Fuzzy Control of Queuing Systems

Runtong Zhang, Yannis A. Phillis and
Vassilis S. Kouikoglou

Fuzzy Control of Queuing Systems

With 77 Figures

Springer

Runtong Zhang, PhD

Institute of Information Systems, Northern Jiaotong University, Beijing, 100044, People's Republic of China

Yannis A. Phillis, PhD
Vassilis S. Kouikoglou, PhD

Department of Production Engineering and Management, Technical University of Crete, Chania 73100, Greece

British Library Cataloguing in Publication Data
Zhang, Runtong
Fuzzy control of queuing systems
1. Queuing theory 2. Automatic control 3. Fuzzy systems 4. Soft computing
I. Title II. Phillis, Yannis A., 1950– III. Kouikoglou, Vassilis S., 1961–
519.8′2
ISBN 1852338245

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

ISBN 1-85233-824-5 Springer-Verlag London Berlin Heidelberg
Springer Science+Business Media
springeronline.com

Typesetting: Camera-ready by authors
69/3830-543210 Printed on acid-free paper SPIN 10942444

Preface

The study of queuing control began in the 1960s and gave rise to a vast amount of literature. The basic tools of this field were drawn from dynamic programming and the theory of Markov processes. Such issues as the operational characteristics of controlled systems as well as questions of existence of optimal controls and their structural properties were and still are being studied.

The experience of four decades, however, is not encouraging. Most queuing control problems cannot be solved explicitly because of their complexity and enormous computational demands. Queuing control is mathematically more demanding than the analysis of queues, which also reaches its limits when non-Markovian problems are studied.

To overcome analytical difficulties, researchers turned to approximate or heuristic methods. Lately fuzzy logic was employed to problems of queuing control.

This book provides a number of results in queuing control from the point of view of fuzzy logic. Fuzzy control is an effective approach in nonlinear or large-scale systems control, especially when mathematical models are difficult to obtain or do not exist at all. It turns out that fuzzy control is computationally efficient and, in conjunction with analytical results, precise.

A number of control problems will be presented, which were developed by the authors in the past decade. This is the first systematic effort of solving queuing control problems using the tools of fuzzy logic.

The material of this book can be useful to advanced undergraduate and graduate students. Also, researchers and practitioners in the field of queuing control, systems analysis, manufacturing, and communications may benefit from it.

The material is organized into nine chapters. The introductory chapter outlines the book and discusses background. Chapters 2 and 3 provide technical background on fuzzy logic and fuzzy control. Chapters 4–7 cover fuzzy queueing control. These chapters are organized along the lines of problem description, architecture of the fuzzy logic controllers, and numerical examples. Comparisons are provided whenever feasible. Chapter 8 presents applications to the Internet. Chapter 9 concludes with suggestions for further investigations. The Appendix provides a brief introduction to Markov queuing models and simulation, which were used to validate the performance of the fuzzy algorithms. A list of references is given at the end, which is by no means exhaustive.

We are indebted to a number of people who assisted us in the writing of the book. Several anonymous referees gave us invaluable advice in the course of our research.

We thank Nili Phillis for proofreading the manuscript.

Runtong Zhang would like to thank his colleagues at the Technical University of Crete, the Beijing Jiaotong University, the Nokia Research Center, and the Swedish Institute of Computer Science for supporting him to complete this project. Last but not least, he is thankful to his wife Xiaomin, daughter Ziwei, and son Zijian for their constant encouragement and patience during this project.

Vassilis Kouikoglou thanks his wife Vasso for her understanding and support during the writing of this book.

Beijing, China	*Runtong Zhang*
Crete, Greece	*Yannis A. Phillis*
Crete, Greece	*Vassilis S. Kouikoglou*

Winter 2004

Contents

1 Introduction

1.1 Queues and Queuing Theory

Queues are ubiquitous. We wait in cars, banks, hotels, supermarkets, box offices, airports, hospitals, and so on. These are examples of visible queues. In fact, queues of voice calls or data packets in communication channels are common but invisible. Queues are often undesirable because they cost time, money, and resources. They exist because the service resources are not sufficient to satisfy demand. This is because of a number of reasons. Servers may be unavailable because of space or cost limitations, or it may not always pay to provide the level of service necessary to prevent waiting. The burstiness of traffic in communication lines or computer networks is also a reason why queues cannot be easily avoided.

Queuing theory uses mathematical tools to predict the behavior of queuing systems. Predictions deal with the probability to have n customers in the system, mean length of queues, mean waiting time, throughput, and so on. A queuing system consists of a stream of arriving customers, a queue, and a service stage. To model such a system, the following basic elements are needed:

- A stochastic process describing the arrivals of customers
- A stochastic process describing the service or departures of customers
- Number of servers (m)
- System capacity (K)
- Size of customer population (N)
- Queue discipline

A queuing system is described using the notation A/B/m/K/N/Z, where A and B specify the distributions of the interarrival and service times, respectively: M (memoryless) denotes the exponential distribution, E_k stands for the k-Erlang distribution, G is used for an arbitrary distribution, D for deterministic times, and so on. Finally, Z is the queue discipline: FIFO (First In First Out), LIFO (Last In First Out), and so on.

If any of the descriptors K, N, and Z is missing, then we assume that $K = \infty$, $N = \infty$, and Z = FIFO, respectively.

1.2 Models of Queuing Control

1.2.1 Introduction

The majority of the early queuing optimization problems were static, where the system characteristics would not change over time. Static optimization, however, cannot meet the demand for dynamic allocation of customers or resources in complex manufacturing or communication systems. One has to move from the descriptive, where all the parameters are given, to the prescriptive or normative, where parameters change.

Design problems, usually with the aid of descriptive models and some performance measure, provide among others the optimal mean arrival rate λ, mean service rate μ, and system capacity K. Control, on the other hand, is dynamic and provides actions according to the state of the system. Customers can be directed to different servers depending on the queue size, arriving customers may be denied entry, waiting customers may be denied service although servers are available until the queue reaches a threshold size, and so on.

Control problems are mostly solved using dynamic programming techniques. Heuristic algorithms are not uncommon in complex situations. This book focuses on a new approach using fuzzy logic. Fuzzy queuing control seems to be a promising method where dynamic programming does not work. Indeed, dynamic programming handles simple and mathematically well-posed problems. But the majority of practical problems do not have nice mathematical descriptions or they are so complicated as to defy analysis. Fuzzy logic seems to be well suited to fill this gap. This is so for a number of reasons, as we shall see later. Here it suffices to say that fuzzy logic uses words to develop models and perform computations emulating an experienced operator. It is thus able to handle highly nonlinear problems and provide efficient control policies.

The control problems of the book belong to the following general categories:

1. *Control of the number of servers.* Servers are removable and may be turned on or off according to the state of the system. The varying number of active servers must be determined.
2. *Control of the service rate.* This category generalizes (1). We change the service process by varying the service rate rather that modifying the number of servers.
3. *Control of the queue discipline.* The order of service is determined among different classes of customers, and an allocation of customers to servers is made.
4. *Control of the admission of customers.* The arrival rate can be modified, customers may be denied entry, or customers control the decision for entry.

Next we give a brief review of each category.

1.2.2 Control of the Number of Servers

The systems in this category are usually called queuing systems with removable servers or with vacations. The servers may be unavailable over certain intervals of time, and a decision should be made about the time of activity for each one. The objective is to minimize the expected system cost. Vacation models are motivated by problems in which the vacation time is used by other jobs, which could even belong to another system, so that the idle time of a server is not necessarily lost. The notion of vacation can be generalized to various queuing problems in which the service station is subject to breakdowns.

Three types of policies have been discussed in the literature.

- *N-policy*: The state of removable servers depends on the number of customers present in the system. A common type of N-policy is called (ν, N)-policy, with $0 \leq \nu \leq N < +\infty$, according to which the server is turned on when N customers are present and the server is turned off when it terminates a service with ν customers left in the system.
- *D-policy*: The state of removable servers depends on the total amount of work in the system.
- *T-policy*: An active server goes on serving the queue as long as there is at least one customer in the system, but when the system empties, the server becomes unavailable for some length of time (a vacation).

Bibliographical Notes

Yadin and Naor (1963) were the first to introduce queuing systems with removable servers applying a $(0, N)$-policy. Most of the subsequent work was devoted to single-server systems. Using an average criterion, Heyman (1968) proved that the optimal N-policy is either $(0, N)$ with $0 \leq N < +\infty$ or $(0, 0)$, i.e., always an *exhaustive policy*. For a discounted criterion, the optimal stationary operating policy depends on the starting state, which for simplicity is $(0, 0)$; that is, the server is off and there are no customers in the system. In this case, Heyman (1968) and Bell (1971) proved that the optimal N-policy is either $(0, N)$ with $0 \leq N < +\infty$ (as in the average case) or $(0, +\infty)$ or $(0, N)^*$ with $1 \leq N < +\infty$. This policy turns on the server when N customers are present, and the server stays on thereafter. Kimura (1981) used a diffusion approximation model to determine explicit solutions for a problem with a discounted criterion. Bell (1973) introduced a vacation model with priority queue; Teghem (1987) and Wang and Huang (1995) considered the same problem for M/G/1 and $M/E_k/1$ queues, respectively, with finite queuing capacity; Makis (1984) studied the batch service problem; Altman and Nain (1993) proposed a new model with a removable server. Boxma (1976) investigated D-policies. Heyman (1977) studied the T-policy for the M/G/1 queue.

Unlike the single-server case, very few studies have been devoted to multiserver queues with removable servers. Bell (1975) discussed the difficulties of this problem. He showed that an optimal policy is not necessarily an *efficient policy*, which is defined as an operating rule that never allows the number of available servers to be larger than the number of customers present. Bell (1980) further investigated an M/M/2 system under an N-policy, and Winston (1978) considered an M/M/m queue with removable servers in which the arrival rate depends on the number of customers. The characterization and the explicit determination of optimal control policies are still open problems.

1.2.3 Control of the Service Rate

A queuing system with variable rate, as the name suggests, is one where the mean service rate may be chosen from a set of finite service rates $\{\mu_k \mid k \in K\}$, where K is a fixed set of service types. The service time of a customer is a random variable independent of the arrival process and the previous service times. This problem is a generalization of the removable server problem in Section 1.2.2. Two types of control policies are considered in this category, N-policy where the state of the system is the queue length and D-policy where the state of the system is the workload. The set K may be countable or uncountable.

Most of the work in this category is devoted to countable service rates under an N-policy, and the following policies have been introduced:

1. *Hysteretic policy*: Whenever the queue size reaches a value $\overline{R}_{k+1}$ from below while the service in progress is of type k, the next service will be of type $k+1$; whenever the queue size reaches a value $\underline{R}_k$ from above while the service is of type $k+1$, the next service will be of type k. The parameters $\overline{R}_k$ and $\underline{R}_k$ are increasing in k and $\overline{R}_{k+1} \geq \underline{R}_k$. The length of the hysteresis loop is $H_k = \overline{R}_{k+1} - \underline{R}_k$. It is possible that $\overline{R}_k = \overline{R}_{k+1}$ and $\underline{R}_k = \underline{R}_{k+1}$; in which case, service type k is not used in the policy. When there are two available service rates including zero, the optimal (ν, N)-policy is $\underline{R}_1 = \nu$ and $\overline{R}_2 = N$.
2. *Monotone hysteretic policy*: A hysteretic policy is said to be *increasing* if the service rates satisfy $\mu_{k+1} \geq \mu_k$ and *decreasing* if $\mu_{k+1} \leq \mu_k$.
3. *Uniform hysteretic policy*: A uniform hysteretic policy results when H_k is constant $\forall k$.
4. *Connected policy*: A connected policy is a uniform hysteretic policy with $H_k = 1$, $\forall k$.

Bibliographical Notes

Yadin and Naor (1967) and Gebhard (1967) were the first to introduce the hysteretic policy. They provided some useful properties of the steady-state distribution of the queue length for systems with an exponential server. An extension of such properties is given in the work of Sobel (1982). Federgruen and Tijms (1980) described a method for recursively computing the steady-state queue length distribution. Under an average criterion, Crabill (1972) and Lippman (1973) proved the optimality of connected increasing policies for an exponential server without switching costs. Similar results were obtained by Sabeti (1973). This system may give a decreasing optimal policy if its queue capacity is limited (e.g., Schassberger 1975 and Beja and Teller 1975). Crabill (1974) proved the existence of an optimal increasing hysteretic policy for exponential distributions of service time with switching costs. Rath (1975) dealt with asymptotic results in heavy traffic conditions in a GI/G/1 queue, that is, a system whose interarrival times are independent of each other and independent of the service times. An interesting paper by Lu and Serfozo (1984) analyzed an M/M/1 queuing decision process in which the finite set of decisions concerns not only the service rate but also the arrival rate. Finally, Weber and Stidham (1987) extended these results to multiserver systems.

For countable sets of service rates, the work of Jo and Stidham (1983) gave properties of both N-policies and D-policies. D-policies may also be found in the papers of Tijms and van der Schouten Duyn (1978) and Cohen (1986).

N-policies and D-policies also apply to uncountable sets of service rates. Most of the results in the literature are derived for zero switching costs. For an exponential server, Lippman (1975) proved the existence of a monotone increasing optimal policy with a discounted cost criterion. Jo (1983) extended Lippman's results to a more general cost structure as well as an average cost criterion. Gallish (1979) studied general service time distributions. The paper by Zacks and Yadin (1970) is among the few studies considering nonzero switching costs. All the systems mentioned in this paragraph have a single server. Undoubtedly, systems with multiple servers are of greater importance, but the analysis is mathematically hard. Rosberg *et al.* (1982) considered the optimal service rate control to a tandem queuing system. Doshi (1978) considered M/G/1 queues with control of the workload (D-policy) for uncountable sets of service rates and proved the existence of an increasing connected optimal policy for both discounted and average criteria. Deshmukh and Jain (1977) integrated design and control with variable service rates. Finally, Stidham and Weber (1989) established the monotonicity of optimal policies for combined control of arrival and service rates.

1.2.4 Control of the Queue Discipline

Think of a queuing system where different classes of customers arrive for service. The arrivals may line up in different queues, and the servers may offer diverse services. We say that the system has heterogeneous customers, queues, and servers. Think also of a dynamic assignment of customers to idle servers so as to minimize an expected cost. This is a problem of control of the queue discipline. There is an abundant literature in this area for single-server systems. Multiserver systems may have parallel, tandem, or tandem-parallel servers.

Perhaps the most well known strategy in single-server scheduling problems is the *$c\mu$ rule*. This rule states that when service times are exponential or geometric, serving the customer with the largest $c_i\mu_i$ minimizes the expected discounted cost, where c_i is the holding cost rate of customer i and μ_i is its service rate.

Bibliographical Notes

Various aspects of the $c\mu$ rule have been examined in the literature (Cox and Smith 1961; Klimov 1974; Harrison 1985; Baras *et al.* 1985; Buyukkoc *et al.* 1985; Righter and Shanthikumar 1989; Shanthikumar and Yao 1992).

Many results have been reported on optimal routing of customers to multiple servers. Weber (1978) and Ephremides *et al.* (1980) show that if a system consists of multiple identical M/M/1 queues in which the queue sizes are observable at any time, then the expected discounted cost is minimized by the *shortest queue* policy, which routes a new arrival to the queue with the shortest queue length. For a similar system, Whitt (1986) provides counterexamples of service time distributions for which it is *not* optimal to always join the shortest queue and, if the elapsed service time of customers in service is known, the long run average delay is *not* always minimized by customers joining the queue, which minimizes their individual expected delay. Hlynka *et al.* (1994) show that, under certain conditions, the smart customer can lower his expected sojourn time in the system by waiting and observing rather than immediately joining the shortest queue. Lin and Kumar (1984) and Walrand (1984) prove that the optimal policy is of the *threshold type* for M/M/2 systems with heterogeneous servers. Hajek (1984) considers the case of two interacting nonidentical service stations.

Heterogeneity in service functions was investigated in Xu *et al.* (1992), where a certain kind of customer can be served only by a certain kind of server. Optimality of the threshold policy was established. However, optimal control of queuing systems with server heterogeneity in both service rates and service functions is still an open problem.

Bell's paper (1980) on the optimal operation of an M/M/2 queue with removable servers can be viewed in the context of optimal routing problems. Chow and Kohler (1979) and Seidmann and Schweitzer (1984) studied dynamic routing of customers among multiple servers in queuing systems arising in manufacturing networks. Recently, Phillis and Kouikoglou (1995) proposed a general entropy approach for the problem of queue discipline control. Relevant work can be found in Baras and Dorsey (1981), Rosberg and Kermani (1989), Courcoubetis and Varaiya (1984), among others.

1.2.5 Control of the Admission of Customers

This category is so named because the system can either accept or reject an arriving customer or, in some cases, the arriving customer may refuse to join the system. Usually the objectives are to determine the optimal admission control policies to maximize the expected profit.

It is inevitable to face different criteria of either *individual* or *social* optimization when one deals with problems of admission and routing control. Most of the routing control problems briefly discussed in Section 1.2.4 use a social optimization criterion. The former criterion depends on the customer's own benefits, and the latter views system performance as a whole. The decrease in utility imposed on future customers by an arriving customer's decision to enter the system is often referred to as the *external effect* (as opposed, to the *internal effect*, which is associated with a customer's delay). It is believed that, because of the presence of externality, the policy implemented by self-interested individuals does not lead in general to the best social outcome. These terms have aroused considerable interest among economists and operations researchers. To bring the two policies into agreement, it is often proposed to impose an admission toll or entrance fee on customers who decide to join.

