is the expected reward per unit time collected when the process is at state x_t and action a_t is chosen at time t, and

$$V^f = \lim_{T \to \infty} T^{-1} \int_0^T P(t,f)R(f)dt = P^*(f)R(f).$$

Our goal is to find an optimal policy that maximizes the vector V^f. Note that there is only one recurrent class under any policy from F. Such CTMDPs are called unichain. Since the CTMDP is unichain, all rows of $P^*(f)$ are equal to the unique steady state probability vector $(\pi_0^*(f), \pi_1^*(f), \ldots, \pi_N^*(f))$. Thus, all elements of V^f are equal and $v_n^f = v^f = \sum_{s=0}^{N-1} \pi_s^*(f)R(s, f_s)$, $n = 0, 1, \ldots, N$. A policy $f \in F$ is optimal if $v^f \geq v^\phi$ for all $\phi \in F$. Let F^* be the set of stationary optimal policies and v^* be the average reward per unit time under any policy from F^*. For a unichain CTMDP, a policy f is called canonical if and only if there exists a function y_j^f such that

$$v^f = R(i, f_i) + \sum_{j=0}^{N} q(j|i, f_i)y_j^f = \max_{a \in A(i)} \{R(i,a) + \sum_{j=0}^{N} q(j|i,a)y_j^f\} \tag{2.1}$$

for $i = 0, 1, \ldots, N$. For a unichain CTMDP with finite states and action sets, a canonical policy always exists, is optimal, and Howard's policy iteration algorithm computes it; see [29]. Similar to discrete-time MDPs, an optimal policy may not be canonical.

2.3 Average Reward Optimality

The results of Feinberg and Reiman [9, Theorem 3.1] imply that, for any M/M/k/N queue with m customer types and without holding costs, any stationary optimal policy is a TRP; see also [17] for a simpler proof of this fact. In addition, Feinberg and Reiman [9] showed that, under any optimal policy f, if $R_l > R_k$, then $M_l^f \geq M_k^f$, and, if $R_i = \max_{j=1,\dots,m} R_j$, then $M_i^f = N$. In this section, we prove the existence of an optimal TRP for the problem with holding costs and obtain the structure of the canonical policies. Contrary to the results for problems without holding costs [9, 17], a stationary optimal policy may not be a TRP; see Examples 2.1 and 2.2. However, under some conditions, we show that every stationary optimal policy has a closely related form which we call essential trunk reservation. Moreover, because $r_j(n)$ depend on the customer types and the states, we show in Example 2.1 that $r_i(0) > r_j(0)$ does not imply $M_i^f \geq M_j^f$ under a stationary optimal TRP f.

In order to compute a stationary optimal policy for our problem, we consider the following equations for a CTMDP [19]:

$$V^f = P^*(f) \times R(f), \tag{2.2}$$

$$R(f) + Q(f) \times y^f = V^f, \tag{2.3}$$

$$P^*(f) \times y^f = 0, \tag{2.4}$$

where y^f is the bias vector and $f \in F$. According to [19],

$$y^f = \int_{t=0}^{\infty} (P(t,f) - P^*(f)) R(f) \, dt.$$

We denote by y_n^f, $n = 0, 1, \ldots, N$, the $(n+1)$-st element of y^f. Let $\nabla y_n^f = y_n^f - y_{n+1}^f$ for $n = 0, 1, \ldots, N-1$.In addition, for any $f \in F$ and $n = 0, 1, \ldots, N$, we define

$$H^f(n, z) = \sum_{j \in f_n} \lambda_j (r_j(n) - z),$$

where $H^f(n, z) = 0$ if $f_n = \emptyset$. In particular, $H^f(N, z) = 0$. Also, we let $\nabla y_{-1}^f = 0$ for any $f \in F$.

Miller [19] presented a version of Howard's policy iteration algorithm [12] for the optimal admission problem for any M/M/k/loss queue with several customer types. In that version, Miller [19] considered relative bias functions ∇y_n^f instead of bias functions y_n^f. In order to compute relative bias functions ∇y_n^f, he transformed (2.3) to the following formula:

$$H^f(n, \nabla y_n^f) + \mu_n \times \nabla y_{n-1}^f = v^f, \tag{2.5}$$

, for $n = 0, 1, \ldots, N$. The same transformation can be done for birth-and-death processes and reward functions depending on the states. Below we present a modification of Miller's algorithm [19] for an M/M/k/N queue with reward functions depending on states and customer types.

Algorithm 5 *Policy Iteration Algorithm*

1. Choose a policy $f \in F$.

2. For policy f, compute v^f and ∇y_n^f, $n = 0, 1, \ldots, N-1$, by solving linear

equations

$$H^f(n, \nabla y_n^f) + \mu_n \times \nabla y_{n-1}^f = v^f, \qquad n = 0, 1, 2, \ldots, N.$$

3. a. If $H^f(n, \nabla y_n^f) = \max_{a \in A(n)}\{\sum_{j\in a} \lambda_j(r_j(n) - \nabla y_n^f)\}$ for $n = 0, 1, \ldots, N-1$, then f is an optimal policy. Stop.

   b. For all $n = 0, 1, \ldots, N-1$ such that $H^f(n, \nabla y_n^f) < \max_{a\in A(n)}\{\sum_{j\in a} \lambda_j(r_j(n) - \nabla y_n^f)\}$, change f_n to a^n such that $\sum_{j\in a^n} \lambda_j(r_j(n) - \nabla y_n^f) = \max_{a\in A(n)}\{\sum_{j\in a} \lambda_j(r_j(n) - \nabla y_n^f)\}$. Go to Step 2.

Since the state space and the action sets of the CTMDP for our problem are finite, according to [29], a canonical policy always exists and the policy iteration algorithm computes it. The following statements are the descriptions for a canonical policy.

Proposition 2.1 *A policy f is canonical if and only if one of the following equivalent statements hold:*

(a) $H^f(n, \nabla y_n^f) = \max_{a\in A(n)}\{\sum_{j\in a} \lambda_j(r_j(n) - \nabla y_n^f)\}$ *for all* $n = 0, 1, \ldots, N-1$*;*

(b) $H^f(n, \nabla y_n^f) = \sum_{j=1}^{m} \lambda_j(r_j(n) - \nabla y_n^f)^+$ *for all* $n = 0, 1, \ldots, N-1$*;*

(c) $f(n, j) = \{accept\}$ *if* $r_j(n) > \nabla y_n^f$ *and* $f(n, j) = \{reject\}$ *if* $r_j(n) < \nabla y_n^f$.

Proof (a)–(b) According to [19], (2.5) is the same as (2.3). Thus, from (2.1), a policy f is canonical if and only if

$$v^* = H^f(n, \nabla y_n^f) + \mu_n \nabla y_{n-1}^f = \max_{a \in A(n)} \{\sum_{j \in a} \lambda_j (r_j(n) - \nabla y_n^f) + \mu_n \nabla y_{n-1}^f\}$$
$$= \max_{a \in A(n)} \{\sum_{j \in a} \lambda_j (r_j(n) - \nabla y_n^f)\} + \mu_n \nabla y_{n-1}^f = \sum_{j=1}^{m} \lambda_j (r_j(n) - \nabla y_n^f)^+ + \mu_n \nabla y_{n-1}^f.$$

Therefore, f is canonical if and only if $H^f(n, \nabla y_n^f) = \max_{a \in A(n)}\{\sum_{j \in a} \lambda_j (r_j(n) - \nabla y_n^f)\} = \sum_{j=1}^{m} \lambda_j (r_j(n) - \nabla y_n^f)^+$ for $n = 0, 1, \ldots, N-1$.

(c) According to (b), if $r_j(n) - \nabla y_n^f > 0$, then $f(n, j) = \{accept\}$. On the other hand, if $r_j(n) < \nabla y_n^f$, then $f(n, j) = \{reject\}$. ■

Let $n^f = \min\{n = 1, \ldots, N : \ f_n = \emptyset\}$, where $f \in F$. Then, under policy f, the recurrent class $R^f = \{0, 1, \ldots, n^f\}$ and the set of transient states $T^f = \{n^f + 1, n^f + 2, \ldots, N\}$. This is true because of the following reasons:

(i) state 0 is accessible from any state $n = 0, 1, \ldots, N$, and, therefore, it is recurrent;

(ii) any two states in R^f communicate;

(iii) any state in T^f is not accessible from state n^f.

Note that $T^f = \emptyset$ if and only if $n^f = N$. For problems without holding costs, $n^f = N$ if f is an optimal policy. [9, 17].

Theorem 2.1 *Every $f \in F^*$ always accepts type l customers with $r_l(0) = \max_{j=1,2,\ldots,m} r_j(0)$ when there are less than k customers in the system. Thus, $n^f \geq k$ for $f \in F^*$.*

Proof Assume $f \in F^*$ reject all customer types at state n, $n < k$. Then f is also optimal for an M/M/k/loss queue. However, according to [9, 17, 19], f always accepts type l customers with $r_l(0) = \max_{j=1,2,\dots,m} r_j(0)$ if there is at least one empty server. This contradiction implies this theorem. ■

For the problem without holding costs, all stationary optimal policies are TRPs [9, 17] and, thus, all canonical policies are TRPs. In the case with holding costs, $r_j(n)$ may decrease with respect to n for $n \geq k-1$, and it is possible that all customer types shall be rejected, even if there is available space in the system. Thus, the corresponding Markov chain may contain transient states. If a stationary optimal TRP is changed at these states, then it remains optimal, but it is not a TRP anymore. The following example illustrates this possibility.