Optimal policies for individual customers are usually easy to obtain and take simple and explicit forms, whereas socially optimal policies, which are of greater practical importance, often defy simple analysis.

For M/M/1 queues, the individually and socially optimal policies are of the *threshold* (or *critical-number* or *control-limit*) type, but the critical numbers that characterize the two policies are not necessarily equal, and typically, the socially optimal critical number is less than its individual counterpart. Similar results exist for GI/M/1 and GI/M/m queuing systems.

Bibliographical Notes

Naor (1969) first showed that the optimal policies for M/M/1 queues are of the threshold type. The individually vs. socially optimal problem was discussed by Bell and Stidham (1983) and Hassin (1975). The determination of threshold values for involved systems is difficult (see, for example, van Nunen and Puterman 1983 and Xu and Shanthikumar 1993). Miller (1969) studied an M/M/m/K system, i.e., a multiserver system with losses, with a finite number of customer classes, each having a different reward and no holding cost. Lippman and Stidham (1977) extended Miller's model to one with infinite capacity and linear holding costs. In a subsequent paper, Stidham (1978) considered a batch-arrival GI/M/1 system allowing for a nonlinear holding cost rate. Langen (1982) extended the results of Stidham (1978) to GI/M/m systems. Johansen and Stidham (1980) proposed a general model for control of arrivals to a stochastic input–output system, which views several single-server versions as special cases. Blanc *et al.* (1992) examined optimal control of admission to an M/M/m queuing system with one controlled and one uncontrolled arrival stream. Surveys on admission control may be found in Stidham (1985) and Stidham and Weber (1993).

1.3 Methodologies of Queuing Control

1.3.1 Dynamic Programming

Dynamic programming is a powerful tool in the field of dynamic control. Its fundamental tenet is so obvious that it may sound trivial. Indeed its informational content is straightforward, but its power in reducing the number of computations when an optimal policy is sought is enormous. This tenet is called the *principle of optimality* and states that if a path P_{123} in a multistage decision problem is optimal from decision 1 to decision 3, then P_{23} is the optimal path from 2 to 3. As the late Richard Bellman, the inventor of the principle put it:

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

When the state of a queuing system is known at each time instant, we say that we have complete observation. In this case, dynamic programming, in principle at least, could solve any control problem. The computations, however, even then could become prohibitive because of the *curse of dimensionality*, as Bellman called the basic drawback of dynamic programming.

The optimal cost is evaluated moving from stage to stage using some quantization levels that represent the possible states at each stage. Assume that we need to evaluate the optimal cost $C^*_{N-k,\,N}$ from stage $N-k$ to the final stage N. Then we need to store the values $C^*_{N-(k-1),\,N}$, which resulted from a previous iteration. For a first-order system with m quantization levels, we need m storage places. However, for an nth order system with m quantization levels for each state variable, we need m^n storage places and this number can easily become prohibitive even for simple problems.

Now think of a process observed as it evolves at times $t_0, t_1, \ldots$ or simply $k = 0, 1, \ldots$. The process has a countable set of possible *states*. After observing the state, an *action* is chosen from a finite set of possible actions and the system jumps to another state with a given probability called the *transition probability*. The new state may depend on the previous state and the last action taken or it may depend on the history of past states and actions. Let X_k be the state and a_k the action taken at time k. Given a course of actions, the state of the process is X_k, $k = 0, 1, \ldots$. If X_k is a Markov chain, that is, a process for which the state transition probabilities satisfy $P(X_k \mid a_{k-1}, X_{k-1}, a_{k-2}, X_{k-2}, \ldots, a_0, X_0) = P(X_k \mid a_{k-1}, X_{k-1})$, then we say that we have a *Markov Decision Chain* (MDC). If the time is continuous, then we speak of a *Markov decision process* when the times between transitions are exponential random variables, or a *semi-Markov decision process* when these times have arbitrary distributions.

To describe an MDC we need, in addition to the *state space S*, which contains all the states of the process, and the set A of actions, a *cost* or *reward* associated with each action and state, which is minimized or maximized upon choosing a course of actions or decision rules. Optimization may take place over a *finite horizon*, where $t_k \in T \subset [0, \infty)$, $\forall k$, or an *infinite horizon*, where $T = [0, \infty)$. Often discounted costs are used because future costs have a small present value. Other costs include the finite horizon expected cost, the long run expected average cost, and so on.

We define a *policy u* to be a sequence of decision rules for choosing actions. Each decision rule d_k may depend on the current time t_k and the history of previous states and actions. A particular class of policies that are optimal for most cost structures is the class of *Markov policies*, which depend only on the current time and current state of the system and not on the past. Thus, a Markov policy $u = (d_0, d_1, \ldots)$ is a sequence of mappings from S to A such that $d_k(n) \in A$ is an action on the queuing system for a state $X_k = n \in \mathrm{S}$ and a time $t_k \in T$. A Markov policy is called *stationary* if $d_k(n)$ depend only on n and not on time. This means that whenever the system is in state n, the controller employs a decision rule independently of the current time and the history of the system. Thus, for a stationary policy u, we have $u = (d, d, \ldots)$ and the resulting decision process is an MDC.

A policy that specifies a unique action $a = d_k(n)$ for a given current state n and time index k is called a *deterministic Markov policy*. If the controller chooses action a with a probability $P_k(n, a)$, then the policy is called a *randomized Markov policy*.

This is in general the setting of dynamic programming. The books by Walrand (1988) and Kitaev and Rykov (1995) deal with the application of dynamic programming to the control of queuing systems. As has already been mentioned, this approach, powerful as it may be, has serious shortcomings. Later we shall see how we overcome such shortcomings using fuzzy logic.

1.3.2 Heuristic Algorithms

Dynamic programming is an efficient mathematical method that works in certain areas. Most realistic problems, however, are not amenable to mathematical techniques because either the resulting dimensions are prohibitive or simply our tools cannot model reality reliably. Heuristics may then be used with varying degrees of success.

Heuristic techniques do not have specific rules although they deal with rules that aim at finding precise and computationally efficient solutions. Heuristics is an approach that relies on experience, intuition, and a few general strategies to provide solutions to complex problems. Such problems may have in principle mathematical solutions, but their dimensions may be such that the hope of numerical results is nil. Take, for example, the problem of analyzing a simple serial production line with intermediate storage. The dimension of this problem increases geometrically with the number of machines and in a multiplicative fashion with the size of the storage. The solution is straightforward using Markov chains, but the numerical analysis is impossible for lines with more than three machines and two storage spaces. Heuristics has solved this problem reasonably well in many cases. Problems of this sort are the rule rather than the exception in many fields including control of queues. A heuristic strategy may lead to the final solution following a step-by-step procedure as it is done in the control of queues, or a hierarchy of subproblems may be developed and the solution proceeds from subproblems to the whole. Some approaches of the production line problem use this strategy. Alternatively, a solution may be obtained by another heuristic or approximate analysis that is modified to achieve better accuracy or computational speed. Finally, incoming information may be used to redirect the solution as it is done in a game of chess.

These approaches do not necessarily stand on their own but may be combined together or may even use mathematical or simulation techniques at some stage. A hierarchical heuristic search may use, for example, an optimization algorithm to solve a subproblem and then validate the efficacy of the solution using simulation.

1.3.3 Fuzzy Logic Control

In 1987, over 2000 patents were issued in Japan related to the technological applications of *fuzzy logic*. Fuzzy logic theory and technology are among the fastest developing areas in science and engineering. Where traditional mathematics was unable to solve complex practical problems, fuzzy logic is filling the gap mostly with tremendous success: home appliances, aircraft control, production systems, medical applications and so on.

Fuzzy control, which is a combination of control theory and fuzzy logic, is probably the spearhead of fuzzy logic. The number of publications as well as applications in this field has been growing by the day.

Aristotle is the founder of logic and deductive reasoning. Much later, a two-valued logic was developed, where a proposition is either true or false but not both, and its epitome, Boolean algebra, is applied today in the analysis and design of digital systems. More recently, Lukasiewicz proposed a three-valued logic where a proposition might assume three values: 1 (true), 2(false), and 1/2 (neuter). Gray areas were introduced in logic, and finally in 1965, Lotfi Zadeh launched fuzzy logic by assuming that there are propositions with an infinite number of truth values in infinitely varying degrees. Any logic then is just a subset of fuzzy logic. There are two extreme values, 1 (totally true) and 0 (totally false), and a continuum in between that justifies the term “fuzzy.”

Fuzzy logic, like probability theory, deals with uncertainty, but unlike probability, this uncertainty is masked in semantic and subjective ambiguity. Different people, for instance, judge and evaluate reality differently. Fuzzy logic, again unlike probability theory, deals with degrees of occurrence, whereas the latter deals only with occurrence. Take, for example, the sentence “there is a 0.15 probability to get a good grade in queuing control.” The number 0.15 is a probability, but the event “good grade” is fuzzy; it is not black or white.

As Zadeh said, fuzzy logic is computation with words and

Fuzzy logic’s primary aim is to provide a formal, computationally-oriented system of concepts and techniques for dealing with modes of reasoning which are approximate rather than exact.

Thus, fuzzy logic deals with degrees of truth that are provided in the context of fuzzy sets by what is called membership functions. To be able to perform logical, albeit fuzzy, reasoning, fuzzy operators such as OR, AND, IF, and THEN ought to be defined.

Fuzzy control systems are rule-based systems in which a set of rules, called fuzzy rules, define a control mechanism to adjust the system. Figure 1.1 shows the block diagram of a fuzzy logic controller for queues that comprises four principal components: a fuzzification interface, a knowledge base, an inference engine, and a defuzzification interface.

The output of the fuzzy logic controller in Figure 1.1 is used to tune the system parameters according to some predefined program based on the state of the system. This control mechanism is adaptive.

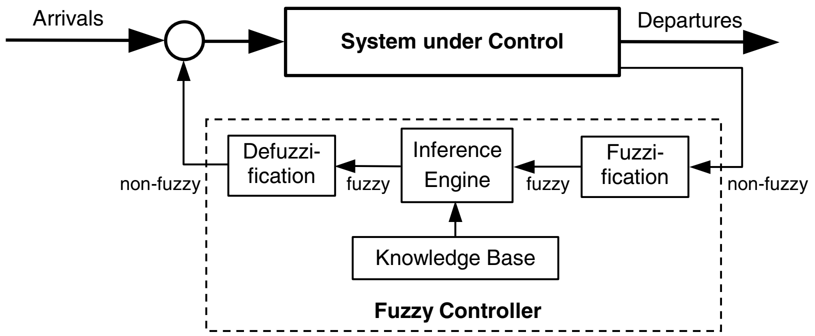


Figure 1.1. Block diagram of a fuzzy logic controller for queues.

The aim of fuzzy control systems is normally to substitute for or replace a skilled human operator with a fuzzy rule-based system. Greater details on fuzzy logic and fuzzy control will be given in Chapter 3. For the moment, we provide a brief introduction to the subject.

1.4 Control of Queuing Systems

Fuzzy control of queuing systems is an application of fuzzy logic theory to the control of queues. A fuzzy logic controller provides a decision mechanism that dynamically determines the parameters, patterns, and policies of a queuing system in some specified optimal sense. Fuzzy queuing control is a combination of artificial intelligence, operations research, and optimal control. The results to be given in this work are among the first in the literature.

Queuing theory has already had a long history and has been used to solve practical problems in manufacturing, communication, and other fields. In the last four decades, there has been an increasing interest in the study of queuing control that has provided queuing theory with renewed vigor. Most of the work in this field uses conventional stochastic control techniques, which, although often successful, have severe computational limitations as already mentioned. Recently, fuzzy logic has made remarkable progress in many applied control problems. Now has come its time to provide powerful results in queuing control.

There are many reasons why fuzzy control is a good and promising choice to control queues.

1. Conventional control theory is very well developed, but its success depends heavily on the quality of the model of the controlled system. Queuing systems are often not amenable to mathematical descriptions, or such descriptions are too complicated to be of any value. Fuzzy control does not require a mathematical model of the system under control. In fact, fuzzy control is suitable in cases of complex systems or ill-defined processes, as long as it is equipped with an operator's experience, knowledge, and learning. Thus, fuzzy logic appears to be an excellent candidate in queuing control.
2. Fuzzy control is well suited to deal with highly nonlinear systems such as queuing systems. It appears then that fuzzy control, in principle at least, should be effective.
3. Analytical solutions for the control of queues exist only for very simple cases. Policy reinforcement and heuristic algorithms determine the optimal control policies for more complicated queuing systems such as tandem, parallel, and tandem-parallel ones. However, the computational demand increases exponentially with the dimension of the system. Fuzzy control seems to be a promising alternative. Conventional policy reinforcement algorithms choose the best actions by eliminating nonoptimal ones, and fuzzy control does this by directly determining the best action. The larger the system scale, the more obvious this advantage becomes.
4. There are queuing systems whose arrival and service rates are described by fuzzy linguistic variables. We call such systems fuzzy queues. In certain situations, they represent reality well. Fuzzy control is the best if not the only choice to dynamically control such systems.

1.5 Issues of Fuzzy Queuing Control

A number of issues arise when fuzzy control is used to make dynamic decisions in queuing systems. We briefly discuss them here.

Optimality

The word “optimal” will be used to qualify the policies determined by the fuzzy controllers. A question, however, arises about how to test optimality. Optimality is one of the most difficult problems in the fuzzy control field, and although it has attracted a lot of attention, the answers are still unsatisfactory. This is because of the nature of fuzzy logic and the common absence of mathematical models of the controlled system. Optimality is pursued by emulating an expert operator. This is the best that can be done in the context of fuzzy logic. It is worth noting, however, that in all cases of queuing control where a mathematically optimal solution is known, the fuzzy controllers yield precisely the same optimal solutions.

Stability

Stability is another open issue in fuzzy control. The lack of analytical descriptions is apparent here too. On the other hand, stability can be achieved by properly training the fuzzy rule bases and the membership functions. A rule of thumb is to choose continuous universes of discourse. The final answer about stability cannot be provided a priori, however. This is done with the aid of simulation, which has shown that all algorithms in this book are stable.

Membership Functions

The form of most membership functions is straightforward. On occasion, however, this form is involved and is derived after the inner workings of each queuing system are brought to light. This is done automatically by the fuzzy controllers in a self-tuning manner.

Freedom from Analytical Models

It should be stressed that all adjustments rely on experience and knowledge of the operator. The fact that fuzzy logic works where conventional mathematics does not is because of this feature. We try to emulate a human operator performing complex tasks such as landing an airplane, parking a car, or prescribing an effective dose of chemotherapy. This, however, is also the main problem of fuzzy logic. We cannot prove rigorously optimality and stability, among others. Freedom from mathematical models has its blessings and curses.

It is important to note that the assumption of Poisson arrivals and exponential service could be dropped when fuzzy control is used. Poisson and exponential assumptions usually lead to Markov or semi-Markov decision processes that have a solid theoretical background. Many practical systems, on the other hand, do not behave in Markovian ways, especially when complicated geometries of networks are present, and thus no precise mathematical solutions are possible. Fuzzy logic holds a promise there.

1.6 Applications of Queuing Control

The areas of application of queuing control abound: communication, transportation, manufacturing, urban systems, and so on. The goal is always to share a limited resource while at the same time a performance measure is optimized. Customers could be information packets waiting to be processed through a channel, aircraft waiting for an available runway, workpieces in a factory to be routed to a machine, and patients waiting for an ambulance. Obviously, in some cases, proper allocation of resources is not only an economical problem but also, literally, a matter of life or death.

There are several possibilities of queue control. In Figure 1.2, we have one server and one queue. Customers arrive at the queue awaiting service. A controller may decide which customers may enter the system and which the expected rate will be.

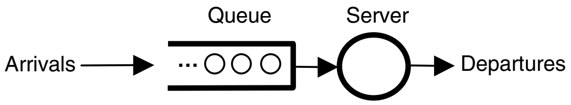


Figure 1.2. Single-queue single-server.

A common manufacturing system comprises a production line with N machines (servers) and $N + 1$ buffers (queues) as shown in Figure 1.3. The controller may adjust the arrival rate of workparts (customers) and the production rates of the machines (service rates). Design actions could define the optimum size of each buffer given a number of constraints as well as the best allocation of repair resources when machines break down.

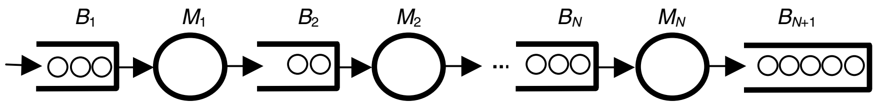


Figure 1.3. Tandem production line.

Another system in Figure 1.4 has m queues and m servers that, in general, have different mean service rates. The controller decides which arriving customer is to be routed where.

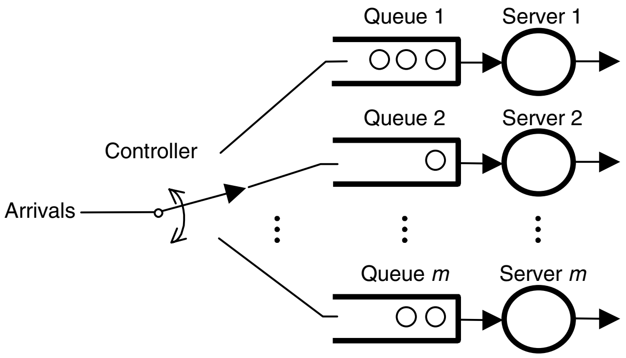


Figure 1.4. Parallel queues.

A system with *m* queues, as shown in Figure 1.5, may have one server and the controller decides which queue to serve next.

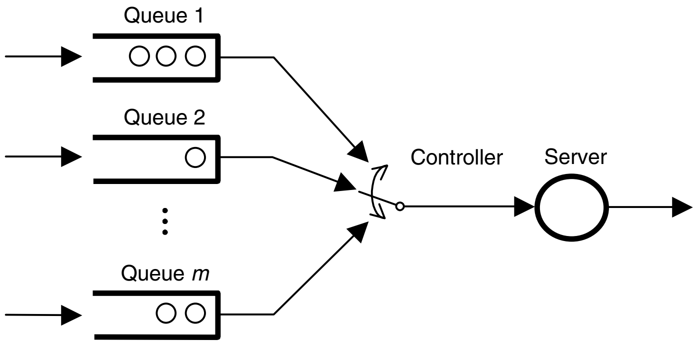


Figure 1.5. Multiple queues one server.

Other systems will be examined in Chapters 4–8.

2 Fuzzy Logic

2.1 Fuzzy Sets

Fuzzy set theory and its attendant *fuzzy logic* were developed by Lotfi Zadeh in 1965 to handle semantic and subjective ambiguity. In classical logic, the number 300 is an integer, whereas 300.7 is not. The same number, however, could be considered large, small, very large, very small, and so on depending on context and subjective opinion. Therefore, the number 300 could be considered large to a certain degree, very large to another, and so on. We then have various *linguistic values* of one *linguistic variable*, which are true to some degree. This degree, subjective as it may be, varies from 0 to 1.

In classical set theory, an element of a set either belongs or does not belong to the set. In fuzzy set theory, an element belongs with a *membership grade* in the interval [0, 1]. All membership grades together form the *membership function*. A classical set is often called *crisp* as opposed to fuzzy.

Definition 2.1. A set is a collection of elements or members. A set may be an element of another set.

Definition 2.2. Let X be a set of elements x. A fuzzy set A is a collection of ordered pairs $(x, \mu_A(x))$ for $x \in X$. X is called the *universe of discourse* and $\mu_A(x)$: $X \to [0, 1]$ is the membership function.

The function $\mu_A(x)$ provides the degree of fulfillment of x in X. When X is countable, the fuzzy set A is represented as

$$A = \mu_A(x_1)/x_1 + \mu_A(x_2)/x_2 + \ldots + \mu_A(x_n)/x_n.$$

This is a common notation in the context of fuzzy sets. It simply states the elements x_i of X and the corresponding membership grades.

Example 2.1

Consider the temperature of a patient in degrees Celsius. Let $X = \{36.5, 37, 37.5, 38, 38.5, 39, 39.5\}$. The fuzzy set A = "High temperature" may be defined

$$A = \{\mu_A(x)/x \mid x \in X\}$$

$$= 0/36.5 + 0/37 + 0.1/37.5 + 0.5/38 + 0.8/38.5 + 1/39 + 1/39.5,$$

where the numbers 0, 0.1, 0.5, 0.8, and 1 express the degree to which the corresponding temperature is high.

Definition 2.3. The support of a fuzzy set A is the crisp set of all elements of X with nonzero membership in A, or symbolically

$$S(A) = \{x \in X \mid \mu_A(x) > 0\}.$$

Example 2.2

Take Example 2.1. $S(A) = \{37.5, 38, 38.5, 39, 39.5\}$.

Definition 2.4. The set of all elements of X with membership in A at least α is called the α-level set, or symbolically

$$A_\alpha = \{x \in X \mid \mu_A(x) \geq \alpha\}.$$

Definition 2.5. A fuzzy set A is said to be *convex* if the membership function is quasiconcave; that is, $\forall x_1, x_2 \in X$, and $\lambda \in [0, 1]$, the following is true:

$$\mu_A[\lambda x_1 + (1 - \lambda)x_2] \geq \min[\mu_A(x_1), \mu_A(x_2)].$$

Definition 2.6. The *height* of a fuzzy set A on X is defined as

$$h(A) = \sup_{x \in X} \mu_A(x).$$

If $h(A) = 1$, A is called *normal*, otherwise *subnormal.*

Definition 2.7. The *nucleus* of a fuzzy set A is the set of values x for which $\mu_A(x) = 1$.

Example 2.3

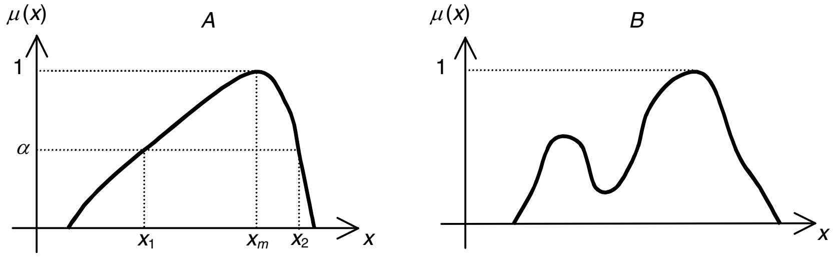


Figure 2.1. Convex-nonconvex fuzzy sets.

In Figure 2.1, A is convex but B is nonconvex. The α-level set of A is the set of $x \in [x_1, x_2]$, the height is $h(A) = 1$, and the nucleus is $\{x_m\}$.

Definition 2.8. A fuzzy number A is a fuzzy set in the reals R for which the following are true:

a. A is normal ($\exists x$: $\mu_A(x) = 1$)
b. A is convex
c. A is upper semicontinuous
d. A has bounded support

Example 2.4

The following are examples of fuzzy numbers.

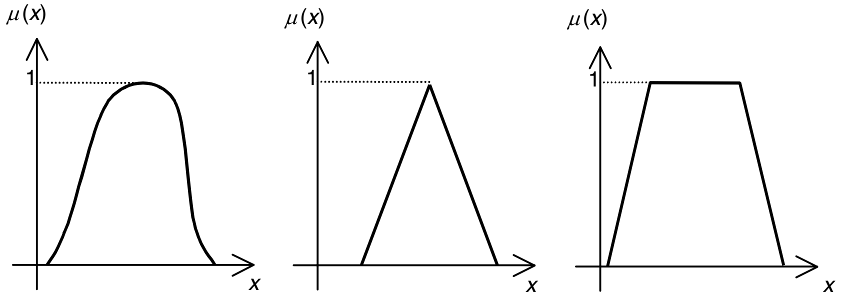


Figure 2.2. Fuzzy numbers.

2.2 Operations of Fuzzy Sets

The basic notions concerning operations on crisp sets will now be extended to fuzzy sets.

Definition 2.9. Two fuzzy sets A and B in X are equal if $\mu_A(x) = \mu_B(x)$, $\forall x \in X$. We write $A = B$.

Definition 2.10. A fuzzy set A in X is a subset of another fuzzy set B also in X if $\mu_A(x) \leq \mu_B(x)$, $\forall x \in X$.

The following definitions are concerned with the complement, the union, and the intersection of fuzzy sets as defined by Zadeh. It should be stressed that these definitions, intuitively appealing as they may be, are by no means unique because of the nature of fuzzy sets. Others have proposed different definitions.

Definition 2.11. The following membership functions are defined:

a. Complement $\bar{A}$ of a fuzzy set A in X

$$\mu_{\bar{A}} = 1 - \mu_A(x)\ , x \in X.$$

b. Union $A \cup B$ of two fuzzy sets in X

$$\mu_{A \cup B} = \max[\mu_A(x),\ \mu_B(x)],\ x \in X.$$

c. Intersection $A \cap B$ of two fuzzy sets in X

$$\mu_{A \cap B} = \min[\mu_A(x),\ \mu_B(x)],\ x \in X.$$

Example 2.5

In the context of Example 2.1 let us define a new fuzzy set B = "Dangerous temperature" as $B = \{0/37.5,\ 0.1/38,\ 0.2/38.5,\ 0.5/39,\ 0.8/39.5,\ 1/40\}$. According to Definition 2.11, we have

$A \cup B$ = "High or dangerous temperature"

$= 0/36.5 + 0/37 + 0.1/37.5 + 0.5/38 + 0.8/38.5 + 1/39 + 1/39.5 + 1/40.$

$A \cap B$ = "High and dangerous temperature"

$= 0/36.5 + 0/37 + 0/37.5 + 0.1/38 + 0.2/38.5 + 0.5/39 +.0.8/39.5$

$+ 1/40.$

$\bar{A}$ = "Not high temperature"

$= 1/36.5 + 1/37 + 0.9/37.5 + 0.5/38 + 0.2/38.5 + 0/39 + 0/39.5.$

The definitions of an intersection and union can be developed from a more general point of view. An intersection may be defined via a *t-norm.*

Definition 2.12. A t-norm is a bivariate function t: $[0, 1]\times[0, 1]\rightarrow[0, 1]$ satisfying:

a. $t(0, 0) = 0$
b. $t(x, 1) = x$
c. $t(x, y) \le t(w, z)$ if $x \le w$ and $y \le z$ (monotonicity)
d. $t(x, y) = t(y, x)$ (symmetry)
e. $t[x, t(y, z)] = t[t(x, y), z]$ (associativity)

This definition provides the tools of combining two membership functions to find the membership function of $A \cap B$. For the union $A \cup B$, we have correspondingly the definition of the *t-conorm* or *s-norm.*

Definition 2.13. A t-conorm is a bivariate function c: $[0, 1]\times[0, 1]\rightarrow[0, 1]$ satisfying:

a. $c(1, 1) = 1$
b. $c(x, 0) = x$
c. $c(x, y) \le c(w, z)$ if $x \le w$ and $y \le z$ (monotonicity)
d. $c(x, y) = c(y, x)$ (symmetry)
e. $c[x, c(y, z)] = c[c(x, y), z]$ (associativity)

From these definitions, for two fuzzy sets $\mu_A(x)$ and $\mu_B(x)$, we obtain $\mu_{A\cap B}(x) = t[\mu_A(x), \mu_B(x)]$ and $\mu_{A\cup B}(x) = c[\mu_A(x), \mu_B(x)]$.

Example 2.6

The following are examples of t-norms and t-conorms.

Name	$t(x, y)$ (intersection)	$c(x, y)$ (union)
Algebraic product-sum	$x\,y$	$x + y - x\,y$
Hamacher product-sum	$\dfrac{xy}{x + y - xy}$	$\dfrac{x + y - 2xy}{1 - xy}$
Einstein product-sum	$\dfrac{xy}{1 + (1 - x)(1 - y)}$	$\dfrac{x + y}{1 + xy}$
Bounded difference product-sum	$\max(0, x + y - 1)$	$\min(1, x + y)$
Dubois-Prade $0 \le p \le 1$	$\dfrac{xy}{\max(x, y, p)}$	$1 - \dfrac{(1 - x)(1 - y)}{\max[(1 - x), (1 - y), p]}$
Minimum-maximum	$\min(x, y)$	$\max(x, y)$

It is worth noting that, contrary to what holds in set theory, when A is a fuzzy set in X, then $A \cup \overline{A} \neq X$ and $A \cap \overline{A} \neq \emptyset$ because it is not certain where A ends and $\overline{A}$ begins. This is the fundamental reason that places probability and fuzzy sets apart, although both handle uncertainty. Probability is suitable for a different kind of uncertainty than fuzzy sets, and in our opinion, the debate about which discipline is "better" or "correct" is rather beside the point. Each of them performs its own scientific function successfully within its capabilities and limitations. Below we outline some of the differences between probability and fuzzy sets.

1. In probability, an event is a crisp subset of a σ-algebra and the uncertainty revolves about the odds of its occurrence. For example, the probability of being 1.75 m tall, or $P(\text{height} = 1.75)$, concerns the frequency of the relevant event. In fuzzy set theory, events do not form σ-algebras. A pertinent question in this context would be "to what degree is 1.75 m tall?"

2. Given a probability space $(\Omega, \mathscr{F}, P)$, where Ω is the universe, $\mathscr{F}$ a σ-algebra of events, P a probability measure, and mutually exclusive events A_i, then by an axiom

$$P(\cup_i A_i) = \textstyle\sum_i P(A_i).$$

This does not happen in fuzzy set theory. A fuzzy measure in [0, 1] could be defined for a finite X, called a possibility measure Π, as follows:

a. $\Pi(\emptyset) = 0$
b. $\Pi(X) = 1$
c. $A \subset B \Rightarrow \Pi(A) \le \Pi(B)$
d. $\Pi(\cup_i A_i) = \sup_i \Pi(A_i)$

Obviously Π and P obey different rules.

3. Finally, although a membership function μ ranges in [0, 1], it does not share all the features of a probability distribution function $F(x)$, which are

$$F(-\infty) = 0, F(+\infty) = 1,$$

$$F(x) = F(x^+),$$

$$F(x_1) \le F(x_2) \text{ if } x_1 < x_2,$$

or of a density function, which are

$$f(x) \ge 0,$$

$$\int_{-\infty}^{\infty} f(x)dx = 1.$$

2.3 The Extension Principle

Functions in mathematics map points $x_1, x_2, \ldots$ in a set X to points in another set Y. Such mappings may occur between fuzzy sets X and Y using the *extension principle*. Let a function f that maps subsets of X into subsets of Y. If

$$A = \mu_1/x_1 + \mu_2/x_2 + \ldots + \mu_n/x_n,$$

then by the extension principle

$$B \triangleq f(A) = \mu_1/f(x_1) + \mu_2/f(x_2) + \ldots + \mu_n/f(x_n)$$

$$= \mu_1/y_1 + \mu_2/y_2 + \ldots + \mu_n/y_n$$

for $x_i \in X$ and $y_i = f(x_i) \in Y$. If the same y corresponds to more than one x_i's, then we use the maximum of the membership grades of the x_i's such that $y = f(x_1) = f(x_2) = \ldots = f(x_n)$; i.e.,

$$\mu_B(y) = \max[\mu_A(x_1), \mu_A(x_2), \ldots, \mu_A(x_n)].$$

If the function f is defined on vector spaces of proper dimensions, i.e., $B = f(A_1, A_2, \ldots, A_k)$, then

$$\mu_B(y) = \min[\mu_{A_1}(x_1), \mu_{A_2}(x_2), \ldots, \mu_{A_k}(x_k)],$$

which is the minimum of the membership grades of the values x_i that produce y. Furthermore, if

$$y = f(x_1, x_2, \ldots, x_k) = f(x_1', x_2', \ldots, x_k') = \ldots,$$

then

$$\mu_B(y) = \max_{\substack{(x_1, \ldots, x_k) \\ (x_1', \ldots, x_k') \\ \vdots}} \{\min[\mu_{A_1}(x_1), \ldots, \mu_{A_k}(x_k)], \min[\mu_{A_1}(x_1'), \ldots, \mu_{A_k}(x_k')], \ldots\}.$$

Example 2.7

Let $A = 0.5/x_1 + 0.2/x_2 + 0.7/x_3$ and $y = f(x_1) = f(x_2) = f(x_3)$. Then

$$B = \max(0.5, 0.2, 0.7)/y = 0.7/y.$$

Example 2.8

Let A be a fuzzy set in [2, 4] with the triangular membership function of Figure 2.3 and $y = \ln x$.

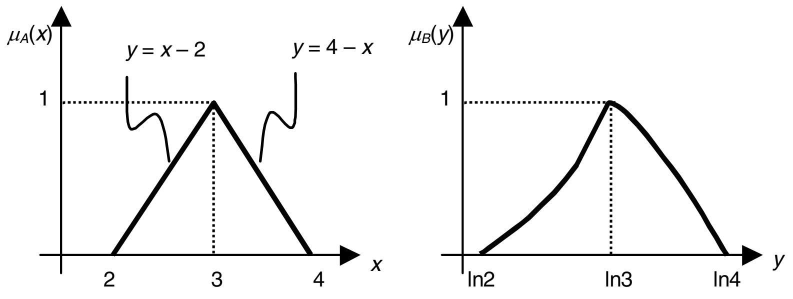


Figure 2.3. Membership functions of Example 2.8.

Then we obtain $x = e^y$ and

$$\mu_B(y) = \begin{cases} e^y - 2 & \text{if } \ln 2 \le y \le \ln 3, \\ 4 - e^y & \text{if } \ln 3 \le y \le \ln 4. \end{cases}$$

Example 2.9

Now we have two fuzzy sets

$$A_1 = 0.1/x_1 + 0.4/x_2 + 0.8/x_3$$

$$A_2 = 0.6/x_1' + 1/x_2'$$

and a function f that maps x_i and x_i' into y_i as follows:

$$y_1 = f(x_1, x_1') = f(x_1, x_2') = f(x_3, x_2'),$$

$$y_2 = f(x_2, x_1'),$$

$$y_3 = f(\mathrm{x}_2, x_2') = f(x_3, x_1'),$$

or in matrix form

$$\begin{array}{cc} & \begin{matrix} x_1' & x_2' \end{matrix} \\ \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} & \begin{bmatrix} y_1 & y_1 \\ y_2 & y_3 \\ y_3 & y_1 \end{bmatrix} \end{array}.$$

Then

$$\mu_B(y_1) = \max[\min(0.1, 0.6), \min(0.1, 1), \min(0.8, 1)] = 0.8,$$

$$\mu_B(y_2) = \max[\min(0.4, 0.6)] = 0.4,$$

$$\mu_B(y_3) = \max[\min(0.4,1), \min(0.8, 0.6)] = 0.6.$$

Therefore, $B = 0.8/y_1 + 0.4/y_2 + 0.6/y_3$.