Example 2.1 Consider an M/M/1/4 queue with two customer types, where $\lambda_1 = \lambda_2 = \mu = 1$, $r_1(0) = 10$, $r_1(1) = 6$, $r_1(2) = 2$, $r_1(3) = -2$, $r_2(0) = 9$, $r_2(1) = 8$, $r_2(2) = 6$, and $r_2(3) = 3$. Consider a policy $d \in F$ with $d_0 = \{1,2\}$, $d_1 = \{2\}$, $d_2 = \emptyset$, $d_3 = \{2\}$, and $d_4 = \emptyset$. By applying (2.5) with policy d, we have

$$\begin{aligned}
(10 - \nabla y_0^d) + (9 - \nabla y_0^d) &= v^d; \\
(8 - \nabla y_1^d) + \nabla y_0^d &= v^d; \\
\nabla y_1^d &= v^d; \\
(3 - \nabla y_3^d) + \nabla y_2^d &= v^d; \\
\nabla y_3^d &= v^d.
\end{aligned}$$

		State 0	State 1	State 2	State 3	State 4
Policy d	Type 1	A	R	R	R	R
	Type 2	A	A	R	A	R
Policy f	Type 1	A	R	R	R	R
	Type 2	A	A	R	R	R

Table 2.1: Policies d and f for Example 2.1 are optimal. A=Accept; R=Reject.

After solving the linear equations above, we have $v^d = \nabla y_3^d = \nabla y_1^d = 7$, $\nabla y_2^d = 11$, and $\nabla y_0^d = 6$. Since $r_2(3) = 3 < 7 = \nabla y_3^d$ and $d_3 = \{2\}$, after one policy iteration, we have another policy $f \in F$, with $f_0 = \{1, 2\}$, $f_1 = \{2\}$, and $f_2 = f_3 = f_4 = \emptyset$. By applying (2.5) with policy f, we have

$$(10 - \nabla y_0^f) + (9 - \nabla y_0^f) = v^f;$$
$$(8 - \nabla y_1^f) + \nabla y_0^f = v^f;$$
$$\nabla y_1^f = \nabla y_2^f = \nabla y_3^f = v^f.$$

After solving the linear equations, we have $v^f = \nabla y_1^f = \nabla y_2^f = \nabla y_3^f = 7$ and $\nabla y_0^f = 6$. Since $H^f(n, \nabla y_n^f) = \sum_{j=1}^m \lambda_j (r_j(n) - \nabla y_n^f)^+$ for $n = 0, 1, \ldots, 4$, policy f is a canonical policy. Since $v^d = v^f$, both d and f are optimal. However, f is a canonical policy and a TRP, but d is neither a canonical policy nor a TRP. Note that, under policy f, the control levels $M_2^f > M_1^f$, even though $r_1(0) = 10 > 9 = r_2(0)$.

The policies from Example 2.1 satisfy the properties described in the following lemma.

Lemma 2.1 *Let $f \in F^*$. Then the following statements hold for $j = 1, 2, \ldots, m$ and $n < N$.*

(i) *if* $r_j(n) > \nabla y_n^f$ *and* $n \in R^f$, *then* $f(n,j) = \{accept\}$;

(ii) *if* $r_j(n) < \nabla y_n^f$ *and* $n \in R^f$, *then* $f(n,j) = \{reject\}$;

(iii) *if* $g(n,j) = f(n,j)$ *for all* $n = 0, 1, \ldots, N-1$ *such that* $r_j(n) \neq \nabla y_n^f$, *then* $\nabla y_n^g = \nabla y_n^f$ *for all* $n = 0, 1, \ldots, N-1$ *and* $g \in F^*$;

(iv) $H^f(n, \nabla y_n^f) = \max_{a \in A(n)}\{\sum_{j \in a} \lambda_j (r_j(n) - \nabla y_n^f)\} \geq 0$ *for all* $n \in R^f$.

Proof For $f, d \in F$, let $e_n = H^d(n, \nabla y_n^f) - H^f(n, \nabla y_n^f)$. Then $e_n = H^d(n, \nabla y_n^d) - H^f(n, \nabla y_n^f) + \lambda_n^d(\nabla y_n^d - \nabla y_n^f)$. Equations (2.5) imply $v^d - v^f = e_n + \mu_n(\nabla y_{n-1}^d - \nabla y_{n-1}^f) - \lambda_n^d(\nabla y_n^d - \nabla y_n^f)$. By multiplying both sides with $\pi_n^*(d)$, summing them in n and using the fact that $\lambda_n^d \pi_n^*(d) = \mu_{n+1}\pi_{n+1}^*(d)$, we have

$$\sum_{n=0}^{N} \pi_n^*(d)(v^d - v^f) = v^d - v^f = \sum_{n=0}^{N} \pi_n^*(d) e_n. \tag{2.6}$$

(2.6) is a particular form of formula (8.39) in [**?**] and we use it to prove (i)–(iii).

(i) Assume that $r_j(n) > \nabla y_n^f$ and $f(n,j) = \{reject\}$ for some $n \in R^f$. Then consider a policy g such that $g_s = f_s$ for $s \neq n$, and $g_n = f_n \bigcup \{j\}$. Observe that $n \in R^g$ and, therefore, $\pi_n^*(g) > 0$. Then, from (2.6)

$$v^g - v^f = \pi_n^*(g)\lambda_j(r_j(n) - \nabla y_n^f).$$

Since $\pi_n^*(g) > 0$, $\lambda_j > 0$, and $r_j(n) - \nabla y_n^f > 0$, we have $v^g - v^f > 0$. Thus, $f \notin F^*$.

(ii) Assume that $r_j(n) < \nabla y_n^f$ and $f(n,j) = \{accept\}$ for some $n \in R^f$.

Then consider a policy g such that $g_s = f_s$ for $s \neq n$, and $g_n = f_n \backslash \{j\}$. Observe that $n \in R^g$. Then, from (2.6)

$$v^g - v^f = -\pi_n^*(g)\lambda_j(r_j(n) - \nabla y_n^f).$$

Since $\pi_n^*(g) > 0$, $\lambda_j > 0$, and $r_j(n) - \nabla y_n^f < 0$, we have $v^g - v^f > 0$ and, thus, $f \notin F^*$.

(iii) For $n = 0, 1, \ldots, N-1$, let $Z_n = \{j = 1, 2, \ldots, m : \; r_j(n) = \nabla y_n^f\}$. Consider any policy $g \in F$ such that $g_n \backslash Z_n = f_n \backslash Z_n$ for all $n = 0, 1, \ldots, N-1$. Then $H^g(n, \nabla y_n^f) - H^f(n, \nabla y_n^f) = 0$ for all $n = 0, 1, \ldots, N-1$, and (2.6) implies $v^g - v^f = 0$. Thus, if $f \in F^*$, then $g \in F^*$. Since $v^g = v^f = v^*$, from (2.5), $\nabla y_{N-1}^g = \nabla y_{N-1}^f = \frac{v^*}{k\mu}$. Since $g(n, j) = f(n, j)$ when $r_j(n) \neq \nabla y_n^f$, by solving (2.5) with g and f, we have that $\nabla y_{N-2}^g = \nabla y_{N-2}^f$. By repeating this argument $N-1$ times, we have $\nabla y_n^g = \nabla y_n^f$ for $n = 0, 1, \ldots, N-1$.

(iv) From (i)–(iii) of this lemma, $\max_{a \in A(n)}\{\sum_{j \in a} \lambda_j(r_j(n) - \nabla y_n^f)\} = \sum_{j=1}^m \lambda_j(r_j(n) - \nabla y_n^f)^+ = \sum_{j \in f_n} \lambda_j(r_j(n) - \nabla y_n^f) = H^f(n, \nabla y_n^f) \geq 0$ for $n = 0, 1, \ldots, n^f$. ■

Below are the definition of the restriction of a policy and the definition of an essential trunk reservation policy.

Definition 2.2 *We call $f^R \in F$ the restriction of policy $f \in F$ if*

$$f_n^R = \begin{cases} f_n, & \text{if } n \in R^f; \\ \emptyset, & \text{if } n \in T^f. \end{cases}$$

A policy $f \in F$ is called an essential trunk reservation policy (ETRP) if $f^R \in$

F is a TRP.

The following lemma and theorem describe the relations between f and its restriction f^R.

Lemma 2.2 *$v^f = v^{f^R}$ for any $f \in F$, and, therefore, $f \in F^*$ if and only if $f^R \in F^*$.*

Proof Since $n^{f^R} = n^f$, we have that $R^{f^R} = R^f = \{0, 1, \ldots, n^f\}$ and $T^{f^R} = T^f = \{n^f + 1, \ldots, N\}$. The policies f and f^R coincide in $\{0, 1, \ldots, n^f\}$. Thus, $v^f = v^{f^R}$. ■

Let C^* be the set of all canonical policies, E^* be the set of all optimal ETRPs, and T^* be the set of all optimal TRPs. Then the following theorem links the sets of optimal policies F^*, canonical policies C^*, optimal ETRPs E^*, and optimal TRPs T^*.

Theorem 2.2 *A policy $f \in F^*$ if and only if $f^R \in C^*$. In addition, $T^* \subseteq E^* \subseteq F^*$. In the case with zero holding costs, $T^* = C^* = E^* = F^*$.*

Proof If $f^R \in C^*$, then $f^R \in F^*$. From Lemma 2.2, $f \in F^*$. If $f \in F^*$, then, from Lemma 2.2, $f^R \in F^*$. Now let $n^f = N$. Then Lemma 2.1(iv) implies that $H^{f^R}(n, \nabla y_n^{f^R}) = \max_{a \in A(n)}\{\sum_{j \in a} \lambda_j (r_j(n) - \nabla y_n^{f^R})\}$ for $n = 0, 1, \ldots, N-1$. Proposition 2.1(a) implies $f^R \in C^*$. If $n^f \leq N - 1$, then Lemma 2.1 implies $\nabla y_{n^f}^{f^R} \geq \max_{j=1,2,\ldots,m} r_j(n^f)$. Since $f_n^R = \emptyset$ for $n \geq n^f$, formula (2.5) implies $\nabla y_{n^f-1}^{f^R} = \nabla y_{n^f}^{f^R} = \cdots = \nabla y_{N-1}^{f^R} = \frac{v^*}{k\mu} \geq \max_{j=1,2,\ldots,m} r_j(n^f)$. Since $r_j(n)$ are nonincreasing in $n = 0, 1, \ldots, N-1$, we have $\nabla y_n^{f^R} \geq \max_{j=1,2,\ldots,m} r_j(n)$ for $n \in T^f = \{n^f + 1, n^f + 2, \ldots, N\}$ and

	State 0	State 1	State 2	State 3	State 4	State 5
Type 1	A	A	R	R	R	R
Type 2	A	R	R	R	R	R
Type 3	A	A/R	A/R	A/R	A/R	R

Table 2.2: Canonical policies for Example 2.2. A=Accept; R=Reject.

$H^{f^R}(n, \nabla y_n^{f^R}) = 0 = \max_{a\in A(n)}\{\sum_{j\in a}\lambda_j(r_j(n) - \nabla y_n^{f^R})\}$ for $n \in T^f$. Lemma 2.1(iv) implies $H^{f^R}(n, \nabla y_n^{f^R}) = \max_{a\in A(n)}\{\sum_{j\in a}\lambda_j(r_j(n)-\nabla y_n^{f^R})\}$ for $n \in R^f$. From Proposition 2.1(a), $f^R \in C^*$.

Definition 2.2 implies that $T^* \subseteq E^* \subseteq F^*$. In the case without holding costs, the result of Feinberg and Reiman [9, Theorem 3.1] implies that $T^* = F^*$. Thus, in the case without holding costs, $T^* = C^* = E^* = F^*$. ■

The following example shows that there may exist a canonical policy which is not an ETRP for our problem.

Example 2.2 Consider an M/M/1/5 queue with three customer types, where $\lambda_1 = \lambda_2 = \lambda_3 = \mu = 1$, $r_1(0) = 10$, $r_1(1) = 7$, $r_1(2) = 4$, $r_1(3) = 1$, $r_1(4) = -2$, $r_2(0) = 8$, $r_2(1) = 6$, $r_2(2) = 4$, $r_2(3) = 2$, $r_2(4) = 0$, and $r_3(0) = r_3(1) = r_3(2) = r_3(3) = r_3(4) = 6.5$. Consider a TRP d with $d_0 = \{1, 2, 3\}$, $d_1 = \{1, 3\}$, and $d_2 = d_3 = d_4 = d_5 = \emptyset$. Equations (2.5) imply that d is a canonical policy with $\nabla y_0^d = 6$ and $\nabla y_1^d = \nabla y_2^d = \nabla y_3^d = \nabla y_4^d = v^d = 6.5$. Now consider another policy f with $f_0 = \{1, 2, 3\}$, $f_1 = \{1\}$, $f_2 = \{3\}$, and $f_3 = f_4 = f_5 = \emptyset$. Note that $f = f^R$ and f^R is not an ETRP. Equations (2.5) also imply that f is a canonical policy with $\nabla y_0^f = 6$ and $\nabla y_1^f = \nabla y_2^f = \nabla y_3^f = \nabla y_4^f = v^f = 6.5$.

The following formula describes an arbitrary canonical policy for Example

2.2:

$$f(n,j) = \begin{cases} \{accept\}, & \text{if } n = 0 \text{ or } (n,j) = (1,1); \\ \{reject\}, & \text{if either } (n,j) = (1,2) \text{ or } n = 2,3,4 \text{ and } j = 1,2; \\ arbitrary, & \text{if } n = 1,2,3,4 \text{ and } j = 3. \end{cases}$$

This description contrasts to the result without holding costs [17], where there exist at most two optimal control levels for each customer type. We also observe that, in Example 2.2, there is only one customer type with more than two optimal control levels. In Theorem 2.4, we will show that, if, as stated in Condition 2.1, different customer types have different net reward functions, then there exists only one customer type with more than two optimal control levels under an additional condition. This additional condition is Condition 2.2 which provides necessary and sufficient conditions when such customer type exists. In order to formulate Condition 2.2, Definition 2.4, and Theorems 2.3 and 2.4, we need the following two lemmas.

Lemma 2.3 *Let* $\tilde{R} = \sum_{j=1}^{m} \lambda_j \times r_j(0)$. *Then* $0 < v^* < \tilde{R}$.

Proof Since $R(n, f_n) \leq \tilde{R}$, $n = 0, 1, \ldots, n^f - 1$, $R(n^f, f_{n^f}) = 0$, and $\pi^*_{n^f}(f) > 0$, we have $v^f = \sum_{n=0}^{n^f-1} \pi^*_n(f) R(n, f_n) \leq \sum_{n=0}^{n^f-1} \pi^*_n(f) \tilde{R} < \tilde{R}$. Furthermore, let f accept all customer types if and only if the system is empty. Then $v^* \geq v^f = \pi^*_0(f) \times \tilde{R} > 0$. ■

Equations (2.5) and Theorem 2.1 imply $\nabla y^f_{n^f-1} = \frac{v^*}{k\mu}$ when $f \in F^*$. For $f \in F^*$, we define $N^f = \min\{n = 0, 1, \ldots, n^f - 1 : \nabla y^f_n = \frac{v^f}{k\mu}\}$. This definition yields $N^f \leq n^f - 1$.

Lemma 2.4 *If $f, g \in F^*$, then $N^f = N^g$ and $\nabla y_n^f = \nabla y_n^g$ for $n = 0, 1, \ldots, \min\{n^f, n^g\} - 1$.*

Proof We prove this lemma by contradiction. Let $n^f = \min\{n^f, n^g\}$. Then state n^f is recurrent under both f and g. Assume that $\nabla y_n^g \neq \nabla y_n^f$ for some $n = 0, 1 \ldots, n^f - 1$. Since $g_n \neq \emptyset$ and $f_n \neq \emptyset$, Lemma 2.1 implies $H^g(n, \nabla y_n^g) \neq H^f(n, \nabla y_n^f)$. From (2.5), $H^g(n, \nabla y_n^g) + \mu_n \nabla y_{n-1}^g = v^* = H^f(n, \nabla y_n^f) + \mu_n \nabla y_{n-1}^f$. Thus, $\nabla y_{n-1}^g \neq \nabla y_{n-1}^f$. By repeating this argument n times, we have $\nabla y_0^g \neq \nabla y_0^f$. Since $f, g \in F^*$, Lemma 2.3 implies $v^f > 0$ and $v^g > 0$. However, since $\nabla y_0^f \neq \nabla y_0^g$, Lemma 2.1 and (2.5) yield $v^g = H^g(0, \nabla y_0^g) \neq H^f(0, \nabla y_0^f) = v^f$. Thus, either g or f is not optimal. This contradicts our assumption that $f, g \in F^*$. Thus, if $f, g \in F^*$, then $\nabla y_n^g = \nabla y_n^f$ for $n = 0, 1, \ldots, n^f - 1$. Since $N^f \leq n^f - 1$ and $n^f = \min\{n^f, n^g\}$, we have $N^g = N^f$. ■

From Lemma 2.4, we know that N^f does not depend on $f \in F^*$. We let $N^* = N^f$ for any optimal policy f. Lemma 2.4 implies

$$N^* \leq \min\{n^f : f \in F^*\} - 1. \tag{2.7}$$

Example 2.2 satisfies the following condition, which indicates the existence of a special customer type i with more than two optimal control levels. As shown later, this is a necessary and sufficient condition for the existence of a canonical policy that is not a TRP.

Condition 2.2 *$N^* < N - 1$ and there exists a customer type i such that $r_i(n) = \frac{v^*}{k\mu}$ for all $n = 0, 1, \ldots, N - 1$.*

We need the following definitions before we describe the structures of canonical policies and optimal TRPs for our problem.

Definition 2.3 *An integer M_j is called an optimal control level for type j customers if there exists a policy $f \in T^*$ such that $M_j^f = M_j$.*

Definition 2.4 *Let $n_i^f = \min\{n = N^*, N^*+1, \ldots, N : f(n,i) = \{reject\}\}$. Then a policy $f^{R,i}$ is called the restriction of f for customer type i if*

$$f^{R,i}(n,j) = \begin{cases} f(n,j), & \text{for } j \neq i; \\ f^{R,i}(n,i) = \{accept\}, & \text{for } n < n_i^f; \\ f^{R,i}(n,i) = \{reject\}, & \text{for } n \geq n_i^f. \end{cases}$$

Theorems 2.3 and 2.4 and Corollary 2.1 below are the main results for our problem under average reward criterion.

Theorem 2.3 *If Condition 2.2 does not hold, then: (a) every stationary optimal policy is an ETRP; (b) $T^* = C^* \subseteq E^* = F^*$; (c) for each customer type $j = 1, 2, \ldots, m$, there exists two optimal control levels $M_j, M_j' = 0, 1, \ldots, N$ such that*

(i) $M_l \geq k$ *for customer type* l *with* $r_l(0) = \max_{j=1,2,\ldots,m} r_j(0)$;

(ii) $\max_{j=1,2,\ldots,m} M_j > N^* \geq k-1$;

(iii) *either* $M_j' = M_j$ *or* $M_j' = M_j + 1$;

(iv) *a policy f belongs to T^* if and only if it is a TRP with $M_j^f \in \{M_j, M_j'\}$ for all $j = 1, 2, \ldots, m$.*

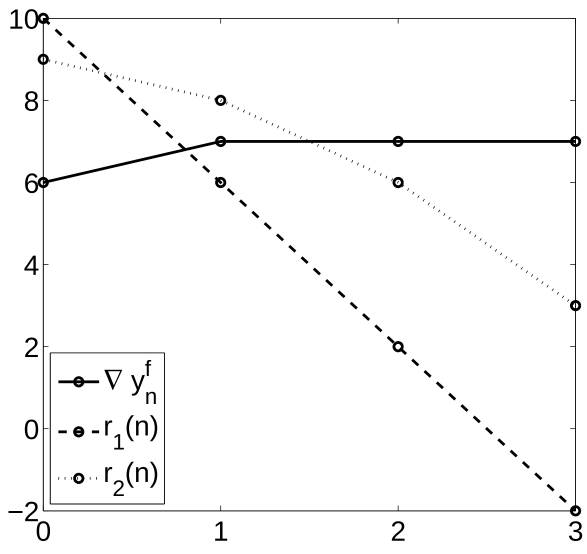

Figure 2.1: Condition 2.2 does not hold in Example 2.1

In particular, statement (c) in Theorem 2.3 means that, for each customer type j, there exist at most two optimal control levels and, if there are two different optimal control levels M_j and M_j', where $M_j' > M_j$, then $M_j' = M_j + 1$. A policy is canonical if and only if it is a TRP that follows one of these optimal control levels for each customer type. Theorem 2.3 implies the following corollary.