2.4 Linguistic Variables

Loosely speaking, a *linguistic variable* is a variable "whose values are words or sentences in a natural or artificial language," as Zadeh has put it. Take, for example the concept "Height," which can be seen as a linguistic variable with values "very tall," "tall," "not tall," "average," "short," "very short," and so on. To each of these values, we may assign a membership function. Let the height range over a region [0, 230 cm] and assume that the linguistic terms are governed by a given set of rules. Then we define formally a linguistic variable.

Definition 2.14. A linguistic variable is a 4-tuple (T, X, G, M), where

T is a set of natural language terms called *linguistic values*
X is a universe of discourse
G is a context free grammar used to generate elements of T
M is a mapping from T to the fuzzy subsets of X

Example 2.10

In the example above,

$$T = \{\text{very tall, tall, } \ldots\},\ X = [0, 230]$$

and M for tall:

$$\mu_A(x) = \begin{cases} 0 & \text{if } x \le 170, \\ \dfrac{x-170}{15} & \text{if } 170 < x \le 185, \\ 1 & \text{if } 185 < x. \end{cases}$$

Linguistic variables are fundamental when we want to represent knowledge in approximate reasoning. Often the meaning of a term needs to be modified.

Examples of modifiers are the following:

$$\mu_{\text{VERY}} = [\mu_A(x)]^2,$$

$$\mu_{\text{MORE OR LESS}} = \sqrt{\mu_A(x)}\,,$$

$$\mu_{\text{INDEED}} = \begin{cases} 2[\mu_A(x)]^2 & \text{if } 0 \le \mu_A(x) < 0.5, \\ 1 - 2[1 - \mu_A(x)]^2 & \text{if } 0.5 < \mu_A(x) \le 1. \end{cases}$$

2.5 Fuzzy Reasoning

A queue is observed, and the conclusion "the queue is positive small" is derived. This conclusion may be formally written as "s is PS" by choosing a symbol s for queue size and a symbol PS for "positive small." Experience has shown that in fuzzy control, a large number of linguistic variables can be represented by seven linguistic values: NB (negative big), NM (negative medium), NS (negative small), ZO (zero), PS (positive small), PM (positive medium), and PB (positive big). A common domain for these values is the *standard* domain [–6, 6] or the *normalized* one [–1, 1]. A large number of control problems can be solved efficiently over these domains.

The proposition "s is PS" is called *atomic* and assumes a certain membership grade, say $\mu_{\mathrm{PS}} = 0.4$. Atomic propositions together with *connectives* such as AND, OR, NOT, or IF-THEN form *compound* propositions. For example, the expressions

$$\text{IF } X \text{ is } A, \text{ THEN } X \text{ is } B,$$

$$X \text{ is } A \text{ OR } B,$$

and so on are compound propositions.

The connective AND corresponds to logical *conjunction* "X is $A \cap B$" where A and B are fuzzy sets and the appropriate membership function is $\mu_{A \cap B}$. Similarly OR corresponds to *disjunction* "X is $A \cup B$" and $\mu_{A \cup B}$ and NOT corresponds to "X is $\bar{A}$" and $\mu_{\bar{A}}$.

Now consider two queues in parallel with queue lengths s_1 and s_2 and one server with variable service rates. An experienced operator decides in terms of natural language "if the queue size s_1 is large and the queue size s_2 is also large, then the server should run at a high rate." This statement can be written

$$\text{IF } s_1 \text{ is PB AND } s_2 \text{ is PB, THEN } r \text{ is PB.}$$

This proposition has the form

$$\text{IF (antecedents) THEN (consequents)}$$

and is called a *fuzzy conditional* or *fuzzy if-then* production rule.

2.6 Rules of Inference

Classical logic is based on tautologies of the following type (we use "$\wedge$" for "AND," "$\vee$" for "OR," and "$\rightarrow$" for "implies").

1. *Modus ponens*

Premise: A is true
$\underline{\text{Implication: if } A \text{ then } B}$
Conclusion: B is true

Symbolically: $[A \wedge (A \rightarrow B)] \rightarrow B$

2. *Modus tollens*

Premise: not B is true
Implication: if A then B
Conclusion: not A is true

Symbolically: $[\bar{B} \wedge (A \rightarrow B)] \rightarrow \bar{A}$

3. *Syllogism*

Premise: "if A then B" is true
Implication: if B then C
Conclusion: "if A then C" is true

Symbolically: $[(A \rightarrow B) \wedge (B \rightarrow C)] \rightarrow (A \rightarrow C)$

Such rules can be generalized in the context of fuzzy logic. Two common rules of approximate reasoning are the *Generalized Modus Ponens* (GMP) and the *Compositional Rule of Inference* (CRI). Let A, A', B, and B' be fuzzy sets and X, Y be linguistic variables. Then we define

GMP Premise: X is A'
Implication: if X is A, then Y is B
Conclusion: y is B'

Example 2.11

GMP Premise: a student has a very high IQ
Implication: if a student has a high IQ, then he is academically good
Conclusion: The student is academically very good

The compositional rule of inference is a special case of the generalized modus ponens and has the form

CRI Premise: X is A'
Implication: $X \; R \; Y$ (X is related to Y)
Conclusion: y is B'

Here R substitutes for "if-then."

Example 2.12

CRI Premise: Jim is tall
Implication: Jim is somewhat taller than George
Conclusion: George is rather tall

2.7 Mamdani Implication

The meaning of "if-then" rules is represented by relevant membership functions. As expected, there is a long list of ways to represent the meaning of "if-then" rules. They are all subjective, but their efficacy depends on the application.

In fuzzy control, the most commonly used and the most efficient implication is called *Mamdani implication*. It is defined by

$$\mu_C(x, y) = \min[\mu_A(x), \mu_B(y)]$$

for the rule *if X is A, then Y is B*. In the sequel, we shall see numerous applications of the Mamdani implication in practical control problems. Here we give a simple example.

Example 2.13

Let

$$A = 0.2/x_1 + 0.3/x_2 + 0.4/x_3,$$

$$B = 0.1/y_1 + 0.2/y_2 + 0.6/y_3 + 0.7/y_4.$$

The following table summarizes the Mamdani implication for the rule *if X is A, then Y is B*:

	y_1	y_2	y_3	y_4
x_1	0.1	0.2	0.2	0.2
x_2	0.1	0.2	0.3	0.3
x_3	0.1	0.2	0.4	0.4

3 Knowledge and Fuzzy Control

3.1 Introduction

The previous chapter provided all the basics we need to proceed with fuzzy control of queuing systems. The main ingredient of a fuzzy control system is knowledge.

Knowledge-based systems in the context of control are mechanisms for incorporating knowledge in control systems. This knowledge cannot be included in the conventional mathematical model but is important in achieving good performance and robustness and is ordinarily handled by such means as manual operation.

A knowledge-based system may assist a closed-loop controller by directly and fully substituting for the control loop, which conventionally consists of a mathematical algorithm, or by just supervising the control procedure. Simply put, knowledge-based systems take the knowledge and experience of a human operator or designer, which cannot take the form of elegant mathematics, and transfer it to practical control situations.

3.2 Knowledge-Based Systems as Controllers

What follows is a rough classification of knowledge-based systems as controllers. Such a list is indicative of the potential of knowledge-based systems.

Process Monitoring

An operator receives various signals about a process: deviations of quality specifications, sudden disruptions or changes, and so on. The operator then uses past and present data to identify causes and select actions. The same operator aided by a knowledge-based system may act faster and more systematically.

A knowledge-based system assists the operator in monitoring the various stages of the process providing early warnings about impending changes and zeroing in on causes of alarms. The system stores histories of operation and provides guidance in real time.

The knowledge-based system should be capable of combining information in numerical and symbolic forms, assisting decision making in real time, and validating its procedures.

Process Planning

Process planning entails a complicated task of meeting demand, be it stochastic or deterministic, static or dynamic, for a process while satisfying an array of technological, environmental, economic, and other constraints. This sort of planning is done conventionally through a host of operations research tools such as linear and nonlinear programming, discrete event control, as well as special techniques of scheduling theory.

Knowledge-based systems may successfully complement these methods where mathematical models do not represent reality well or simply do not exist at all. Such systems incorporate knowledge about the controlled process at as many levels as possible: physical, experiential, and mathematical.

Process Fault Diagnosis

This category of knowledge-based systems deals with the detection of faults relying on detailed knowledge of the process. The system performs routine measurements and focuses on errors between the expected and current outputs. Expected outputs result from knowledge that could be model based.

Supervisory Process Control

Supervisory control is usually applied in conjunction with a conventional controller with the aim of tuning the process so as to achieve certain goals. In effect, supervisory control complements conventional control, enhancing its effectiveness.

A conventional controller needs a mathematical model. It is often the case that such a model does not exist or, if it exists, it is ineffective. A knowledge-based system may then substitute completely for the conventional controller. It is called a knowledge-based controller. If the knowledge and the inference are fuzzy, then we have a *Fuzzy Knowledge-Based Controller* (FKBC).

In this book, we examine the control of queuing systems, which belong to the category of discrete event systems, using FKBCs. As we proceed, detailed descriptions of all details of FKBCs will be given. In the present chapter, we shall examine some general features of fuzzy controllers.

3.3 Fuzzification

The fuzzification interface of a FKBC functions as follows:

1. Identifies and measures the input variables.
2. Performs a scale transformation of the physical domain into a normalized or *standard* universe of discourse. This transformation is not always necessary. There are, however, cases where the physical domain is inconvenient and a transformation facilitates the fuzzy operations significantly. The most common standard domains in this book are [–6, 6], [0, 6], [–4, 4], [0, 4], [–1, 1], and [0, 1].

3. Fuzzifies the crisp input data, whether normalized or not. Fuzzification is a way of dealing with data that, subjective or objective, might be fraught with vagueness and imprecision. The fuzzifier transforms crisp data into suitable linguistic values, corresponding to fuzzy sets, so that these data become compatible with the fuzzy antecedent-consequent mechanism. Thus, for a crisp value x_0, we obtain a fuzzy set X via

$$X = \text{fuzzifier}(x_0). \tag{3.1}$$

Fuzzification is closely related to knowledge because the membership functions used in Equation (3.1) are the result of deep system knowledge, mathematical or experiential.

3.4 Knowledge Base

The knowledge base contains the knowledge related to a particular control problem. It consists of a *data base* and a *rule base*.

Data Base

The data base provides information needed to devise linguistic control rules and the fuzzification/defuzzification procedures. Thus, the fundamental function of the data base is twofold:

1. Selection of membership functions to define the meaning of pertinent input/output variables. This selection may be straightforward or rather involved in queuing control. We shall develop in detail all the ideas concerning membership functions as we develop each queuing system. Of course, engineering judgment and expert knowledge play an important role.
2. Definition of the physical and normalized domains, which boils down to selecting proper normalization/denormalization coefficients.

The number of inputs and corresponding fuzzy sets define the size of the rule base and thus the dimension of the system. As we shall see, this dimension grows geometrically with the number of fuzzy sets. Therefore, the choice of membership functions should be as economical as possible. Such a choice, on the other hand, may provide for an inaccurate system model. The final number of fuzzy sets is a tradeoff between computational speed and accuracy, which is the result of trial-and-error as well as experience.

Rule Base

The rule base summarizes the control actions of an expert in the form

IF (input variables) THEN (control policy).

In other words, a linguistic description based on expert knowledge provides the "best" policy. We then have an antecedent or IF side that incorporates the linguistic values of the inputs, expressed as linguistic variables, and a consequent or THEN side that includes the control outputs also in a linguistic form.

Example 3.1

For a 3-input, 2-output system, the fuzzy control rules have the form:

Rule 1: IF x_1 is $A_1^{(1)}$ AND x_2 is $A_2^{(1)}$ AND x_3 is $A_3^{(1)}$, THEN u_1 is $B_1^{(1)}$ AND u_2 is $B_2^{(1)}$

Rule 2: IF x_1 is $A_1^{(2)}$ AND x_2 is $A_2^{(2)}$ AND x_3 is $A_3^{(2)}$, THEN u_1 is $B_1^{(2)}$ AND u_2 is $B_2^{(2)}$

$\vdots$

Rule n: IF x_1 is $A_1^{(n)}$ AND x_2 is $A_2^{(n)}$ AND x_3 is $A_3^{(n)}$, THEN u_1 is $B_1^{(n)}$ AND u_2 is $B_2^{(n)}$

The following are needed to construct a rule base:

1) Input/output variables

A proper choice of input/output variables is crucial in the description and performance of the system. This choice determines the structure of the controller and relies on experience and engineering knowledge. Typical inputs in queuing control are mean incoming or outgoing rates and sizes of queues. Typical control variables are service rates, service discipline, decisions about turning on or off a server, and decisions about which customer goes to which server and when.

2) Range of linguistic values

The choice of the linguistic values is closely tied to the choice of membership functions as we have already seen. Their range is a matter of achieving the best performance and needs tuning via some computational procedure.

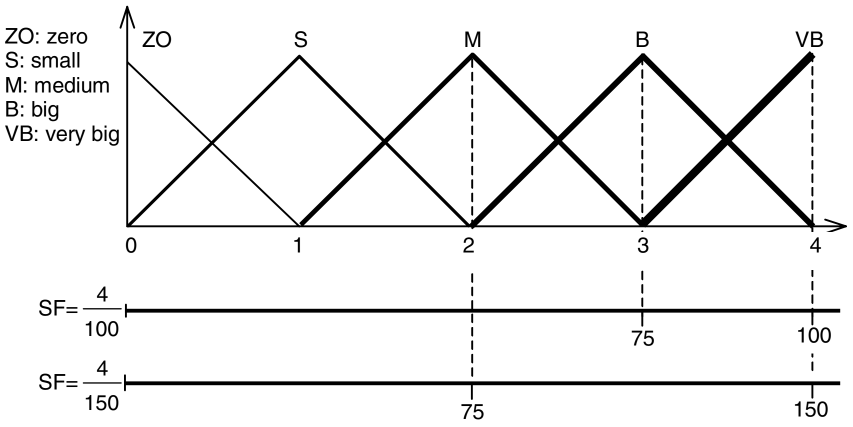


Figure 3.1. Tuning scaling factors.

Tuning may affect the scaling coefficients of the normalized domain or the shape of membership functions. To illustrate this process, suppose we have a queue with capacity 100 and normalized range [0, 4]. The physical domain [0, 100] needs to be scaled down to [0, 4] using a factor of 0.04. Consider the membership functions of Figure 3.1. A queue size of 75 corresponds to *B* with membership grade 1.

If the scaling factor now becomes 4/150, then 75 corresponds to *M* with grade 1, thus reducing the sensitivity of the controller with respect to the input.

This sensitivity reduction is across the board and uniform. A selective alteration of the shape of the membership functions achieves a sensitivity change over specific ranges. A controller using the membership functions of Figure 3.2 is more sensitive to large values for queues than to small ones. If such adaptations are done automatically by the controller, we speak of a self-tuning controller.

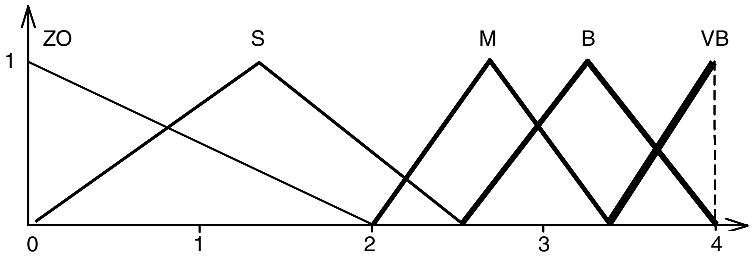


Figure 3.2. Change of membership functions.

3) Derivation of fuzzy rules

A large amount of information and concomitant reasoning in everyday life is linguistic. In a sense, we operate as fuzzy controllers in an impressive number of ways: when we drive, open a faucet, tune in on a radio station, play soccer or tennis, give a shot, squeeze an orange, and so on. Fuzzy control rules, consciously or subconsciously, are ubiquitous and may enable experts express their knowledge in convenient ways. Loosely speaking, we have the following ways of building a rule base:

- *Experience and engineering knowledge*: We have already spoken about daily tasks requiring experience. Such experience and engineering knowledge expressed linguistically are the core of a rule base. Devising a rule may be aided by properly constructed questionnaires directed to specialists or operators.
- *Fuzzy model*: The linguistic description of a system comprises a fuzzy model of the system. The rule base is then constructed from this model.
- *Mathematical model*: If a mathematical model of the system exists, it may be used to develop a fuzzy rule base and control algorithm.

Fuzzy control rules belong in two categories depending on objectives.

- *State evaluation rules*: Such rules evaluate the state of the system such as queue length, expected rate of arrivals, and expected service rate at time t and then compute a control policy such as to turn on or off the server so as to achieve minimum cost.
- *Object evaluation rules*: Rules of this type are associated with the so-called *object evaluation or predictive fuzzy control.* The control now is the result of objectives in the following linguistic form.

IF performance index x_1 is A_1 AND … AND performance index x_n is A_n when control u is U, THEN this rule is selected and U is the control chosen.

Compactly

IF [u is $U \rightarrow (X_1$ is A_1 AND … AND X_n is $A_n)$] THEN u is U.

Example 3.2

This example of object evaluation rules is taken from an automatic fuzzy control algorithm for train operation. "If the train stops in the predetermined allowance zone when the control notch is not changed, then the control notch is not changed."