Corollary 2.1 *If either of the following conditions holds:*

(i) $r_j(n)$ *are decreasing in* $n = k-1, k, \ldots, N-1$*;*

(ii) $r_j(n) > r_{j+1}(n)$ *for* $j = 1, 2, \ldots, m-1$ *and* $n = 0, 1, \ldots, N-1$*;*

then statements (a)–(c) of Theorem 2.3 hold.

Theorem 2.4 *If Condition 2.2 holds, then, for each customer type* $j \in J$*, where* $J = \{1, 2, \ldots, m\} \backslash \{i\}$ *, there exist two optimal control levels* $M_j, M'_j = 0, 1, \ldots, N$ *such that*

(i) $M_l \geq k$ *for customer type* l *with* $r_l(0) = \max_{j=1,2,\ldots,m} r_j(0) > r_i(0)$*;*

(ii) $\max_{j \in J} M_j > N^* \geq k - 1$*;*

(iii) either $M'_j = M_j$ *or* $M'_j = M_j + 1$*;*

(iv) a policy f *belongs to* T^* *if and only if it is a TRP with* $M_j^f \in \{M_j, M'_j\}$*,* $j \in J$*, and* $M_i^f \in \{N^*, N^* + 1, \ldots, N\}$*.*

In addition, a policy f *belongs to* C^* *if and only if* $f^{R,i}$ *belongs to* T^**.*

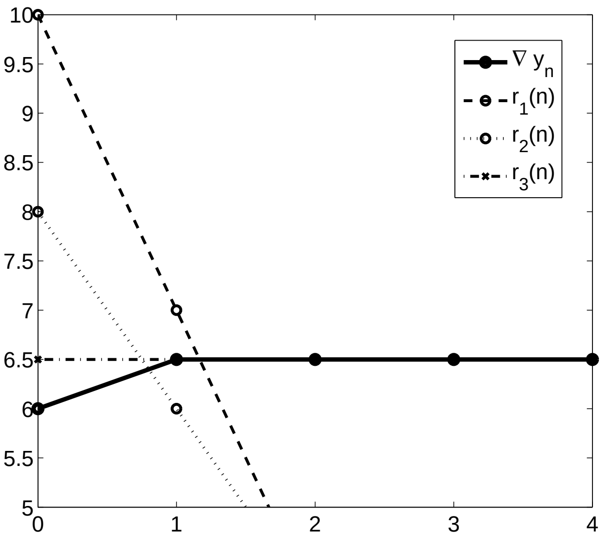

Figure 2.2: Condition 2.2 holds in Example 2.2, $i = 3$, and $N^* = 1$

Statement (iii) in Theorem 2.4 means that, for each customer type j, $j \neq i$, there exist at most two optimal control levels. If there are two different optimal control levels M_j' and M_j, where $M_j' > M_j$, then $M_j' = M_j + 1$. Since $N^* < N-1$, for customer type i, there exist more than two optimal control levels, which are $N^*, N^*+1, \ldots, N$. A TRP is optimal if and only if it follows these optimal control levels. To prove Theorem 2.3, Theorem 2.4, and Corollary 2.1, we need the following lemmas.

Lemma 2.5 *If $f \in C^*$, then the following statements hold:*

(i) $H^f(n, \nabla y_n^f) \geq 0$ for $n = 0, 1, \ldots, n^f - 1$ and $H^f(n, \nabla y_n^f) = 0$ for $n = n^f, n^f+1, \ldots, N$;

(ii) if $n^f < N$, then $\nabla y_n^f = \frac{v^*}{k\mu} \geq \max_{j=1,2,\ldots,m} r_j(n)$ for $n = n^f, n^f + 1, \ldots, N-1$.

Proof Since $f \in C^*$, Proposition 2.1(b) implies $H^f(n, \nabla y_n^f) \geq 0$ for all $n = 0, 1, \ldots, N-1$. If $n^f = N$, then $R^f = \{0, 1, \ldots, N\}$. By definition, $H^f(N, \nabla y_N^f) = 0$. Thus, statement (i) holds when $n^f = N$. Now let $n^f \leq N-1$. Since $f \in C^*$ and $f_{n^f} = \emptyset$, Proposition 2.1(c) implies $\nabla y_{n^f}^f \geq \max_{j=1,2,\ldots,m} r_j(n^f)$. Theorem 2.1 yields $n^f \geq k$ and (2.5) implies $\nabla y_{N-1}^f = \nabla y_{n^f-1}^f = \frac{v^*}{k\mu} \geq \max_{j=1,2,\ldots,m} r_j(n^f) \geq \max_{j=1,2,\ldots,m} r_j(N-1)$. Since $f \in C^*$, Proposition 2.1(b) implies $H^f(N-1, \nabla y_{N-1}^f) = \sum_{j=1}^m \lambda_j (r_j(N-1) - \nabla y_{N-1}^f)^+ = 0$. From (2.5), $k\mu \nabla y_{N-2}^f = v^*$ and, thus, $\nabla y_{N-2}^f = \frac{v^*}{k\mu}$. By repeating this argument $N-1-n^f$ times, we have, for $n^f \leq n \leq N-1$,

$$\nabla y_{n^f-1}^f = \nabla y_n^f = \frac{v^*}{k\mu} \geq \max_{j=1,2,\ldots,m} r_j(n^f) \geq \max_{j=1,2,\ldots,m} r_j(n). \tag{2.8}$$

Thus, $\nabla y_n^f = \frac{v^*}{k\mu} \geq \max_{j=1,2,\ldots,m} r_j(n)$ and $H^f(n, \nabla y_n^f) = \sum_{j=1}^m \lambda_j (r_j(n) - \nabla y_n^f)^+ = 0$ for $n = n^f, n^f+1, \ldots, N-1$. ■

Lemma 2.6 *If $f \in C^*$, then* $\max_{j=1,2,\ldots,m} r_j(0) > \nabla y_n^f$, $n = 0, 1, \ldots, N-1$.

Proof We prove this lemma by contradiction. Assume that $r_l(0) = \max_{j=1,2,\ldots,m} r_j(0) \leq \nabla y_n^f$ for some $n = 0, 1, \ldots, N-1$. Since $f \in C^*$, from Proposition 2.1(c), $H^f(n, \nabla y_n^f) = 0$. From Lemma 2.5(i), $H^f(n+1, \nabla y_{n+1}^f) \geq 0$. From (2.5), $H^f(n+1, \nabla y_{n+1}^f) + \mu_{n+1}\nabla y_n^f = v^f = v^* \geq H^f(n+1, \nabla y_{n+1}^f) + \mu_{n+1} r_l(0) \geq \mu_{n+1} r_l(0)$ and $\mu_n \nabla y_{n-1}^f = v^*$ because of $H^f(n, \nabla y_n^f) = 0$. Since $H^f(n+1, \nabla y_{n+1}^f) + \mu_{n+1}\nabla y_n^f = v^* = \mu_n \nabla y_{n-1}^f$ and $H^f(n+1, \nabla y_{n+1}^f) \geq 0$, we have $\mu_n \nabla y_{n-1}^f \geq \mu_{n+1} r_l(0)$. Thus, $\nabla y_{n-1}^f \geq \frac{\mu_{n+1}}{\mu_n} r_l(0) \geq r_l(0)$. By repeating this argument n times, we obtain $\nabla y_0^f \geq r_l(0)$. Since $\nabla y_0^f \geq r_l(0)$, from Proposition 2.1(b) and (2.5), $H^f(0, \nabla y_0^f) = \sum_{j=1}^m \lambda_j (r_j(0) - \nabla y_0^f)^+ = 0 = v^*$. This contradicts Lemma 2.3. Thus, $r_l(0) > \nabla y_n^f$ for $n = 0, 1, \ldots, N-1$. ■

Lemma 2.7 *The following statements hold:*

(i) if $f \in C^*$, then $k - 1 \leq N^*$ and $0 < \nabla y_0^f < \nabla y_1^f < \cdots < \nabla y_{N^*}^f = \nabla y_{N^*+1}^f = \cdots = \nabla y_{N-1}^f = \frac{v^*}{k\mu}$;

(ii) if $f, g \in C^*$, then $\nabla y_n^f = \nabla y_n^g$ for $n = 0, 1, \ldots, N-1$.

Proof (i) If $\nabla y_0^f \leq 0$, then $v^* = H^f(0, \nabla y_0^f) = \sum_{j=1}^m \lambda_j (r_j(n) - \nabla y_0^f) \geq \sum_{j=1}^m \lambda_j r_j(0)$, where the first equality follows from (2.5) and the second equality follows from Proposition 2.1(b). Therefore, $v^* \geq \sum_{j=1}^m \lambda_j r_j(0)$ and this contradicts Lemma 2.3. Thus, $\nabla y_0^f > 0$. Next we show that $\nabla y_n^f < \nabla y_{n+1}^f$ for $n = 0, 1, \ldots, k-2$. From Theorem 2.1, $f_n \neq \emptyset$ for $n = 0, 1, \ldots, k-1$. Assume $\nabla y_n^f \geq \nabla y_{n+1}^f$ for some n, $0 \leq n \leq k-2$. Since $f_n \neq \emptyset$, $f_{n+1} \neq \emptyset$, $\nabla y_n^f \geq \nabla y_{n+1}^f$, and $r_j(n) = r_j(n+1) = r_j(0)$ for $j = 1, 2, \ldots, m$, from Proposition 2.1(b), we have $H^f(n, \nabla y_n^f) = \sum_{j=1}^m \lambda_j(r_j(0) - \nabla y_n^f)^+ \leq \sum_{j=1}^m \lambda_j(r_j(0) - \nabla y_{n+1}^f)^+ = H^f(n+1, \nabla y_{n+1}^f)$. From (2.5), $H^f(n+1, \nabla y_{n+1}^f) + (n+1)\mu \nabla y_n^f = v^* = H^f(n, \nabla y_n^f) + n\mu \nabla y_{n-1}^f$. Since $H^f(n, \nabla y_n^f) \leq H^f(n+1, \nabla y_{n+1}^f)$, we have $\nabla y_{n-1}^f \geq \frac{n}{n-1} \nabla y_n^f > \nabla y_n^f$. By repeating this argument n times, we have

a. $\nabla y_{n+1}^f \leq \nabla y_n^f < \cdots < \nabla y_1^f < \nabla y_0^f$;

b. $H^f(0, \nabla y_0^f) < H^f(1, \nabla y_1^f) < \cdots < H^f(n, \nabla y_n^f) \leq H^f(n+1, \nabla y_{n+1}^f)$.

From (2.5), $H^f(0, \nabla y_0^f) = v^* = H^f(1, \nabla y_1^f) + \mu_1 \nabla y_0^f$. Since $H^f(0, \nabla y_0^f) \leq H^f(1, \nabla y_1^f)$, we have $\nabla y_0^f \leq 0$. This contradicts the result that $\nabla y_0^f > 0$. Therefore, if $f \in C^*$, then

$$0 < \nabla y_0^f < \nabla y_1^f < \cdots < \nabla y_{k-1}^f. \tag{2.9}$$

From Theorem 2.1, $n^f \geq k$ for any $f \in C^*$. If $n^f = k$, then (2.8) implies that $\nabla y_{k-1}^f = \nabla y_k^f = \cdots = \nabla y_{N-1}^f = \frac{v^*}{k\mu}$, $N^* = k-1$, and (i) is proven. Thus, in the rest of the proof of (i), we consider $n^f > k$.

Let $n = k-1, k, \ldots, n^f - 2$. First we shall prove that $\nabla y_n^f \leq \nabla y_{n+1}^f$. Assume $\nabla y_n^f > \nabla y_{n+1}^f$ for some $n = k-1, k, \ldots, n^f - 2$. From (2.5), $H^f(n+1, \nabla y_{n+1}^f) + k\mu \nabla y_n^f = v^* = H^f(n+2, \nabla y_{n+2}^f) + k\mu \nabla y_{n+1}^f$. Thus, $H^f(n+1, \nabla y_{n+1}^f) < H^f(n+2, \nabla y_{n+2}^f)$. Since $H^f(n+1, \nabla y_{n+1}^f) < H^f(n+2, \nabla y_{n+2}^f)$, Proposition 2.1(b) implies $H^f(n+1, \nabla y_{n+1}^f) = \sum_{j=1}^m \lambda_j(r_j(n+1) - \nabla y_{n+1}^f)^+ < \sum_{j=1}^m \lambda_j(r_j(n+2) - \nabla y_{n+2}^f)^+ = H^f(n+2, \nabla y_{n+2}^f)$. Since $r_j(n+1) \geq r_j(n+2)$ for $j = 1, 2, \ldots, m$, we have $\nabla y_{n+1}^f > \nabla y_{n+2}^f$. By repeating this argument $n^f - 1 - n$ times, we have

a. $\nabla y_{n^f}^f < \nabla y_{n^f-1}^f < \cdots < \nabla y_{n+1}^f < \nabla y_n^f$;

b. $H^f(n^f, \nabla y^f_{n^f}) > H^f(n^f - 1, \nabla y^f_{n^f-1}) > \cdots > H^f(n+2, \nabla y^f_{n+2}) > H^f(n+1, \nabla y^f_{n+1})$.

Since $f_{n^f} = \emptyset$, we have $H^f(n^f, \nabla y^f_{n^f}) = 0$. Thus, $0 > H^f(n^f - 1, \nabla y^f_{n^f-1})$. Lemma 2.5(i) implies $f \notin C^*$ and this contradicts $f \in C^*$. Therefore, if $f \in C^*$, then

$$\nabla y^f_{k-1} \leq \nabla y^f_k \leq \cdots \leq \nabla y^f_{n^f-1}. \tag{2.10}$$

From (2.8), (2.9), (2.10), $\nabla y^f_{k-2} < \nabla y_{k-1} \leq \cdots \leq \nabla y^f_{n^f-1} = \nabla y^f_{n^f} = \cdots = \nabla y^f_{N-1} = \frac{v^*}{k\mu}$. By definition, we obtain $k - 1 \leq N^* \leq n^f - 1$ and

$$\nabla y^f_{k-1} \leq \cdots \leq \nabla y^f_{N^*-1} < \nabla y^f_{N^*} = \nabla y^f_{N^*+1} = \cdots = \nabla y^f_{N-1} = \frac{v^*}{k\mu}. \tag{2.11}$$

Next we prove that $\nabla y_n < \nabla y_{n+1}$ for $n = k-1, k, \ldots, N^* - 2$. Assume that$\nabla y^f_n = \nabla y^f_{n+1}$ for some $k - 1 \leq n \leq N^* - 2$. Then (2.5) yields $H^f(n+1, \nabla y^f_{n+1}) = H^f(n+2, \nabla y^f_{n+2})$. If $\nabla y^f_{n+1} < \nabla y^f_{n+2}$, then Proposition 2.1(b) implies that the only possibility is that $H^f(n+1, \nabla y^f_{n+1}) = H^f(n+2, \nabla y^f_{n+2}) = 0$. From (2.5), $H^f(n+2, \nabla y^f_{n+2}) + k\mu\nabla y^f_{n+1} = H^f(n+3, \nabla y^f_{n+3}) + k\mu\nabla y^f_{n+2}$. Since $\nabla y^f_{n+1} < \nabla y^f_{n+2}$ and $H^f(n+2, \nabla y^f_{n+2}) = 0$, we have $H^f(n+3, \nabla y^f_{n+3}) < 0$ and this contradicts Proposition 2.1(b). Thus, $\nabla y^f_{n+1} = \nabla y^f_{n+2}$. By repeating this argument $N^* - n$ times, we have $\nabla y^f_n = \nabla y^f_{n+1} = \cdots = \nabla y^f_{N^*-1} = \nabla y^f_{N^*} = \frac{v^*}{k\mu}$ and this contradicts (2.11). Therefore, $\nabla y^f_n < \nabla y^f_{n+1}$ for $n = k-1, k, \ldots, N^* - 2$ and, from (2.9) and (2.11), (i) is proven.

(ii) Lemma 2.4 implies that $\nabla y^g_n = \nabla y^f_n$ for all $n = 0, 1, \ldots, N^*$. From Lemma 2.7(i), $\nabla y^f_{N^*} = \nabla y^f_{N^*+1} = \cdots = \nabla y^f_{N-1} = \frac{v^*}{k\mu}$ for any $f \in C^*$. Thus, if $f, g \in C^*$, then $\nabla y^f_n = \nabla y^g_n$ for all $n = 0, 1, \ldots, N-1$. ■

In view of Lemma 2.7(ii), we set $u_n = \nabla y_n^f$ for any $f \in C^*$. Lemma 2.7(i) implies that

$$0 < u_0 < u_1 < \cdots < u_{N^*} = u_{N^*+1} = \cdots = u_{N-1} = \frac{v^*}{k\mu}, \tag{2.12}$$

and this lemma and (2.7) imply

$$k - 1 \le N^* \le \min\{n^f : f \in F^*\} - 1. \tag{2.13}$$

For $j = 1, 2, \ldots, m$, we define M_j and M'_j as follows:

$$M_j = \begin{cases} \min\{n = 0, 1, \ldots, N-1 : r_j(n) \le u_n\}, & \text{if } r_j(N-1) \le u_{N-1}; \\ N, & \text{otherwise.} \end{cases} \tag{2.14}$$

$$M'_j = \begin{cases} \min\{n = 0, 1, \ldots, N-1 : r_j(n) < u_n\}, & \text{if } r_j(N-1) < u_{N-1}; \\ N, & \text{otherwise.} \end{cases} \tag{2.15}$$

The lemma below shows a very important property of any policy $f \in C^*$.

Lemma 2.8 *A policy $f \in C^*$ if and only if, for all $j = 1, 2, \ldots, m$,*

$$f(n, j) = \begin{cases} \{accept\}, & \textit{if } n < M_j; \\ \{reject\}, & \textit{if } n \ge M'_j. \end{cases} \tag{2.16}$$

Proof Condition 2.1 and (2.12) imply respectively that the functions $r_j(n)$, $j = 1, 2, \ldots, m$, are nonincreasing in n and the numbers u_n are nondecreasing in n. This implies that

$$\{n = 0, 1, \ldots, N-1 : n < M_j\} = \{n = 0, 1, \ldots, N-1 : r_j(n) > u_n\},$$

$$\{n = 0, 1, \ldots, N-1 : M_j \le n < M'_j\} = \{n = 0, 1, \ldots, N-1 : r_j(n) = u_n\},$$

$$\{n = 0, 1, \ldots, N-1 : M'_j \le n \le N-1\} = \{n = 0, 1, \ldots, N-1 : r_j(n) < u_n\}.$$

Let $f \in C^*$. Then $u_n = \nabla y_n^f$, $n = 0, 1, \ldots, N-1$, and Proposition 2.1(c) implies (2.16). Now let f satisfy (2.16). Consider $g \in C^*$. Then, according to the proven necessary part, g satisfies (2.16) too with f substituted with g in (2.16). Observe that $H^f(n, u_n) = H^g(n, u_n)$, $n = 0, 1, \ldots, N-1$. From (2.12), $\nabla y_n^g = u_n$ for all n. Equations (2.5) imply $\nabla y_n^f = \nabla y_n^g = u_n$ for all n. Proposition 2.1(c) yields $f \in C^*$. ■

Lemma 2.9 *Consider M_j and M'_j, $j = 1, 2, \ldots, m$, defined in (2.14) and (2.15). If Condition 2.2 does not hold, then statements (i)–(iv) of Theorem 2.3 hold. If Condition 2.2 holds, then statements (i)–(iv) of Theorem 2.4 hold and, in addition, $M_i = N^*$ and $M'_i = N$.*

Proof Statement (i) of Theorem 2.3 and the same inequality in statement (i) of Theorem 2.4 follow from Theorem 2.1. If Condition 2.2 holds, then Lemma 2.6 and Lemma 2.7(i) imply that $r_l(0) = \max_{j=1,2,\ldots,m} r_j(0) > u_{N-1} = \frac{v^*}{k\mu} = r_i(0)$. Statement (i) of Theorem 2.4 is proven.