A number of issues arise in the context of building a rule base. These are as follows:

1) Consistency
A rule base should be designed in such a way that no contradictions ensue.

Definition 3.1. A rule base is *consistent* if it contains no rules with the same antecedents and different consequents.

Take, for example, a rule base where the following rules are encountered:

IF s_1 is PS AND s_2 is ZO, THEN u is ZO,
IF s_1 is PS AND s_2 is ZO, THEN u is PS.

This rule base is inconsistent. It should be stressed that all the rule bases in this book are formulated so that no two rules have the same antecedents and thus they are consistent.

2) Completeness
A rule base is a matrix of linguistic values for a control output given linguistic combinations of the inputs. There might be combinations of inputs that produce a null output. Then we say that the rule base is not complete.

Definition 3.2. A rule base is *complete* if all combinations of inputs produce non-null outputs.

Incompleteness of rule bases in FKBCs is common because not all input combinations are of interest.

3.5 Inference Engine

The inference engine is also known as rule firing and is the mechanism whereby the input rules are combined to produce a control output. There are two basic types of inference engines, which for Mamdani implication are equivalent.

- *Composition inference*: This engine aggregates all rules in one fuzzy relation. Then the fuzzified inputs and the aggregated fuzzy relation are combined with the aid of the composition operation to obtain the fuzzy control output. This type of engine is not so common in fuzzy control.

- *Individual rule firing*: Here each rule is fired individually and the control output is computed. As was already mentioned, Mamdani implication is commonly used to obtain the fuzzy output. In the following chapters, the use of this implication will be illustrated in detail. Individual rule firing is the most frequently used implication in fuzzy control. We use it exclusively in queuing control.

Example 3.3

Consider two queues in series with expected arrival rates λ_1 and λ_2 and queue lengths s_1 and s_2, respectively. The second queue also receives all serviced customers of queue 1. An incoming customer may be accepted or rejected at each entry point, depending on cost considerations. The following comprise individual rule firing via Mamdani implication (min-inference):

If s_1 is PM with grade 0.09 and s_2 is ZO with grade 0.72 and λ_1 is ZO with grade 0.74 and λ_2 is PS with grade 0.67, then d_1 is NO (reject) with grade 0.09 = min(0.09, 0.72, 0.74, 0.67).

If s_1 is PS with grade 0.74 and s_2 is PS with grade 0.62 and λ_1 is PS with grade 0.6 and λ_2 is PM with grade 0.67*,* then d_1 is NO with grade 0.6 = min(0.74, 0.62, 0.6, 0.67).

The final crisp decision is obtained via defuzzification.

3.6 Defuzzification

Defuzzification is a mechanism for converting fuzzy control actions into nonfuzzy or crisp ones. This is done because in practice crisp values of controls can be used. The defuzzifier, in addition to converting fuzzy values into crisp ones, *denormalizes* the control values if normalized domains are used.

There are several defuzzification methods. Below we show the most common ones when rules fire individually.

Center of Gravity

Let u_i, $i = 1, \ldots, k$ be the values of the control policies and the corresponding membership grades $\mu_A(u_i)$. Then the center of gravity defuzzification formula provides a crisp value u_c for the control as follows:

$$u_c = \frac{\sum_{i=1}^{k} u_i \mu_A(u_i)}{\sum_{i=1}^{k} \mu_A(u_i)},$$

which for continuous values becomes

$$u_c = \frac{\int u\mu_A(u)du}{\int \mu_A(u)du}.$$

Height Method

First we give one definition. Recall that the nucleus of a normal fuzzy set A is the set of values x for which $\mu_A(x) = 1$.

Definition 3.3. The *peak value* of a normal fuzzy set A is its nucleus if there is only one point for which $\mu_A(x) = 1$.

The method of height defuzzification takes account of k outputs u_i. Each u_i has a membership grade or height f_i, whereas the corresponding peak values are e_i. Then

$$u_c = \frac{\sum_{i=1}^{k} e_i f_i}{\sum_{i=1}^{k} f_i}. \tag{3.2}$$

This method will be used in queuing control.

Example 3.4

We have two rules that result in "u is PS with grade 0.1" and "u is ZO with grade 0.2." Let the membership functions be as in Figure 3.3. Then the heights are $e_1 = 0$, $e_2 = 1$, and Equation (3.2) gives

$$u_c = \frac{0 \times 0.1 + 1 \times 0.2}{0.1 + 0.2} = \frac{2}{3}.$$

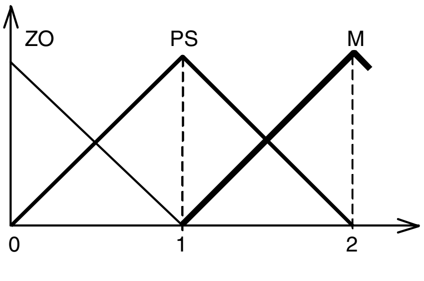


Figure 3.3. Membership functions of ZO and PS in Example 3.4.

Other methods include *first of maxima*, *center of largest area*, and *middle of maxima*.

It turns out that the height method is the most convenient and advantageous one. It exhibits continuity (small input changes produce small output changes), lack of ambiguity (it chooses unambiguously between different fuzzy subsets), small computational demand, and weight counting (takes account of each rule via f_i).

3.7 Design Parameters of a Fuzzy Logic Controller

The principal design parameters for a fuzzy logic controller are (Driankov *et al.* 1996):

1. Fuzzification methods and meaning
2. Knowledge base
   a. discretization/normalization of universes of discourse
   b. choice of inputs and outputs
   c. choice of membership functions
   d. derivation of fuzzy control rules
   e. consistency, completeness of fuzzy control rules
3. Inference engine
   a. definition of fuzzy implication
   b. inference mechanism
4. Defuzzification method

3.8 Fuzzy Queue Control

Queuing systems are controlled using fuzzy control by emulating a skilled human operator at each decision epoch. The current state is observed, and then an inference engine equipped with a fuzzy rule base fires an on-line decision to adjust the system behavior in order to guarantee that the system is optimal in some sense.

The architecture of the fuzzy logic controllers depends on the features of each queuing system. However, the basic principles of each fuzzy logic controller are similar. First of all, the inner workings of a control action and the state of the system are explored and then proper policies are devised by mimicking the human way of thinking.

The universes of discourse for all fuzzy sets are continuous, and the membership functions are chosen to be *triangular*. We make this choice because the parametric, functional descriptions of triangular membership functions are the most economic ones and because such membership functions can approximate any other membership function. To present the fuzzy rules, we usually use NB, NM, NS, ZO, PS, PM, and PB to indicate “negative big,” “negative medium,” “negative small,” “zero,” “positive small,” “positive medium,” and “positive big,” respectively, unless otherwise explained.

Queuing systems are simulated and controlled using the C programming language. Mamdani implication is used to present the meaning of “if-then” rules. This implication is most popular in the fuzzy control field because it produces good results in most practical applications.

To change the fuzzy output into a usable crisp one, the *height method* of defuzzification is used.

Finally, it should be stressed that the policies of this book will be called optimal although optimality cannot be proven mathematically, but this is the case with many fuzzy logic applications, as we have already pointed out. The existence of such policies will always be assumed in the sense of deterministic stationary ones.

4 Control of the Service Activities

4.1 Introduction

In this chapter, we consider queuing systems in which the service rate is the controlled variable. The cost depends on the queue length and selected rate. The objective is to choose the service rate dynamically, based on the state of the system so as to minimize the average cost over an infinite horizon. Six problems, either known in the literature or new, are studied in detail:

- M/M/1 and M/M/*m* queuing systems with server vacations
- Single-server queuing systems with and without switching costs
- Tandem queuing systems with and without service costs

A fuzzy control approach is presented to solve these problems. Simulation shows that the approach is efficient and promising, in cases where analytical solutions do not exist.

It should be stressed at the outset that the reservations about optimality expressed in Section 1.5 are valid throughout the rest of the book. It should also, however, be stressed that the fuzzy algorithms yielded identical results to those of analytical models whenever available. In this chapter, the problem of Section 4.2 has an analytical solution, and the problems of Sections 4.3 and 4.4 have analytical solutions only for special cases. The remaining problems have no analytical counterparts.

4.2 Single Server with Vacations

4.2.1 Problem Description

A queuing system in which a server may be turned off is said to be a system with vacations or with a removable server, as illustrated in Figure 4.1.

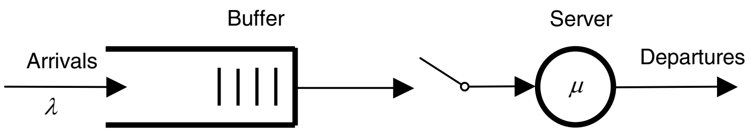


Figure 4.1. Queuing system with vacations.

Customers arrive into the system, according to a Poisson process with parameter λ, and the queue has infinite capacity. There is one exponential server in the system with service rate μ, where $\mu > \lambda$. This is an M/M/1 queuing system where the service rate may be adjusted to zero during certain intervals of time depending on the state of the system.

The operation of the system is associated with three types of cost:

1. service cost r, which is the cost per time unit when the server is on
2. switch-on cost R, which is the fixed nonnegative cost incurred whenever the server is turned on
3. holding cost h, which is the holding cost per time unit per customer in the system, including the one in service (if any)

Here we assume that there is no running cost when the server is off or switched off. However, the model can easily be extended to admit such cost parameters (see Section 4.2.4).

The system objective is to find an optimal control policy, which dictates when the server is turned off, that minimizes the average cost rate of the system over an infinite time horizon. This queuing process is a semi-Markov decision process. Indeed, because of the presence of the controller, the times between successive service completions are no longer exponentially distributed. However, at the instants of state transitions, this stochastic process behaves like Markov. It is then called semi-Markov.

Studying a more general process (see Section 4.2.4), Heyman (1968) has proved that it is optimal to keep the server always on when

$$\left[\frac{2r(1-\rho)}{h}+1\right]^2 - \frac{8\lambda R(1-\rho)}{h} < 0, \tag{4.1}$$

where $\rho = \lambda/\mu$ is the traffic intensity, otherwise to turn the server on when there are N customers waiting before a dormant server and to turn it off when the system enters an idle period; N is one of the integers about

$$n^* = \sqrt{\frac{2\lambda R(1-\rho)}{h}}\,, \tag{4.2}$$

whichever gives the smallest cost. This policy is called an *exhaustive hysteretic policy* or a (0, N) *N-policy* with $N = 0$ when inequality (4.1) is true. Thus, under an optimal policy, it seems beneficial to keep the server always on when the switch-off cost R is sufficiently larger than the service cost rate r.

In the next sections, we develop a controller for this system using fuzzy logic. To emphasize the dynamic aspect of the problem, we assume that inequality (4.1) is not true. Furthermore, we adopt the first part of the optimal policy, which dictates that the server is kept on whenever customers are present in the system and it is switched off when the system empties. Thus, the fuzzy controller determines when the server must be turned on. Equation (4.2) will be used to compare the optimal and proposed policies.

4.2.2 Architecture of the Fuzzy Knowledge-Based Controller

State Evaluation

At any time t, the system may be empty or nonempty and the server may be on (busy or idle) or off. In the latter case, we say that the server is *dormant*. The server is *idle* if it is kept on although there are no customers in the system.

Conventional control techniques usually describe the state of the system by (k, s), where $k = k(t)$ is the state of the server at time t, taking the value 1 if the server is on or 0 if the server is off, and $s = s(t)$ is the number of customers in the system at time t. The state variable s changes whenever a new customer arrives or the server completes service. The state variable k changes whenever we decide to switch the server on or off. By the memoryless (Markovian) property of the interarrival and service times, we restrict the decision epochs when the server can be turned on or off to the times at which the state changes. Therefore, time can be regarded as discrete rather than continuous with no loss of generality. In this setting, the average cost rate of the system over an infinite time horizon can be expressed as follows:

$$g = \lim_{T\to\infty}\left[\sum_{t=1}^{T}\frac{hs(t)}{T}+\sum_{t=1}^{T}\frac{rk(t)}{T}+\sum_{t=1}^{T}\frac{R\times 1_{\{k(t-1)=0,k(t)=1\}}}{T}\right], \tag{4.3}$$

where $1_{\{x\}}$ is the indicator function that takes the value 1 if condition x is true and 0 otherwise; thus, $R\times 1_{\{k(t-1)=0,\ k(t)=1\}}$ is the switching cost incurred if the server is turned on at time t.

In order to treat switching costs, we introduce one more parameter c, which is the accumulated holding cost within the current server state. The cost c is given by

$$c = h\sum_{i=1}^{n} s_i\ , \tag{4.4}$$

where n is the total time the server rests in the current state k starting from the last time it was switched to that state, i is the ith consecutive time unit within the current server state, and s_i is the number of customers present in the system in the ith time unit. By comparing the accumulated holding cost and the switch-on cost R, we can determine the time beyond which it is no longer beneficial to keep the server off. We will explain the role of c in detail soon.

Derivation of Heuristic Decision Criteria

In this section, we present a number of decision criteria, which are derived by examining simple versions of the problem, where the state variables and/or parameters of the process take extreme (i.e., very large or very small) values. The optimal decisions for the original (complex) process will then be derived by using fuzzy logic to aggregate all these decision criteria.

Recall that we have already adopted part of the optimal policy, which dictates that the server is always kept on whenever customers are present in the system and it is switched off when the system empties. Thus, we only need to specify when to turn on the server.

We examine three distinct cases.

1) If there are no switching costs, it is trivially optimal to turn on the server when a customer arrives. The existence of a switch-on cost may lead to a delay in turning on the server even though there may be customers present in the system. The optimal turn-on time will be determined by the relationships between holding and switching costs. Clearly, we have the following:

Criterion (1): When the accumulation of holding cost c during a vacation period is high enough to compete with the switching cost, it is optimal to turn on the server.

2) The traffic intensity $\rho = \lambda/\mu$ is the average use of the server over an infinite horizon $T \rightarrow \infty$. The quantity $1 - \rho$ is the fraction of the time the server can be idle or dormant. If $\rho \geq 1$, that is, the arrival rate is greater than or equal to the service rate, then the number of customers in the system will grow without bound. If we decide to keep the server off during a fraction of the total operation time T, then we will save a *constant cost rate* r (because the service cost will be zero during this fraction), but the holding cost rate will keep on *growing* at an average rate of $h\lambda$. As a result, the average holding cost rate, expressed by the first summation term in Equation (4.3), will dominate the other cost rates of g after some initial transient period. Therefore, in this case, it is optimal to keep the server always on. Using the same arguments, we can show that when ρ is less than but close to one, again we should urgently turn a dormant server on. On the other hand, when $\lambda = \rho = 0$, it is optimal to serve all customers, if any, initially present in the system and then switch the server off permanently. To avoid this trivial situation, we assume that $\rho > 0$. Finally, if ρ is close to zero, then the server can stay in the dormant state for some time until the accumulation of holding cost satisfies Criterion (1). These observations are summarized by Criterion (2) as follows:

Criterion (2): The higher the ρ, the easier it is to make the decision to turn the server on.

3) If $h = 0$, keeping the server always off achieves the minimum cost $g = 0$, although the number of customers in the system explodes as $T \rightarrow \infty$. Again, this is a trivial situation; hence, we assume that $h > 0$. Using the same arguments as in the previous case, we reach the following:

Criterion (3): The higher the h, the easier it is to make the decision to turn the server on.

It is interesting to note that the rule base is independent of the service cost rate r. This is because of the property of the removable server model where the single server under a long-run criterion is busy with probability ρ at any time instant, independent of the service cost rate.

Next, we use the above criteria to build a rule base of the fuzzy controller and derive the functional forms of the linguistic values for each input and output.

Rule Base

The inputs of the rule base are the parameters c, h, and ρ. Each parameter is represented by four linguistic values, ZO, PS, PM, and PB, which stand for “zero,” “positive small,” “positive medium,” and “positive big,” respectively. The complete rule base comprises $4^3 = 64$ rules. The output of each rule, denoted by d, is the decision concerning whether to turn the server on and it is represented by the linguistic values YES and NO.

According to Criteria (1)–(3), whenever all input parameters are zero, it is optimal to keep the server off. This gives us the first rule of the rule base

Rule 1: *if c is ZO and h is ZO and ρ is ZO, then d is NO.*

Dual to the above is the last rule of the rule base

Rule 64: *if c is PB and h is PB and ρ is PB, then d is YES.*

All other rules fall within these two extreme cases. The complete rule base is shown in Table 4.1.

Table 4.1. Rule base.

Rules 1–16				*Rules 17–32*				*Rules 33–48*				*Rules 49–64*			
c	h	ρ	d	c	h	ρ	d	c	h	ρ	d	c	h	ρ	d
ZO	ZO	ZO	NO	ZO	ZO	PS	NO	ZO	ZO	PM	NO	ZO	ZO	PB	YES
PS	ZO	ZO	NO	PS	ZO	PS	NO	PS	ZO	PM	YES	PS	ZO	PB	YES
PM	ZO	ZO	NO	PM	ZO	PS	YES	PM	ZO	PM	YES	PM	ZO	PB	YES
PB	ZO	ZO	YES	PB	ZO	PS	YES	PB	ZO	PM	YES	PB	ZO	PB	YES
ZO	PS	ZO	NO	ZO	PS	PS	NO	ZO	PS	PM	YES	ZO	PS	PB	YES
PS	PS	ZO	NO	PS	PS	PS	YES	PS	PS	PM	YES	PS	PS	PB	YES
PM	PS	ZO	YES	PM	PS	PS	YES	PM	PS	PM	YES	PM	PS	PB	YES
PB	PS	ZO	YES	PB	PS	PS	YES	PB	PS	PM	YES	PB	PS	PB	YES
ZO	PM	ZO	NO	ZO	PM	PS	YES	ZO	PM	PM	YES	ZO	PM	PB	YES
PS	PM	ZO	YES	PS	PM	PS	YES	PS	PM	PM	YES	PS	PM	PB	YES
PM	PM	ZO	YES	PM	PM	PS	YES	PM	PM	PM	YES	PM	PM	PB	YES
PB	PM	ZO	YES	PB	PM	PS	YES	PB	PM	PM	YES	PB	PM	PB	YES
ZO	PB	ZO	YES	ZO	PB	PS	YES	ZO	PB	PM	YES	ZO	PB	PB	YES
PS	PB	ZO	YES	PS	PB	PS	YES	PS	PB	PM	YES	PS	PB	PB	YES
PM	PB	ZO	YES	PM	PB	PS	YES	PM	PB	PM	YES	PM	PB	PB	YES
PB	PB	ZO	YES	PB	PB	PS	YES	PB	PB	PM	YES	PB	PB	PB	YES

To complete the rule base, we applied the following heuristic procedure. To each linguistic value we assign an integer weight: ZO$\rightarrow$0, PS$\rightarrow$1, PM$\rightarrow$2, and PB$\rightarrow$3. For each rule, we compute the sum of the weights associated with the linguistic values of its input parameters. We set $d =$ NO, if the sum is less than or equal to 2; otherwise we set $d =$ YES. Thus, for example, the rule with antecedent part *if c is ZO and h is PS and ρ is PS* has a total weight $0 + 1 + 1 = 2$, and therefore, its consequent part is *d is NO*, whereas when at least one linguistic value is PB, the rule yields *d is YES*.

Membership Functions

The membership functions of the input variables are chosen to be triangular or trapezoidal. Although we use the same linguistic values, ZO, PS, PM, and PB, to represent the inputs, the functional forms of the membership functions of these values depend on the corresponding input and will be determined according to the criteria and rules derived in the previous subsections.

The physical domain of the accumulated cost c is the interval $[0, +\infty)$. Let us consider an extreme situation of the system where h and ρ are close to zero, i.e., these two parameters do not bear any effects on the decision to turn on the server. According to Criterion (1), it is optimal to turn on the server when the accumulation of holding cost c in the period of server vacation is equal to the switching cost R. Hence, we see that for this extreme situation, it is optimal to turn on the dormant server when c is greater than or equal to R. This situation is equivalent to Rule 4 of Table 4.1, *if c is PB and h is ZO and ρ is ZO, then d is YES*. In other words, when $c \geq R$, then c belongs to the fuzzy set PB with membership grade 1.0, or $\mu_{\text{PB}}(c) = 1.0$. Thus, PB for c has a trapezoidal membership function with nucleus $\{c \mid c \geq R\}$. The membership functions of ZO, PS, and PM for c are triangular with their peak values equally distributed between the values 0 and R. Hence, the peak value of ZO is 0, the peak value of PS is $R/3$, and the peak value of PM is $2R/3$. The membership functions of c are shown in Figure 4.2a.

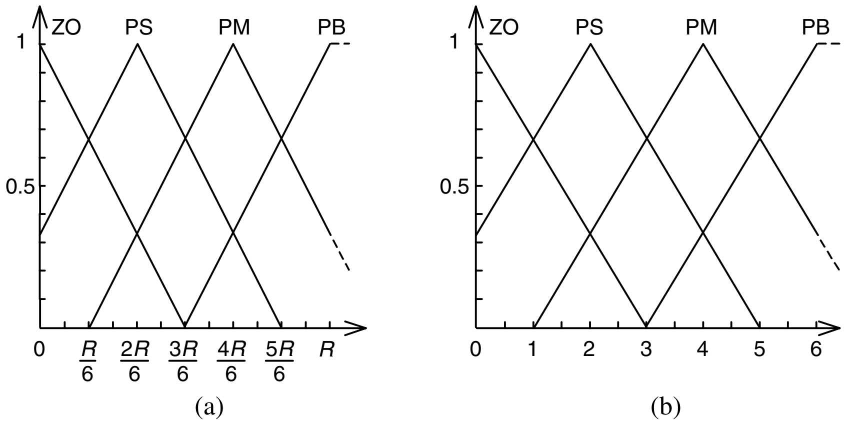


Figure 4.2. Membership functions: (a) original and (b) normalized input variable c.

The physical domains of the inputs are transformed into their normalized counterparts using appropriate *scaling factors*. The normalized values of c are obtained by multiplying the observed values by the scaling factor $6/R$. The membership functions of the normalized variable are shown in Figure 4.2b.

Next, we hypothesize the functional forms of membership functions of the linguistic values for ρ and h. These functions will be specified in part by examining the features of the problem at hand, taking into account our previous choice of membership functions for the accumulated holding cost c.

Arguing as before, it follows that when $h \geq R$, then h belongs to the fuzzy set PB with membership grade 1.0, and when $h \to 0$, then h belongs to ZO again with membership grade 1.0. Finally, as $\rho \in (0, 1)$, PB for ρ with membership grade 1.0 is fixed at 1 and ZO is fixed at 0. Hence, the scaling factor of h is $6/R$ and that of ρ is 6.

Suppose that the system empties and the server is switched off at some time, which, with no loss of generality, we denote by $i = 0$. Hence, the number of customers in the system is $s = s_i = 0$. Suppose further that the server stays off during n consecutive time units, $i = 1, 2, \ldots, n$. Each time unit corresponds to the arrival of a new customer in the system whereby s is increased by one; thus, $s_i = s_{i-1} + 1$. From Equation (4.4), we observe that c increases with n as the sequence $1h$, $3h$, $6h$, …, which is given by

$$c_n = c_{n-1} + hs_n = c_{n-1} + hn = h\frac{n(n+1)}{2}, \quad n = 1, 2, \ldots. \tag{4.5}$$

We observe that the expected number n of arrivals in any interval is proportional to λ, which, in turn, is proportional to ρ. Thus, for example, if ρ is increased to the value $\rho' = 2\rho$, then the expected holding cost in the same period will be $c' = 3c$; if $\rho' = 3\rho$, then $c' = 6c$; and so on. It then follows that the peak values of the linguistic values of ρ are smaller than the peak values of the corresponding linguistic values of c.

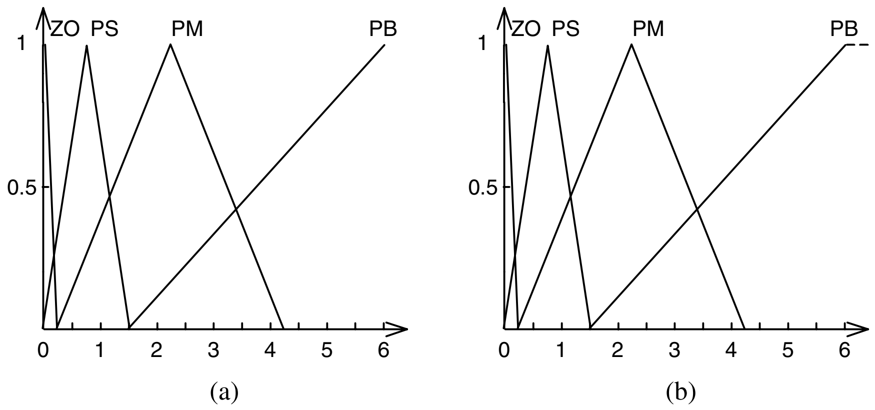


Figure 4.3. Membership functions: (a) ρ and (b) h.

The membership functions of ρ, after normalization, are shown in Figure 4.3a. These functions showed the best performance among various candidates in a number of numerical experiments with different problem instances. From these experiments, we found that the "optimal" membership functions for h, shown in Figure 4.3b, are the same as for ρ, the only difference being that the normalized values of h are not bounded from above.

Defuzzification is the final operation of the knowledge base. At this stage, we have a number of rules whose antecedent parts are YES or NO and the degrees to which they are applicable to a given decision epoch, and we must make a crisp decision YES or NO as to whether to turn the server on or keep it off. One possibility for defuzzification would be to assume that the output d is continuous in the interval [0, 1], where the value 0 corresponds to the crisp decision NO and 1 corresponds to YES. Any value d between 0 and 1 is assigned a membership grade that represents the degree of fulfillment of d to the corresponding decision. If several rules are applicable, then the crisp decision could be obtained by applying any method of defuzzification and integrating the result to 0 or 1 (for example, $d \geq 0.5$ could give YES, otherwise NO).

Another possibility for defuzzification results from the observation that the rule base of Table 4.1 is somehow biased because the majority of its rules yield YES. To compensate for this bias, we consider an alternative defuzzification method whereby a decision YES is made whenever all the applicable fuzzy outputs are YES. Numerical experiments showed that the system incurs a smaller operational cost when this method is followed.

4.2.3 A Numerical Example

We examine an M/M/1 system with the following parameters: arrival rate $\lambda = 1/20$, service rate $\mu = 1/6$, holding cost rate per customer $h = 1.2$, and fixed switching cost $R = 96$.

Here $\rho = 0.3$ is scaled to $0.3 \times 6 = 1.8$, which from Figure 4.3a is interpreted as *ρ is PM with grade 0.825 and PB with grade 0.067*. In addition, $R = 96$ implies that the scaling factors for c and h are $6/R = 0.0625$, and hence, h is scaled down to $1.2 \times 0.0625 = 0.075$, which from Figure 4.3b corresponds to the statement *h is ZO with grade 0.7 and PS with grade 0.1*.

The fuzzy inference procedures are briefly illustrated as follows. At each decision epoch, we compute the current cost accumulation c and fuzzify it into suitable linguistic values. Then the applicable fuzzy rules are determined and the corresponding fuzzy outputs are collected. A decision YES is made whenever all the fuzzy outputs are YES. For example, let us assume that the current cost accumulation $c = 6$. This value is scaled down to $6 \times 0.0625 = 0.375$, which from Figure 4.2b corresponds to ZO with grade 0.875 and PS with grade 0.458. According to the fuzzy rule base (Table 4.1) and Mamdani implication, the fuzzy decisions d are formulated as follows:

If c is ZO with grade 0.875 and h is ZO with grade 0.7 and ρ is PM with grade 0.825, then d is NO with grade 0.7.

If c is PS with grade 0.458 and h is ZO with grade 0.7 and ρ is PM with grade 0.825, then d is YES with grade 0.458.

If c is ZO with grade 0.875 and h is PS with grade 0.1 and ρ is PM with grade 0.825, then d is YES with grade 0.1.

If c is PS with grade 0.458 and h is PS with grade 0.1 and ρ is PM with grade 0.825, then d is YES with grade 0.1.

If c is ZO with grade 0.875 and h is ZO with grade 0.7 and ρ is PB with grade 0.067, then d is YES with grade 0.067.

If c is PS with grade 0.458 and h is ZO with grade 0.7 and ρ is PB with grade 0.067, then d is YES with grade 0.067.

If c is ZO with grade 0.875 and h is PS with grade 0.1 and ρ is PB with grade 0.067, then d is YES with grade 0.067.

If c is PS with grade 0.458 and h is PS with grade 0.1 and ρ is PB with grade 0.067, then d is YES with grade 0.067.

Among the eight fuzzy outputs d, one is NO. Therefore, the decision is NO.

We simulated the system starting from an initial state $k = 0$, $s = 0$, and $c = 0$, employing the fuzzy control at each decision epoch (see the Appendix for an introduction to simulation and Examples A.5 and A.7 for a description of the simulation algorithm). The evolution of k and s for the first 3000 time units is shown in Figure 4.4 (the first 130 time units were considered a warmup period).

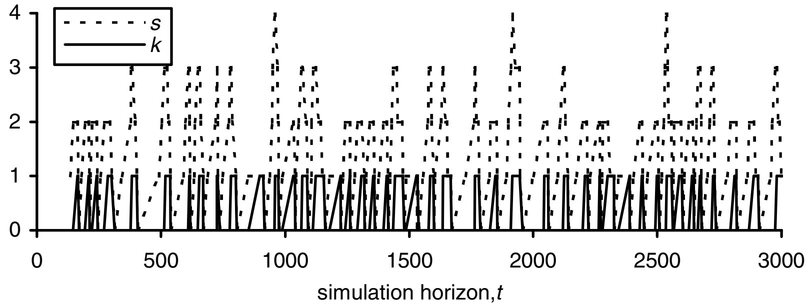


Figure 4.4. Evolution of k and s.

By careful examination of Figure 4.4, it follows that a dormant server is not turned on immediately after a customer arrives (that is, when s is increased from 0 to 1) but after a delay of two or more arrivals.

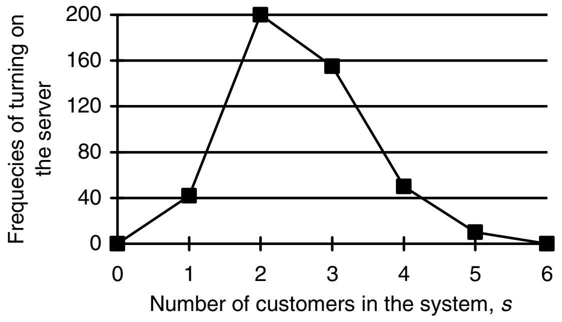


Figure 4.5. Frequency of switches versus number of customers in the system. (IEEE T Syst Man Cyb B, Vol. 29, p. 507, by Yannis A. Phillis and Runtong Zhang. © 1999 by IEEE. Used with permission.)

This gives us an idea of the hysteretic property and is illustrated in Figure 4.5, which shows the number of times the server is turned on as a function of s during a period of 30,000 time units. We see that the server is most often turned on when there are two customers in the system. Hence, we obtain the optimal policy with the hysteretic property as shown in Figure 4.6. By Heyman's Equation (4.2), n^* is 2.366, which does coincide with our result.

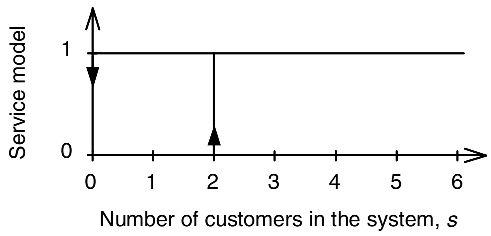


Figure 4.6. The optimal policy. (IEEE T Syst Man Cyb B, Vol. 29, p. 507, by Yannis A. Phillis and Runtong Zhang. © 1999 by IEEE. Used with permission.)

4.2.4 An Extension

We now examine systems that incur additional costs when the server is off or switched off. We shall give an informal proof that the optimal switching policy is the same with or without additional costs.

Consider two queuing systems, each with a single removable server. The first system is identical to the one described in Section 4.2.1 with cost parameters r, R, and h. The second system has the following cost parameters:

1. service cost r_k, $k = 0, 1$, per unit time when the server is on ($k = 1$) or off ($k = 0$), with $r_0 \leq r_1$
2. switching cost R_k, $k = 0, 1$, per unit time incurred whenever the server is turned on ($k = 1$) or off ($k = 0$)
3. holding cost rate h, as in Section 4.2.1

Assume that $r = r_1 - r_0$ and $R = R_0 + R_1$. The optimal policy for the first system is a $(0, N)$ policy. We now show that this policy is also optimal for the second system. We do this by comparing the sample paths of the two systems using a common probability space and *any* control policy. This means that the arrivals, departures, and server switchings are synchronized in both systems. Each system will evolve according to an alternating sequence of uptimes and downtimes in which the server is kept on and off, respectively. During any time interval consisting of an uptime followed by a vacation, the total switching costs are $R = R_1 + R_2$, that is, equal for both systems. Over a long time T, the frequencies of uptimes and vacations differ by no more than one. Hence, the average switching cost rates of the two systems are equal. Furthermore, the second system incurs an additional cost of r_0T (this is so because when the server is on, the second system incurs a cost rate $r_1 = r + r_0$ and when the server is off, the cost rate is r_0). It then turns out that the average cost rate of the second system equals that of the first system plus $(r_0T)/T = r_0$, provided the systems use the same policy. Therefore, as $(0, N)$ is optimal for the first system, it must be optimal for the second one.

4.3 Parallel Servers with Vacations

4.3.1 Problem Description

We consider a queuing system with m exponential servers in parallel and a single queue with infinite capacity, as illustrated in Figure 4.7. The number of working servers can be adjusted to 0, 1,…, or m by turning one or more servers on or off. Customers arrive into the system according to a Poisson process with parameter λ, and service times are independent exponentially distributed random variables with mean $1/\mu$, where $m\mu > \lambda$. This is an M/M/m queuing system with a variable number of working servers that extends the one we discussed in the previous section.

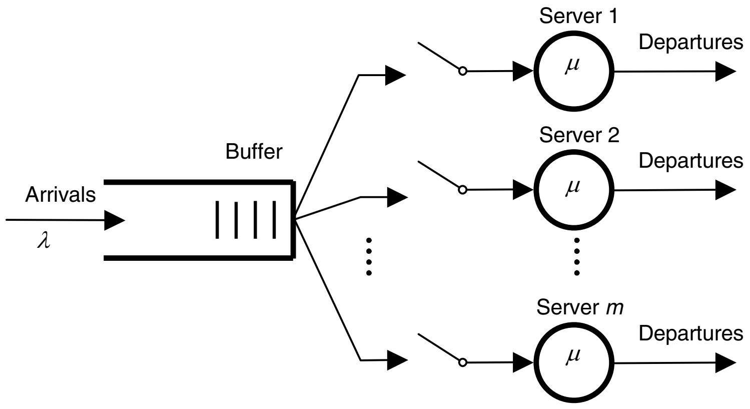


Figure 4.7. M/M/m system with vacations.