For (ii), (2.5) implies that $H^f(N^*+1, u_{N^*+1}) + k\mu u_{N^*} = v^* = H^f(N^*, u_{N^*}) + k\mu u_{N^*-1}$ for any $f \in C^*$. From (2.12), $u_{N^*} = \frac{v^*}{k\mu} > u_{N^*-1}$. Therefore, $H^f(N^*+1, u_{N^*+1}) > H^f(N^*, u_{N^*}) \ge 0$, where the second inequality follows from Proposition 2.1(b). Since $H^f(N^*, u_{N^*}) > 0$, Proposition 2.1(b)(c) imply there exists some customer type l, $l \neq i$, such that $r_l(N^*) - u_{N^*} > 0$ and $f(N^*, l) = \{\mathit{accept}\}$ for any $f \in C^*$. Lemma 2.8 implies $M_l > N^*$. For statement (ii) of Theorem 2.3, $\max_{j=1,2,\ldots,m} M_j \geq M_l > N^*$. For statement (ii) of Theorem 2.4, since $r_l(N^*) > u_{N^*} = \frac{v^*}{k\mu} = r_i(N^*)$, we have $l \neq i$. Thus, we obtain $\max_{j \in J} M_j \geq M_l > N^*$. The first inequalities in statements (ii) of both theorems are proven. The second inequalities in statements (ii) of both theorems follow from (2.13).

For (iii), let either Condition 2.2 does not hold or $j \neq i$. If $M_j = N$ or $M_j = N - 1$, then the statement is obvious. Thus, let $M_j < N - 1$. From (2.14), either $r_j(M_j) = u_{M_j}$ or $r_j(M_j) < u_{M_j}$. If $r_j(M_j) < u_{M_j}$, then, from (2.14) and (2.15), we have $M'_j = M_j$. If $r_j(M_j) = u_{M_j}$, then we shall prove that $M'_j = M_j + 1$. Assume that $M'_j > M_j + 1$. Since $M'_j \leq N$, we have $M_j \leq N - 2$. Since $r_j(M_j) = u_{M_j}$ and $M'_j > M_j + 1$, (2.14) and (2.15) imply that $r_j(M_j + 1) = u_{M_j+1}$. Since $r_j(M_j + 1) \leq r_j(M_j)$ and $u_{M_j} \leq u_{M_j+1}$, we have $r_j(M_j+1) = r_j(M_j)$ and $u_{M_j} = u_{M_j+1}$. Formula (2.12) yields $u_{M_j} = \frac{v^*}{k\mu} = r_j(M_j) = r_j(M_j + 1)$ and $N - 2 \geq M_j \geq N^*$. From (2.13), $M_j \geq N^* \geq k - 1$. Since $r_j(M_j) = r_j(M_j + 1) = \frac{v^*}{k\mu}$ and $M_j \geq k - 1$, Condition 2.1 implies that $r_j(n) = \frac{v^*}{k\mu}$ for all $n = 0, 1, \ldots, N-1$ and, thus, $j = i$. From (2.12) and (2.14), we have $N^* = M_i \leq N - 2$. Since $N^* < N - 1$ and there exists a customer type i, Condition 2.2 holds. This contradicts our assumption that Condition 2.2 does not hold. Therefore, if either Condition 2.2 does not hold or $j \neq i$, then:

$$M'_j = \begin{cases} M_j + 1, & \text{if and only if } r_j(M_j) = u_{M_j}; \\ M_j, & \text{otherwise.} \end{cases}$$

Statements (iii) of both theorems are proven.

For (iv), we prove statement (iv) of Theorem 2.3 first. Consider $f \in T^*$. Then, from Theorem 2.2, $f = f^R \in C^*$. Since $f \in C^*$, statement (iii) of Theorem 2.3 and Lemma 2.8 imply that $M_j^f \in \{M_j, M'_j\}$ for all $j = 1, 2, \ldots, m$. Thus, f is a TRP with $M_j^f \in \{M_j, M'_j\}$ for all $j = 1, 2, \ldots, m$. Now let f be a TRP with $M_j^f \in \{M_j, M'_j\}$ for $j = 1, 2, \ldots, m$. Lemma 2.8 implies $f \in C^*$. Since $C^* \subseteq F^*$, we have $f \in T^*$. Next we prove statement (iv) of Theorem 2.4. Consider $f \in T^*$. Then, from Theorem 2.2, $f = f^R \in C^*$. Statement (iii) of Theorem 2.4 and Lemma 2.8 imply that $M_j^f \in \{M_j, M'_j\}$, $j \neq i$. In addition, (2.12) and Condition 2.2 imply that

$$r_i(n) = u_n = \frac{v^*}{k\mu}, \text{ for } n = N^*, N^* + 1, \ldots, N - 1, \tag{2.17}$$

and (2.14) and (2.15) imply that $M_i = N^*$ and $M'_i = N$. Lemma 2.8 implies that $M_i^f \in \{N^*, N^* + 1, \ldots, N\}$. Now let f be a TRP with $M_j^f \in \{M_j, M'_j\}$, $j \neq i$, and $M_i^f \in \{N^*, N^*+1, \ldots, N\}$. Then Lemma 2.8 implies $f \in C^*$. Since $f \in C^* \subseteq F^*$, $f \in T^*$. ■

Proof of Theorem 2.3: (c) Statments (i)–(iv) follow from Lemma 2.9.

In the following, we prove statement (b) first and statement (a) follows from (b).

(b) If $f \in C^*$, then statement (iii) of this theorem and Lemma 2.8 imply that $M_j^f \in \{M_j, M_j'\}$ for all $j = 1, 2, \ldots, m$ and, thus, $f \in T^*$. Now let $f \in T^*$. Then from Theorem 2.2, $f = f^R \in C^*$. Thus, $T^* = C^*$. Now consider any $f \in F^*$. Then, from Theorem 2.2, $f^R \in C^*$. Since $C^* = T^*$, $f^R \in T^*$ is a TRP. From Definition 2.2, f is an ETRP and, thus, $f \in E^*$. Since any $f \in F^*$ is an ETRP, we obtain $F^* = E^*$. Since $C^* \subseteq F^*$, we obtain $T^* = C^* \subseteq E^* = F^*$.

(a) From (b), $E^* = F^*$. Thus, every stationary optimal policy is an ETRP.

■

Proof of Corollary 2.1: We show that Condition 2.2 does not hold in each case.

(i) Since $r_j(n)$ are decreasing in $n = k-1, k, \ldots, N-1$ for all $j = 1, 2, \ldots, m$, there does not exist a customer type i with $r_i(n) = \frac{v^*}{k\mu}$ for $n = 0, 1, \ldots, N-1$. Thus, Condition 2.2 does not hold in this case and statements (a)–(c) of Theorem 2.3 hold.

(ii) If $N = k$, then Lemma 2.7(i) implies $N^* = k - 1 = N - 1$. Thus, Condition 2.2 does not hold. Now let $N > k$. If there does not exist a customer type i with $r_i(n) = \frac{v^*}{k\mu}$ for $n = 0, 1, \ldots, N-1$, then Condition 2.2 does not hold. Thus, the remaining case is when $N > k$ and there exists a customer type i with $r_i(n) = \frac{v^*}{k\mu}$ for $n = 0, 1, \ldots, N-1$. In this case, we shall prove that $N^* = N - 1$. Lemma 2.6 states that there exists a customer type l with $r_l(0) = \max_{j=1,2,\ldots,m} r_j(0) > \frac{v^*}{k\mu} = r_i(0)$. Since $r_j(n) > r_{j+1}(n)$ for all $n = 0, 1, \ldots, N-1$ and $j = 1, 2, \ldots, m-1$ and $r_l(0) > r_i(0) = \frac{v^*}{k\mu}$, we have that $r_l(n) > r_i(n) = \frac{v^*}{k\mu}$ for all $n = 0, 1, \ldots, N-1$. Equations (2.5) imply $u_{N-1} = \frac{v^*}{k\mu} < r_l(N-1)$. Proposition 2.1 implies $f(N-1, l) = \{accept\}$ and $H^f(N-1, u_{N-1}) > 0$. Again, from (2.5), $H^f(N, u_N) + k\mu u_{N-1} = v^* =$

$H^f(N-1, u_{N-1}) + k\mu u_{N-2}$. Since $H^f(N-1, u_{N-1}) > H^f(N, u_N) = 0$, we have $u_{N-1} = \frac{v^*}{k\mu} > u_{N-2}$ and, from Lemma 2.7(i), $N^* = N-1$. Since $N^* = N-1$, Condition 2.2 does not hold. Thus, statements (a)–(c) of Theorem 2.3 hold.■

Proof of Theorem 2.4: Statements (i)–(iv) follow from Lemma 2.9. If $f \in C^*$, then statement (iii) of this theorem and Lemma 2.8 imply that $M_j^f \in \{M_j, M_j'\}$, $j \in J$, and (2.17), (2.12), and Lemma 2.8 imply that $M_i = N^*$, $M_i' = N$, and, thus, $n_i^f \geq N^*$. Definition 2.4 implies $M_j^{f^{R,i}} = M_j^f \in \{M_j, M_j'\}$, $j \in J$, and $M_i^{f^{R,i}} = n_i^f \in \{N^*, N^*+1, \ldots, N\}$. Lemma 2.8 implies $f^{R,i} \in C^*$. Since $f^{R,i} \in C^*$ is a TRP, we have $f^{R,i} \in T^*$. Now let $f^{R,i} \in T^*$. Then $f^{R,i} = f^R \in C^*$. Statements (iii)–(iv) of this theorem and Lemma 2.8 imply that $M_j^{f^{R,i}} \in \{M_j, M_j'\}$, $j \in J$, and $M_i^{f^{R,i}} \in \{N^*, N^*+1, \ldots, N\}$. Since $N^* = M_i \leq M_i^{f^{R,i}}$, from Definition 2.4, $M_j^f = M_j^{f^{R,i}} \in \{M_j, M_j'\}$, $j \in J$, and $f(n,i) = \{accept\}$, $n = 0, 1, \ldots, M_i - 1$. Again Lemma 2.8 implies $f \in C^*$.■

Theorem 2.5 $\emptyset \neq T^* \subseteq C^* \subseteq F^*$.