The objective is to minimize the average cost over an infinite horizon by adjusting the number of working servers. We consider three types of costs:

1. service cost rK per time unit when K servers are on regardless of busy or idle status, $K = 0, 1, \ldots, m$
2. switching cost R incurred whenever a server is turned on
3. holding cost h per unit time per customer in the system, including those in service (if any)

As in the single-server case, we assume that there is no cost when a server is off or switched off. As discussed in Section 4.2.4, the introduction of service costs r_k when the server is on ($k = 1$) or off ($k = 0$) as well as switching costs R_k when the server is turned on ($k = 1$) or off ($k = 0$), where $r = r_1 - r_0$ and $R = R_0 + R_1$, does not affect the optimal policy.

Bell (1980) has shown that the optimal policy for an M/M/2 queuing system has a *hysteresis* form characterized by four parameters, ν_0, ν_1, N_1, and N_2, which are the numbers of the customers in the system when the number of working servers should be adjusted down to 0, 1, and up to 1, 2, respectively. The optimal policy satisfies the following relationships:

$$\nu_1 \geq \nu_0 \geq -1, N_2 \geq N_1, N_2 \geq \nu_1 \text{ and } N_s \geq s,\ s = 1, 2. \tag{4.6}$$

Note that if $\nu_0 = -1$, then it is optimal for this system to have always one server on, although not necessarily the same one, regardless of busy or idle status. However, complete characterization of the M/M/*m* model and explicit determination of the optimal policy, even for the M/M/2 model, are still open problems.

For r sufficiently high and R, $h > 0$, Bell (1975) proves that it is optimal to never allow more working servers than customers present in the system. Such a policy is called *efficient*. Note that if r is close to zero, then it pays more to keep some servers on to avoid switching costs. For simplicity, we restrict our attention to efficient policies.

4.3.2 Fuzzy Controller

The state of the system is described by (K, s, c), where K is the number of working servers, s is the number of customers in the system, and c is the accumulated holding cost within the current server state.

The variable c is defined as follows. Suppose that K is constant during n consecutive events, that is, arrivals or departures of customers. Let s_i be the number of customers present when event i occurs, $i = 1, 2, \ldots, n$. As we consider efficient policies, we must have $K \leq s_i$, which implies that the K servers are always busy. Hence, $s_i - K$ is nonnegative and equals the number of customers in the queue upon the occurrence of event i. At time n, the accumulated holding cost of these customers is given by

$$c = h\sum_{i=1}^{n}(s_i - K)\,. \tag{4.7}$$

For the special case described in Section 4.2 in which $m = 1$ and decisions are made only when $K = 0$, the above reduces to Equation (4.4).

Note that an efficient control policy is independent of the running cost rate r per server. Indeed, as the arrivals occur at mean rate λ, each customer requires on the average $1/\mu$ time units of service, and no idle servers are permitted, the average running cost over a long period T is $r(\lambda T)(1/\mu)$, for every efficient control policy that eventually serves all incoming customers. Thus, the long-term average running cost rate is $r\lambda/\mu$, and this quantity is independent of the particular policy. Therefore, r need not be considered as input to the controller.

The decision epochs at which service channels may be turned on or off are the times of customer arrivals or service completion. In the single-server system, the inputs of the rule base are the state c and parameters h and ρ, and a basic tenet is, the higher the c, h, or ρ, the easier it is to turn the server on. Here we shall *not* use h as an input parameter of the rule base. We do this to simplify the model and to test the flexibility of our approach and its robustness to modeling assumptions.

The output of the controller is the variation δK of the number of servers. The system amends the number of working servers by simply adding the defuzzified crisp output of δK, $\delta K = -(m - 1), \ldots, -1, 0, 1, \ldots, m - 1$, to the current number of working servers K. The decision criteria are similar to the ones derived in the previous case.

If $R = 0$, it is trivially optimal to adjust the number of working servers to $\min(s, m)$. Indeed, when $s \geq m$, as there are no switching costs, we must use the maximum capacity of the system to avoid holding costs, and when $s < m$, we must reduce the number of running servers to s to avoid the service costs incurred by idle servers. The existence of a switch-on cost may lead to a delay in turning on the server even though there may be customers present in the system. The optimal turn-on time will be determined by the relationships between holding and switching costs. In summary:

Criterion (1): When the accumulation of holding cost c during a vacation period is high enough to compete with the switching cost $R\ \delta K$, it is optimal to turn on δK dormant servers.

Let $\rho = \lambda/(m\mu)$ be the traffic intensity of the system when all its servers are on. The second criterion is the same as in the single-server case.

Criterion (2): The higher the ρ, the easier it is to make the decision to turn on a dormant server.

Finally, when the number K of active servers is less than the number s of customers in the system, then δK should be positive or, at least, nonnegative. By the assumption of efficient policies, when K is larger than s, we switch off all idle servers at no cost. These rules are expressed compactly by:

Criterion (3): The variation δK is an increasing function of $s - K$, and $K + \delta K$ is not greater than s.

Next, we describe the linguistic values and membership functions of the input and output variables.

We use four linguistic values, ZO, PS, PM, and PB, for the input variables ρ, K, c, and s, and seven values for the output variable δK, NB (negative big), NM (negative medium), NS (negative small), ZO, PS, PM, and PB.

The membership functions for ρ are determined following the arguments of the single-server case and are shown in Figure 4.3a.

The physical domain of the number K of active servers is $[0, m]$. A normalized value for K is obtained by multiplying by $6/m$. The corresponding membership functions are shown in Figure 4.8.

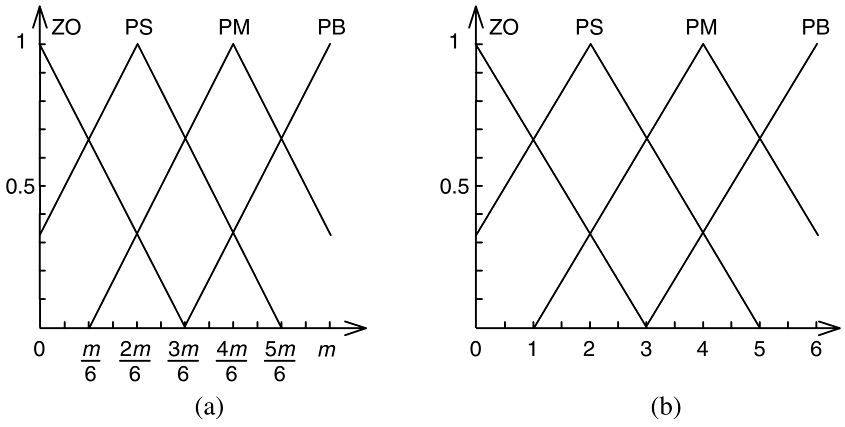


Figure 4.8. Membership functions: (a) original and (b) normalized input variable *K*.

The physical domain of the accumulated cost *c* and the number *s* of customers in the system is [0, ∞). Their normalized values and membership functions are similar to the ones shown in Figure 4.2 (or in Figure 4.8, if we extend the range of the *x*-axis to ∞). When *s* is greater than or equal to *m*, the number of customers is considered PB with grade 1.0. Hence, the scaling factor of *s* is 6/*m*. Finally, as in the single-server case, *c* is declared PB with grade 1.0 when its crisp value is greater than the cost of switching *all* servers on. This cost being *mR*, we obtain the corresponding scaling factor 6/(*mR*).

Finally, the universe of discourse for δK is chosen to be the so-called standard domain [–6, 6]. The membership functions for δK are shown in Figure 4.9.

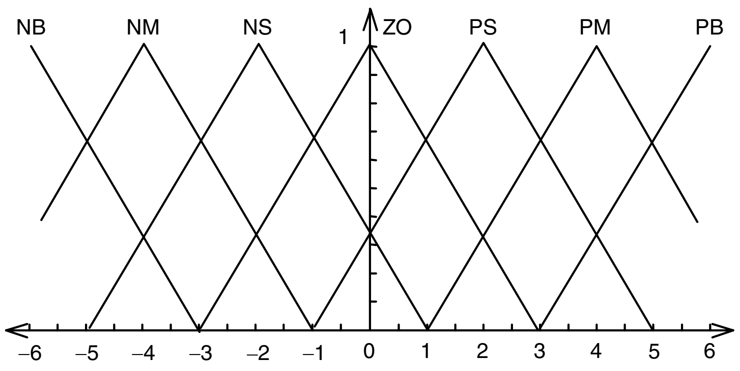


Figure 4.9. Membership functions of the normalized output variable δK.

Table 4.2 shows the rule base of the fuzzy controller, which was determined according to Criteria (1)–(3).

Table 4.2. Rule base

Rules 1–64					*Rules 65–128*					*Rules 129–192*					*Rules 193–256*				
K	s	c	ρ	δK	K	s	c	ρ	δK	K	s	c	ρ	δK	K	s	c	ρ	δK
ZO	ZO	ZO	ZO	ZO	ZO	ZO	ZO	PS	ZO	ZO	ZO	ZO	PM	ZO	ZO	ZO	ZO	PB	ZO
PS	ZO	ZO	ZO	NS	PS	ZO	ZO	PS	NS	PS	ZO	ZO	PM	NS	PS	ZO	ZO	PB	NS
PM	ZO	ZO	ZO	NM	PM	ZO	ZO	PS	NM	PM	ZO	ZO	PM	NM	PM	ZO	ZO	PB	NM
PB	ZO	ZO	ZO	NB	PB	ZO	ZO	PS	NB	PB	ZO	ZO	PM	NB	PB	ZO	ZO	PB	NB
ZO	PS	ZO	ZO	ZO	ZO	PS	ZO	PS	PS	ZO	PS	ZO	PM	PS	ZO	PS	ZO	PB	PM
PS	PS	ZO	ZO	ZO	PS	PS	ZO	PS	ZO	PS	PS	ZO	PM	ZO	PS	PS	ZO	PB	PS
PM	PS	ZO	ZO	NS	PM	PS	ZO	PS	NS	PM	PS	ZO	PM	NS	PM	PS	ZO	PB	NS
PB	PS	ZO	ZO	NM	PB	PS	ZO	PS	NM	PB	PS	ZO	PM	NM	PB	PS	ZO	PB	NM
ZO	PM	ZO	ZO	ZO	ZO	PM	ZO	PS	PS	ZO	PM	ZO	PM	PM	ZO	PM	ZO	PB	PM
PS	PM	ZO	ZO	ZO	PS	PM	ZO	PS	ZO	PS	PM	ZO	PM	PS	PS	PM	ZO	PB	PM
PM	PM	ZO	ZO	ZO	PM	PM	ZO	PS	ZO	PM	PM	ZO	PM	ZO	PM	PM	ZO	PB	PS
PB	PM	ZO	ZO	NS	PB	PM	ZO	PS	NS	PB	PM	ZO	PM	NS	PB	PM	ZO	PB	NS
ZO	PB	ZO	ZO	ZO	ZO	PB	ZO	PS	PS	ZO	PB	ZO	PM	PM	ZO	PB	ZO	PB	PB
PS	PB	ZO	ZO	ZO	PS	PB	ZO	PS	PS	PS	PB	ZO	PM	PS	PS	PB	ZO	PB	PB
PM	PB	ZO	ZO	ZO	PM	PB	ZO	PS	ZO	PM	PB	ZO	PM	PS	PM	PB	ZO	PB	PM
PB	PB	ZO	ZO	ZO	PB	PB	ZO	PS	ZO	PB	PB	ZO	PM	ZO	PB	PB	ZO	PB	PS
ZO	ZO	PS	ZO	ZO	ZO	ZO	PS	PS	ZO	ZO	ZO	PS	PM	ZO	ZO	ZO	PS	PB	ZO
PS	ZO	PS	ZO	NS	PS	ZO	PS	PS	NS	PS	ZO	PS	PM	NS	PS	ZO	PS	PB	NS
PM	ZO	PS	ZO	NM	PM	ZO	PS	PS	NM	PM	ZO	PS	PM	NM	PM	ZO	PS	PB	NM
PB	ZO	PS	ZO	NB	PB	ZO	PS	PS	NB	PB	ZO	PS	PM	NB	PB	ZO	PS	PB	NB
ZO	PS	PS	ZO	PS	ZO	PS	PS	PS	PS	ZO	PS	PS	PM	PS	ZO	PS	PS	PB	PM
PS	PS	PS	ZO	ZO	PS	PS	PS	PS	ZO	PS	PS	PS	PM	ZO	PS	PS	PS	PB	PS
PM	PS	PS	ZO	NS	PM	PS	PS	PS	NS	PM	PS	PS	PM	NS	PM	PS	PS	PB	NS
PB	PS	PS	ZO	NM	PB	PS	PS	PS	NM	PB	PS	PS	PM	NM	PB	PS	PS	PB	NM
ZO	PM	PS	ZO	PS	ZO	PM	PS	PS	PS	ZO	PM	PS	PM	PM	ZO	PM	PS	PB	PB
PS	PM	PS	ZO	PS	PS	PM	PS	PS	PS	PS	PM	PS	PM	PS	PS	PM	PS	PB	PM
PM	PM	PS	ZO	ZO	PM	PM	PS	PS	ZO	PM	PM	PS	PM	ZO	PM	PM	PS	PB	PS
PB	PM	PS	ZO	NS	PB	PM	PS	PS	NS	PB	PM	PS	PM	NS	PB	PM	PS	PB	NS
ZO	PB	PS	ZO	PS	ZO	PB	PS	PS	PM	ZO	PB	PS	PM	PM	ZO	PB	PS	PB	PB
PS	PB	PS	ZO	PS	PS	PB	PS	PS	PS	PS	PB	PS	PM	PM	PS	PB	PS	PB	PB
PM	PB	PS	ZO	PS	PM	PB	PS	PS	PS	PM	PB	PS	PM	PS	PM	PB	PS	PB	PM
PB	PB	PS	ZO	ZO	PB	PB	PS	PS	ZO	PB	PB	PS	PM	PS	PB	PB	PS	PB	PS
ZO	ZO	PM	ZO	ZO	ZO	ZO	PM	PS	ZO	ZO	ZO	PM	PM	ZO	ZO	ZO	PM	PB	ZO
PS	ZO	PM	ZO	NS	PS	ZO	PM	PS	NS	PS	ZO	PM	PM	NS	PS	ZO	PM	PB	NS
PM	ZO	PM	ZO	NM	PM	ZO	PM	PS	NM	PM	ZO	PM	PM	NM	PM	ZO	PM	PB	NM
PB	ZO	PM	ZO	NB	PB	ZO	PM	PS	NB	PB	ZO	PM	PM	NB	PB	ZO	PM	PB	NB
ZO	PS	PM	ZO	PS	ZO	PS	PM	PS	PS	ZO	PS	PM	PM	PS	ZO	PS	PM	PB	PM
PS	PS	PM	ZO	ZO	PS	PS	PM	PS	ZO	PS	PS	PM	PM	ZO	PS	PS	PM	PB	PS
PM	PS	PM	ZO	NS	PM	PS	PM	PS	NS	PM	PS	PM	PM	NS	PM	PS	PM	PB	NS
PB	PS	PM	ZO	NM	PB	PS	PM	PS	NM	PB	PS	PM	PM	NM	PB	PS	PM	PB	NM
ZO	PM	PM	ZO	PM	ZO	PM	PM	PS	PM	ZO	PM	PM	PM	PM	ZO	PM	PM	PB	PB
PS	PM	PM	ZO	PS	PS	PM	PM	PS	PS	PS	PM	PM	PM	PS	PS	PM	PM	PB	PM
PM	PM	PM	ZO	ZO	PM	PM	PM	PS	ZO	PM	PM	PM	PM	ZO	PM	PM	PM	PB	PS
PB	PM	PM	ZO	NS	PB	PM	PM	PS	NS	PB	PM	PM	PM	NS	PB	PM	PM	PB	ZO
ZO	PB	PM	ZO	PM	ZO	PB	PM	PS	PB	ZO	PB	PM	PM	PB	ZO	PB	PM	PB	PB
PS	PB	PM	ZO	PM	PS	PB	PM	PS	PM	PS	PB	PM	PM	PB	PS	PB	PM	PB	PB
PM	PB	PM	ZO	PS	PM	PB	PM	PS	PS	PM	PB	PM	PM	PM	PM	PB	PM	PB	PB
PB	PB	PM	ZO	ZO	PB	PB	PM	PS	ZO	PB	PB	PM	PM	PS	PB	PB	PM	PB	PM
ZO	ZO	PB	ZO	ZO	ZO	ZO	PB	PS	ZO	ZO	ZO	PB	PM	ZO	ZO	ZO	PB	PB	PS
PS	ZO	PB	ZO	NS	PS	ZO	PB	PS	NS	PS	ZO	PB	PM	NS	PS	ZO	PB	PB	NS
PM	ZO	PB	ZO	NM	PM	ZO	PB	PS	NM	PM	ZO	PB	PM	NM	PM	ZO	PB	PB	NM
PB	ZO	PB	ZO	NB	PB	ZO	PB	PS	NB	PB	ZO	PB	PM	NB	PB	ZO	PB	PB	NB
ZO	PS	PB	ZO	PS	ZO	PS	PB	PS	PS	ZO	PS	PB	PM	PS	ZO	PS	PB	PB	PM
PS	PS	PB	ZO	ZO	PS	PS	PB	PS	ZO	PS	PS	PB	PM	ZO	PS	PS	PB	PB	PS
PM	PS	PB	ZO	NS	PM	PS	PB	PS	NS	PM	PS	PB	PM	NS	PM	PS	PB	PB	ZO
PB	PS	PB	ZO	NM	PB	PS	PB	PS	NM	PB	PS	PB	PM	NM	PB	PS	PB	PB	NS
ZO	PM	PB	ZO	PM	ZO	PM	PB	PS	PM	ZO	PM	PB	PM	PM	ZO	PM	PB	PB	PM
PS	PM	PB	ZO	PS	PS	PM	PB	PS	PS	PS	PM	PB	PM	PS	PS	PM	PB	PB	PM
PM	PM	PB	ZO	ZO	PM	PM	PB	PS	ZO	PM	PM	PB	PM	ZO	PM	PM	PB	PB	PS
PB	PM	PB	ZO	NS	PB	PM	PB	PS	NS	PB	PM	PB	PM	NS	PB	PM	PB	PB	ZO
ZO	PB	PB	ZO	PB	ZO	PB	PB	PS	PB	ZO	PB	PB	PM	PB	ZO	PB	PB	PB	PB
PS	PB	PB	ZO	PM	PS	PB	PB	PS	PM	PS	PB	PB	PM	PB	PS	PB	PB	PB	PB
PM	PB	PB	ZO	PS	PM	PB	PB	PS	PS	PM	PB	PB	PM	PM	PM	PB	PB	PB	PB
PB	PB	PB	ZO	ZO	PB	PB	PB	PS	ZO	PB	PB	PB	PM	PS	PB	PB	PB	PB	PM

For comparison with Bell's results for an M/M/2 queue, we will present an example dealing with a two-server queuing model in the next subsection. This example illustrates the excellent accuracy of the fuzzy model when $m = 2$. The same framework can be applied to M/M/m models with m>2, but no comparisons can be made because no analytical solutions exist.