Proof Since S and A are finite, according to [29], $C^* \neq \emptyset$. By definition, $C^* \subseteq F^*$. If Condition 2.2 does not hold, then Theorem 2.3(b) implies $T^* = C^* \subseteq F^*$ and $T^* \neq \emptyset$. If Condition 2.2 holds, then Theorem 2.4 implies $\emptyset \neq T^* \subseteq C^*$. Thus, if Condition 2.2 holds, then $T^* \subseteq C^* \subseteq F^*$ and $T^* \neq \emptyset$.
■

Note that Theorem 2.4(ii) states that $\max_{j=1,2,\ldots,m} M_j > N^*$. Consider a policy f such that $M_j^f \in \{M_j, M_j'\}$ for $j \neq i$, $f(n,i) = \{accept\}$ for $n = 0, 1, \ldots, N^*-1$, $f(N^*, i) = \{reject\}$, and $f(n,i) = \{accept\}$ for $n = N^* +$

$1, N^* + 2, \ldots, N - 1$. Then f is a canonical policy, but it is not an ETRP. Thus, if Condition 2.2 holds, then C^* is not a subset of E^*.

Remark 2.1 If $\lim_{n\to\infty} r_j(n) < 0$ for each $j = 1, 2, \ldots, m$, then the control admission problem for an infinite capacity queue can be reduced to the problem for a finite capacity queue. Under this natural assumption, for each $j = 1, 2, \ldots, m$, there exists a finite number N_j such that $r_j(N_j) < 0$ and it is not optimal to admit customer type j at state n, where $n \geq N_j$. Let $U = \min\{n = k, k+1, \ldots \ : \ r_j(n) \leq 0 \ for \ all \ j = 1, 2, \ldots, m\}$. Then the set C^* for the M/M/k/∞ queue is the same as the set C^* for the M/M/k/U queue. In other words, if $h_j(t) > 0$ for all $j = 1, 2, \ldots, m$ and $t > 0$, then the control admission problem for M/M/k/∞ queues is the same as the problem for M/M/k/U queues.

2.4 Bias Optimal Policy

If there exist two or more stationary optimal policies, bias optimality can be applied to find the policy that maximizes the bias vector among all stationary optimal policies. Haviv and Puterman [10] considered an M/M/k/N queue with one customer type and convex increasing holding costs. They proved that there exist at most two optimal control levels for the customers and, if there are two different optimal control levels, the difference between them is one. They also showed that the bias optimal policy is unique, is the policy that selects the larger optimal control level for the customers, and is also Blackwell optimal. Lewis et al. [17] extended these results to M/M/k/N queues with multiple customer types and without holding costs. For the problem with several customer types and holding costs, we also show that the bias optimal policy is unique, selects the largest optimal control level for each customer type, and is also Blackwell optimal. Unlike the cases considered in [10] and [17], Example 2.2 demonstrates the possibility that there are more than two optimal control levels for one customer type. For this customer type, the optimal control level under the bias optimal policy is N. Condition 2.2 provides the necessary and sufficient condition when there are more than two optimal control levels for some customer type. This customer type exists if and only if Condition 2.2 holds, this type is the type i described in Condition 2.2, and the number of optimal control levels for any other customer type j is either one or two. Below are the definitions of bias optimality and Blackwell optimality.

Definition 2.5 *A policy g is bias optimal if g is optimal and $y_n^g \geq y_n^f$ for $n = 0, 1, \ldots, N$, and for every policy $f \in F^*$.*

Definition 2.6 *A policy ϕ is Blackwell optimal if there exists a number $\alpha^* > 0$ such that $v_\alpha^\phi \geq v_\alpha^f$ for all $\alpha \in (0, \alpha^*]$ and all for $f \in F$, where v_α^f is the total discounted reward under policy f with discount rate α.*

The following theorem describes the bias optimal policy g for our problem.

Theorem 2.6 *There exists a unique bias optimal policy g and it is also Blackwell optimal. This policy is a TRP that assigns the largest optimal control level for each customer type. In particular, if Condition 2.2 holds, then g is a TRP with $M_j^g = M_j'$, $j \neq i$, and $M_i^g = N$; if Condition 2.2 does not hold, then g is a TRP with $M_j^g = M_j'$ for $j = 1, 2, \ldots, m$.*

Since $y_n^f = y_0^f - \sum_{z=0}^{n-1} \nabla y_z^f$, $n = 1, 2, \ldots, N$, equation (2.4) can be rewritten as

$$\pi_0^*(f) y_0^f + \sum_{z=1}^{N} \pi_z^*(f)(y_0^f - \sum_{n=0}^{z-1} \nabla y_n^f) = 0.$$

From $\sum_{z=0}^{N} \pi_z^*(f) = 1$, we have

$$y_0^f = \sum_{z=1}^{N} \pi_z^*(f) \sum_{n=0}^{z-1} \nabla y_n^f. \tag{2.18}$$

After we obtain the value of y_0^f, we can obtain the value of y_n^f by using that fact that $y_n^f = y_0^f - \sum_{z=0}^{n-1} \nabla y_z^f$ for $n = 1, 2, \ldots, N$. Once all the bias vectors of stationary optimal policies are computed, any stationary optimal policy with the optimal bias vector is a bias optimal policy. The following example illustrate this idea of finding a bias optimal policy.

Example 2.3 Consider an M/M/1/2 queue with two customer types, where $\lambda_1 = \lambda_2 = \mu = 1$, $r_1(0) = 10$, $r_1(1) = 0$, and $r_2(0) = r_2(1) = 5$. We start with a policy d that accepts both customer types at state 0, and rejects both at states 1 and 2. Then, by solving (2.5), we have that $d \in C^*$ and $r_2(0) = r_2(1) = \nabla y_0^d = \nabla y_1^d = 5$. According to Lemma 2.1(iii), there exist four stationary optimal policies for this problem, which are:

the policy f under which $f_0 = \{1\}$ and $f_1 = f_2 = \emptyset$.

the policy d under which $d_0 = \{1, 2\}$ and $d_1 = d_2 = \emptyset$.

the policy f' under which $f'_0 = \{1\}$, $f'_1 = \{2\}$, and $f'_2 = \emptyset$.

the policy d' under which $d'_0 = \{1, 2\}$, $d'_1 = \{2\}$, and $d'_2 = \emptyset$.

Note that policies f, d, f' and d' are also canonical policies, $f'(R) = f'$, and the policy $f'(R)$ is not a trunk reservation policy. Since, under any stationary policy π, $P^*(\pi) \times Q(\pi) = 0$, we can calculate the steady state probabilities under each of four stationary optimal policies:

$$P^*(f) = (0.5,\ 0.5,\ 0),$$
$$P^*(d) = (\tfrac{1}{3},\ \tfrac{2}{3},\ 0),$$
$$P^*(f') = (\tfrac{1}{3},\ \tfrac{1}{3},\ \tfrac{1}{3}),$$
$$P^*(d') = (0.2,\ 0.4,\ 0.4).$$

By solving (2.18), we have the bias vector under each policy as follows:

$$y^f = (2.5,\ -2.5,\ -7.5),$$
$$y^d = (\tfrac{10}{3},\ -\tfrac{5}{3},\ -\tfrac{20}{3}),$$
$$y^{f'} = (5,\ 0,\ -5),$$
$$y^{d'} = (6,\ 1,\ -4).$$

Since $y^{d'}$ is the optimal bias vector, the policy d' is the only bias optimal policy.

In Example 2.3, the state space and action sets are small and there exist only four stationary optimal policies. Thus, it is relatively easy to compute the bias vectors for all four stationary optimal policies and find the bias optimal policy. However, this method is not practical for the problems, in which there exist a lot of stationary optimal policies. Thus, in the following, we find the properties of a bias optimal policy. Then we use the properties to prove Theorem 2.6 and find the bias optimal policy for our problem more efficiently.

For discrete-time MDPs, Veinott [26] proved the existence of bias optimal policies and provided an algorithm for their computation. It is also well-known for discrete-time MDPs with finite state and action sets that bias optimal policies are canonical. In particular, one of the most popular versions of Howard's policy iteration algorithm generates a finite sequence of policies $\{f^n\}$ with the property that the pairs (v^{f^n}, y^{f^n}) lexicographically increase, the algorithm stops when the pair cannot be lexicographically improved by following the rules of the algorithm, and the last generated policy is canonical [22, p.461]. Since a bias optimal policy cannot be improved by the algorithm, it is canonical. This is also true for CTMDPs, because uniformization [22] preserves the average rewards, biases, and canonical equations. Thus, we have an important property of any bias optimal policy as the lemma below.

Lemma 2.10 *If g is a bias optimal policy for our problem, then $g \in C^*$.*

In Example 2.3, all four stationary optimal policies are canonical. Thus, Lemma 2.10 does not help us find a bias optimal policy more efficiently. However, the bias optimal policy d' in Example 2.3 satisfies the following lemma.

Lemma 2.11 *If g is a bias optimal policy and $r_j(n) = u_n$ for some $n = 0, 1, \ldots, N-1$ and some $j = 1, 2, \ldots, m$, then $g(n, j) = \{accept\}$.*

Proof The proof is based on contradiction. Let g be a bias optimal policy and $g(n, j) = \{reject\}$ at a pair (n, j) such that $n = 0, 1, \ldots, N-1$, $j = 1, 2, \ldots, m$, and $r_j(n) = u_n$. Consider two cases: (i) $n \in R^g$ and (ii) $n \in T^g$.

(i) Let $n \in R^g$. This means $\pi_n^*(g) > 0$. For any policy d, let $X_\infty(d) \sim \pi^*(d)$ be the limiting number of customers in the queue. Consider the policy f coinciding with g everywhere, except at (n, j). Of course, $f(n, j) = \{accept\}$. Since $g \in C^*$, Lemma 2.8 implies that $f \in C^*$. Observe that $\lambda_s^g = \lambda_s^f$ for $s \neq n$ and $\lambda_n^f = \lambda_n^g + \lambda_j$.

Since $\lambda_s^f \geq \lambda_s^g$ for all $s = 0, \ldots, N-1$, the well-known comparison results for birth-end-death processes, see, e.g., [20, Theorem 5.2.21], imply that $X_\infty(g) \leq_{st} X_\infty(f)$, where $\leq_{st}$ is the usual stochastic order. In particular, for any increasing numbers $c_0 < c_1 < \cdots < c_N$

$$\sum_{s=0}^{N} \pi_s^*(g) c_n \leq \sum_{s=0}^{N} \pi_s^*(f) c_n, \tag{2.19}$$

and the equality holds if and only if $\pi^*(g) = \pi^*(f)$. Since $\lambda_n^g \pi_n^*(g) = \mu_{n+1} \pi_{n+1}^*(g)$, $(\lambda_n^g + \lambda_j)\pi_n^*(f) = \mu_{n+1}\pi_{n+1}^*(f)$, and $\pi_n^*(g) > 0$, the equality $\pi^*(g) = \pi^*(f)$ is impossible. Thus, the strong inequality holds in (2.19). Apply this inequality to $c_0 = 0$ and $c_s = \sum_{z=0}^{s-1} u_z$, $s = 1, 2, \ldots, N$, and recall that $u_s = \nabla y_s^f = \nabla y_s^g$ for all $s = 0, 1, \ldots, N-1$, since $f, g \in C^*$. Formula (2.18) implies

$$y_0^g = \sum_{z=1}^{N} \pi_z^*(g) \sum_{s=0}^{z-1} u_s < \sum_{z=1}^{N} \pi_z^*(f) \sum_{s=0}^{z-1} u_s = y_0^f.$$

Thus, g is not bias optimal. This contradiction implies that case (i) is impossible.

(ii) Let $n \in T^g$. Then $n^g < n \leq N-1$. From (6) we have $N^* < n^g - 1 < N - 1$. From Lemma 2.5(ii), $u_s = \frac{v^*}{k\mu} \geq \max_{l=1,2,\ldots,m} r_l(s)$ for $s = n^g, n^g + 1, \ldots, N-1$. Thus, $r_j(n^g) \leq \frac{v^*}{k\mu}$. By Condition 2.1, $r_j(n^g) \geq r_j(n) = u_n = \frac{v^*}{k\mu}$.

So, $r_j(n^g) = r_j(n)$. Condition 2.1 implies that $r_j(s)$ is constant: $r_j(s) = \frac{v^*}{k\mu}$ for all $s = 0, \ldots, N-1$. Thus, Condition 2.2 holds and $j = i$.

Observe that $g(n^g, j) = \{reject\}$ and $n^g \in R^g$. According to case (i), $y_0^g < y_0^f$ for the policy f that coincides with g at all pairs (n', j') except the pair (n^g, j), where $f(n^g, j) = \{accept\}$. Thus, g is not bias optimal. This contradiction completes the proof of the lemma. ■

Proof of Theorem 2.6: From Lemma 2.10, a bias optimal policy is canonical. Then Lemmas 2.8 and 2.11 imply that a policy g is bias optimal if and only if for all $j = 1, 2, \ldots, m$

$$f(n, j) = \begin{cases} \{accept\}, & \text{if } n < M'_j; \\ \{reject\}, & \text{if } n \geq M'_j. \end{cases}$$

Since there exists only one canonical policy with the largest optimal control level for each customer type, the bias optimal policy is unique. In addition, because Blackwell optimal policies are bias optimal, the unique bias optimal policy is also Blackwell optimal. For discrete-time MDPs, this fact is in [22, Thm. 10.1.5], and for CTMDPs this follows from the discrete-time result and uniformization. The remaining statements follow from Theorems 2.3 and 2.4. ■

From Theorem 2.6, we know that the bias optimal policy g is the unique policy that selects the largest optimal control level for each customer type. Thus, after obtaining a canonical policy f from policy iteration algorithm, the bias optimal policy g can be found by the following formula:

$$g_n = f_n \bigcup \{j = 1, 2, \ldots, m : r_j(n) = u_n\} \quad \text{for } n = 0, 1, \ldots, N-1.$$

In other words, if we choose the canonical policy that always accepts the customer type j such that $r_j(n) = u_n$ at state n, then this canonical policy is the bias optimal policy.

2.5 Conclusion

In this chapter, we studied the optimal admission problem for an M/M/k/N queue with several types of customers and holding costs. We described the structures of stationary optimal, canonical, bias optimal, and Blacwell optimal policies. Among all stationary optimal policies, we proved the existence of an optimal TRP for our problem. Unlike the previous results, we provided examples in which a canonical policy may not be an ETRP. Therefore, for our problem, we formulated the necessary and sufficient conditions under which any stationary optimal policy is an ETRP. When there exist two or more canonical policies for our problem, we showed that the bias optimal policy is unique, is the canonical policy that assigns the largest optimal control level for each customer type, and is also Blackwell optimal.

In the future, it is interesting to look into similar problems with constraints of blocking probabilities for customers. Furthermore, it is also interesting to consider the optimal admission problem for a GI/M/k/N queue with several types of customers and holding costs. If the information for solving the problem for a GI/M/k/N queue is not enough, we are planning to find the conditions as weak as possible, under which we can obtain the similar results.

Bibliography

[1] Ahuja, R.K., Magnanti, T.L., and Orlin, J.B. (1993). *Network Flows*, Prentice Hall, New Jersey.

[2] Bather, J. (1973). Optimal decision procedures for finite Markov chains. Part III. *Advances in Applied Probability.* **5**, 541–553.

[3] Blackwell, D. (1962). *Discrete dynamic programming. Annals of Mathematical Statistics.* **33**, 719-726.

[4] Bountourelis, T. and Reveliotis, S. (2007). Optimal Node Visitation in Stochastic Digraphs. Preprint. School of Industrial & Systems Engineering. Georgia Institute of Technology.

[5] Derman, C. (1970). *Finite State Markov Decision Processes*, Academic Press, New York.

[6] Dynkin, E.B. and Yushkevich, A.A. (1979). *Controlled Markov Processes*, Springer-Verlag, New York.

[7] Fan-Orzechowski, X., and Feinberg, E. A. (2006). Optimality of randomized trunk reservation for a problem with a single constraint. *Applied Probability Trust.* **38**, 199-220.

[8] Fan-Orzechowski, X., and Feinberg, E. A. (2007) Optimality of randomized trunk reservation for a problem with multiple constraints. *Probability in the Engineering and Informational Sciences.* **21**, 189-200.

[9] Feinberg, E. A., and Reiman, M. I. (1994). Optimality of randomized trunk reservation. *Probability in the Engineering and Informational Sciences.* **8**, 483-489.

[10] Haviv, M., and Puterman, M. L. (1998). Bias optimality in controlled queueing systems. *Journal of Applied Probability.* **35**, 136-150.

[11] Helm, W.E., and Waldman, K.-H. (1984). Optimal control of arrivals to multiserver queues in a random environment. *Journal of Applied Probability.* **21**, 602-615.

[12] Howard, R.A. (1960). *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, MA.

[13] Johansen, S.G. (1977). Controlled arrivals in queueing systems with state dependent exponential service times. Dept. Operations Research, Aarhus University, Denmark.

[14] Kallenberg, L.C.M. (2002). Classification problems in MDPs, in: Z. How, J.A. Filar and A. Chen (Eds.), *Markov Processes and Controlled Markov Chains*, Kluwer, Boston, 151-165.

[15] Kallenberg, L.C.M. (2002). Finite state and action MDPs, in: E.A. Feinberg and A. Shwartz (Eds.), *Handbook of Markov Decision Processes*, Kluwer, Boston, 21–87.

[16] Knudsen, N.C. (1972). Individual and social optimization in a multiserver queue with a general cost-benefit structure. *Econometrica.* **40**, 515-528.

[17] Lewis, M.E., Ayhan, H., and Foley, R.D. (1999). Bias optimality in a queue with admission control. *Probability in the Engineering and Informational Sciences.* **13**, 309-327.

[18] McCuaig, W. (1993). Intercyclic digraphs, in: N. Robertson and P. Seymour (Eds.), Graph Structure Theory, *Contemporary Mathathematics*, *147*, Amer. Math. Soc., Providence, RI, 203–245.

[19] Miller, B.L. (1969). A queueing reward system with several customer classes. *Management Science.* **16**, 235-245.

[20] Müller, A., and Stoyan, D. (2002). *Comparison Methods for Stochastic Models and Risks.* John Wiley and Sons, New York.

[21] Naor, P. (1969). The regulation of queue size by levying tolls. *Econometrica.* **37**, 15-24.

[22] Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley and Sons, New York.

[23] Stidham, S. (1978). Socially and individually optimal control of arrivals to a GI/M/1 queue. *Management Science.* **24**, 1598–1610.

[24] Stidham, S. (1985). Optimal control of admission to a queueing system. *IEEE Transactions on Automatic Control.* **30**, 705–713.

[25] Tsitsiklis J.N. (2007). *NP*-hardness of checking the unichain condition in average cost MDPs. *Operations Research Letter.* **35**, 319–323.

[26] Veinott, A. F., Jr. (1966). On finding optimal policies in discrete dynamic programming with no discounting. *Annals of Mathematical Statistics.* **37**, 1284–1294.

[27] Yechiali, U. (1971). On optimal balking rules and toll charges in the GI/M/1 queueing process. *Operations Research.* **19**, 349–370.

[28] Yechiali, U. (1972). Customers' optimal joining rules for the GI/M/s queue. *Management Science.* **18**, 434–443.

[29] Yushkevich, A. A., and Feinberg, E. A. (1979). On homogeneous Markov decision models with continuous time and finite or countable state spaces. *Theory of Probability and its Applications.* **24**, 156–161.