4.3.3 A Numerical Example

We consider an M/M/2 queuing system with vacations, arrival rate $\lambda = 1/30$, service rate of each working server $\mu = 1/40$, switching cost per server $R = 20$, and holding cost rate $h = 0.04$.

As $m = 2$, $\rho = \lambda/(2\mu) = 0.67$. This value is scaled up to $6 \times 0.67 = 4$, which from Figure 4.3a is declared PM with grade 0.125 and PB with grade 0.56. The scaling factor for c is $6/(2R) = 0.15$, and for s and K, it is $6/m = 3$.

Suppose that the current inputs are $c = 1$, $K = 0$, and $s = 1$. Then, c is scaled down to $1 \times 0.15 = 0.15$, which from Figure 4.2b is ZO with membership grade 0.95 and PS with grade 0.38; $K = 0$ is ZO with grade 1.0; and s is scaled up to 3, which from Figure 4.2b is declared PS or PM with grade 0.67. According to Table 4.2, the fuzzy decisions δK are formulated as follows:

If K is ZO with grade 1.0 and s is PS with grade 0.67 and c is ZO with grade 0.95 and ρ is PM with grade 0.125, then δK is PS with grade 0.125.

If K is ZO with grade 1.0 and s is PM with grade 0.67 and c is ZO with grade 0.95 and ρ is PM with grade 0.125, then δK is PM with grade 0.125.

If K is ZO with grade 1.0 and s is PS with grade 0.67 and c is PS with grade 0.38 and ρ is PM with grade 0.125, then δK is PS with grade 0.125.

If K is ZO with grade 1.0 and s is PM with grade 0.67 and c is PS with grade 0.38 and ρ is PM with grade 0.125, then δK is PM with grade 0.125.

If K is ZO with grade 1.0 and s is PS with grade 0.67 and c is ZO with grade 0.95 and ρ is PB with grade 0.56, then δK is PM with grade 0.56.

If K is ZO with grade 1.0 and s is PM with grade 0.67 and c is ZO with grade 0.95 and ρ is PB with grade 0.56, then δK is PM with grade 0.56.

If K is ZO with grade 1.0 and s is PS with grade 0.67 and c is PS with grade 0.38 and ρ is PB with grade 0.56, then δK is PM with grade 0.38.

If K is ZO with grade 1.0 and s is PM with grade 0.67 and c is PS with grade 0.38 and ρ is PB with grade 0.56, then δK is PB with grade 0.38.

From Figure 4.9, the peak values and heights of the fuzzy sets corresponding to the eight decisions δK are $e_1 = e_3 = 2$, $e_2 = e_4 = e_5 = e_6 = e_7 = 4$, $e_8 = 6$ and $f_1 = f_2 = f_3 = f_4 = 0.125$, $f_5 = f_6 = 0.56$, $f_7 = f_8 = 0.38$. By using the height method of defuzzification, the normalized crisp output δK^* is given by

$$\delta K^* = \frac{\sum_{i=1}^{8} e_i f_i}{\sum_{i=1}^{8} f_i} = \frac{9.78}{2.38} = 4.1.$$

As $\delta K^* \in [-6, 6]$ and $\delta K \in \{-(m-1), \ldots, 0, \ldots, m-1\} = \{-1, 0, 1\}$, the crisp output is $4.1 \times (1/6) = 0.68$, which is greater than 0.5. Hence, one server should be turned on.

The operation of the system and the fuzzy controller are simulated for 30,000 time units. From Figure 4.10, we see that when all servers are off ($K = 0$) and there are customers in the system, the fuzzy controller activates one server, usually right after an arrival brings the number of customers to four. Similarly, the number of servers is adjusted from one to two when the number of customers is usually six. Thus, we obtain a hysteretic policy. In addition, the controller always adjusts the number of working servers down to one when a service completion leaves one customer behind or down to zero when the system is empty.

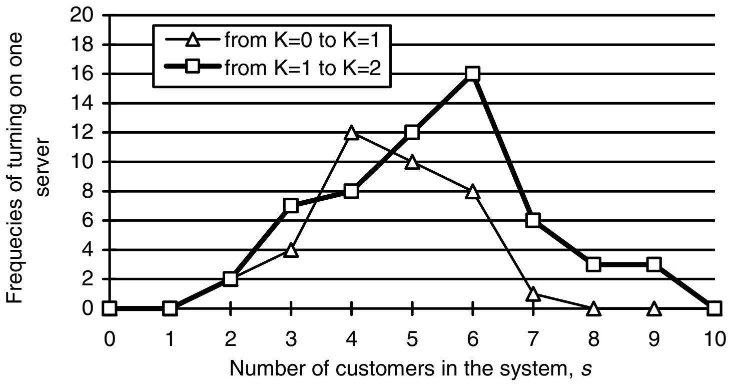


Figure 4.10. Frequency of switches versus number of customers in the system. (IEEE T Syst Man Cyb B, Vol. 29, p. 508, by Yannis A. Phillis and Runtong Zhang. © 1999 by IEEE. Used with permission.)

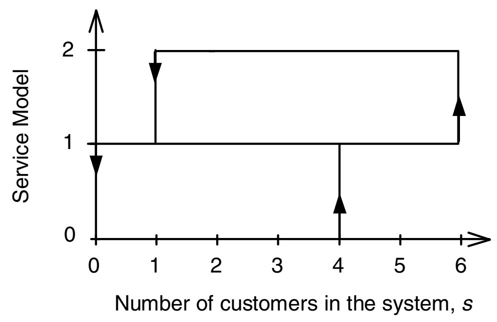


Figure 4.11. The hysteretic policy. (IEEE T Syst Man Cyb B, Vol. 29, p. 509, by Yannis A. Phillis and Runtong Zhang. © 1999 by IEEE. Used with permission.)

From the above, we see that the fuzzy controller approximates a hysteresis policy with parameters $\nu_0 = 0$, $\nu_1 = 1$, $N_1 = 4$, and $N_2 = 6$ (see Figure 4.11). These values satisfy conditions (4.6), which are necessary for an optimal control policy.

4.4 Single Server without Switching Costs

4.4.1 Problem Description

Consider a system with infinite queuing capacity and a single exponential server whose rate can be adjusted to one of a finite set of service rates μ_k, $k = 1, 2, \ldots, m$, ordered as $0 \le \mu_1 < \mu_2 < \ldots < \mu_m < +\infty$. Customers arrive according to a Poisson process with parameter λ. There is a holding cost rate h for each customer in the system and a service cost rate r_k when service rate μ_k is used.

The server may decide which type of service rate k is applied based on the state of the system. The objective is to find an optimal control policy that minimizes the average cost over an infinite time horizon.

We assume that the maximum rate satisfies $\mu_m > \lambda$. Thus, a policy that always uses the maximum service rate is stable; that is, the number of customers in the system is finite at any time. We also assume that the higher the service rate, the higher the corresponding service cost; that is, $0 \le r_1 < r_2 < \ldots < r_m < +\infty$.

The state of the system is described by (s, k), where s is the number of customers in the system and k is the current service type. The decision process is a semi-Markov decision process.

Crabill (1972) and Lippman (1973) have proved that the optimal policy for this problem is a stationary *increasing* policy, whereby more customers in the system lead to a faster service rate. Furthermore, certain conditions exist for excluding certain service rates from consideration as an optimal decision for any state. Define $H_1 = -\infty$, $H_{m+1} = \infty$, and

$$H_k = \frac{r_k - r_{k-1}}{\mu_k - \mu_{k-1}}, \quad k = 2, \ldots, m. \tag{4.8}$$

Crabill (1974) proved that if $H_k > H_{k+1}$, then it is not optimal to employ rate μ_k at any state. When $H_k = H_{k+1}$, rate μ_k is redundant in the sense that if it is optimal for some state, then μ_{k+1} will also be optimal. In the sequel, we assume that $H_k \le H_{k+1}$ for all k.

A stationary policy is specified by an integer function $f(s)$ that assigns a service rate $\mu_{f(s)}$ to each state s. Under a stationary policy, the system is a birth–death queuing system. The equilibrium probabilities P_s, $s = 0, 1, \ldots$, and the long-run cost rate g are computed from the following equations:

$$P_{s+1} = \frac{\lambda}{\mu_{f(s)}} P_s, \quad \sum_{s=0}^{\infty} P_s = 1, \quad g = \sum_{s=0}^{\infty} (r_{f(s)} + hs) P_s . \tag{4.9}$$

Furthermore, any stationary increasing policy is specified by a sequence of integers s_k, $k = 1, \ldots, m - 1$, where s_k is the threshold number of customers in the system when the service rate increases from μ_k to μ_{k+1}. When $m = 2$, the optimal policy can be found by performing a search of possible s_1 values using Equations (4.9). We shall use this procedure to verify our results.

4.4.2 Fuzzy Controller

The state variable s changes at each customer arrival or service completion. We assume that the service rate of a busy server cannot be changed. Then, the decision epochs coincide with the transition instants of s. This is a typical *N-policy* problem, which means that the service type is set according to the number of customers present in the system.

To incorporate the service cost into the decision-making process, we use the holding cost per time unit hs as one input to the fuzzy controller. A decision that changes or maintains the current service rate k depends on hs and the service cost r_k. The higher the hs, the easier it is to make the decision to turn on a dormant server. Also, the lower the r_k, the easier it is to make the decision to increase the service rate, if needed.

Formally speaking, we choose the current service rate type, $k = 1, 2, \ldots, m$, and the current holding cost per time unit of the system $hs \in (0, \infty)$ as the fuzzy inputs, and the variation δk of the service rate type as the fuzzy output. The universes of discourse for the fuzzified variables k and hs are $[0, 6]$ and $[0, \infty)$, respectively. The universe of discourse for δk is chosen to be the standard domain $[-6, +6]$. The corresponding membership functions for hs, k, and δk are shown in Figures 4.3b, 4.8b, and 4.9. We assign the linguistic value ZO to the input k when the service rate is of type 1, because this is the basic type even when there are no customers present in the system. The server amends its service rate type by simply adding the defuzzified crisp output of $\delta k = -(m-1), \ldots, -1, 0, 1, \ldots m-1$, to the current type k. The fuzzy rule base is shown in Table 4.3.

Table 4.3. Rule base

Rules 1–8			*Rules 9–16*		
hs	k	δk	hs	k	δk
ZO	ZO	ZO	ZO	PM	NM
PS	ZO	PS	PS	PM	NS
PM	ZO	PM	PM	PM	ZO
PB	ZO	PB	PB	PM	PS
ZO	PS	NS	ZO	PB	NM
PS	PS	ZO	PS	PB	NM
PM	PS	PS	PM	PB	NS
PB	PS	PM	PB	PB	ZO

An example is in order to understand the rule base. Because there are no switching costs introduced here, when the service rate type has the highest value μ_m, but the current holding cost per time unit is zero (there are no customers in the system at this moment), the server should be turned down to the lowest service rate μ_1. Hence, the variation of the service rate type is negative big. This is the interpretation of the rule *if hs is ZO and k is PB, then δk is NB*. In a similar fashion, we obtain the remaining rule base.

The holding cost of customers increases in proportion to the number of customers in the system. This is the basis for determining the shape of the fuzzy membership functions for hs. It should also be noted that the fuzzy membership functions for hs are determined for the case where the service cost rate is proportional to the corresponding service rate type. When this relation is not proportional, we should devise fuzzy membership functions for hs working in a case-by-case fashion.

The quantitative determination of the fuzzy sets for hs relies on the observation that it is always better to pay more for a higher service rate, but not to pay more for holding a larger number of customers when a higher cost is asked. Therefore, PB for hs should be equivalent to the difference of the service costs per time unit between those of the lowest and the highest service rate. Hence, PB for hs with membership grade 1.0 is fixed at $r_m - r_1$. Using a scaling factor, we change a physical domain into its normalized counterpart; e.g., $r_m - r_1$ corresponds to 6 in Figure 4.8b by means of a scaling factor. In general, a decision to turn the server from type k to l depends on the difference $r_k - r_l$, not on the individual values r_k or r_l.

4.4.3 A Numerical Example

A system has arrival rate $\lambda = 0.05$, seven available service rates $\mu_k = 0.04$, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10, and corresponding cost rates $r_k = 10$, 20, 30, 40, 50, 60, 70, where $k = 1, 2, \ldots, 7$. The holding cost rate is $h = 2.8$.

As there are seven types of service rate and their universe of discourse is [0, 6], the scaling factor for k is 1 and its normalized value is $k - 1$. Also, $r_1 = 10$ and $r_7 = 70$ imply that the threshold value for which hs is PB with grade 1.0 is $r_7 - r_1 = 60$. Therefore, the scaling factor for hs is 0.1.

The fuzzy control procedure is briefly illustrated as follows. At each decision epoch, the fuzzy logic controller captures the current service type k and the holding cost rate hs and then fuzzifies them into suitable linguistic values. Based on the corresponding fuzzy rules, certain fuzzy decisions are fired. Then the defuzzifier changes the fuzzy decisions into a usable crisp one. Finally, the defuzzified output is added to the current service type to adjust the system behavior.

For example, assume that the current service type is $k = 1$ and the number of customers is $s = 1$. From Figure 4.8b, we see that the normalized value $k - 1 = 0$ corresponds to ZO with grade 1.0 and PS with grade 0.333. The value $hs = 2.8 \times 1 = 2.8$ is scaled down to $2.8 \times 0.1 = 0.28$, which from Figure 4.3b corresponds to PS with grade 0.373 and PM with grade 0.015. According to the fuzzy rule base (Table 4.3) and Mamdani implication, the fuzzy decisions δk are formulated as follows:

If hs is PS with grade 0.373 and k is ZO with grade 1.0, then δk is PS with grade 0.373.

If hs is PS with grade 0.373 and k is PS with grade 0.333, then δk is ZO with grade 0.333.

If hs is PM with grade 0.015 and k is ZO with grade 1.0, then δk is PM with grade 0.015.

If hs is PM with grade 0.015 and k is PS with grade 0.333, then δk is PS with grade 0.015.

From Figure 4.9, the peak values and heights of the fuzzy sets corresponding to the four decisions δk are $e_1 = 2$, $e_2 = 0$, $e_3 = 4$, $e_4 = 2$, and $f_1 = 0.373$, $f_2 = 0.333$,

$f_3 = 0.015, f_4 = 0.015$. By the height method of defuzzification, the crisp output δk^* is given by

$$\delta k^* = \frac{\sum_{i=1}^{4} e_i f_i}{\sum_{i=1}^{4} f_i} = 1.1 \approx 1.$$

Then the next service type is $k + \delta k^* = 1 + 1 = 2$. In words, whenever the queue size reaches one from below, while the current service rate type is 1, the next service rate type will be 2. Working similarly, we obtain the policy shown in Figure 4.12, which is a stationary increasing policy.

For this system, Crabill's formulas (4.8) and the exclusion conditions described in Section 4.4.1 imply that the rates $\mu_2, \mu_3, \ldots, \mu_6$ are redundant. Thus, *an* optimal policy is one that employs the rate $\mu_1 = 0.04$ when $s < s_7$ and the rate $\mu_7 = 0.10$ when $s \geq s_7$, for a suitable threshold s_7. By testing various threshold values using Equations (4.9), we have found that the optimal value for s_7 is 5 and the cost rate is 46.6. The cost obtained from simulation with the fuzzy control policy is about 50, which is very close to the optimal one.

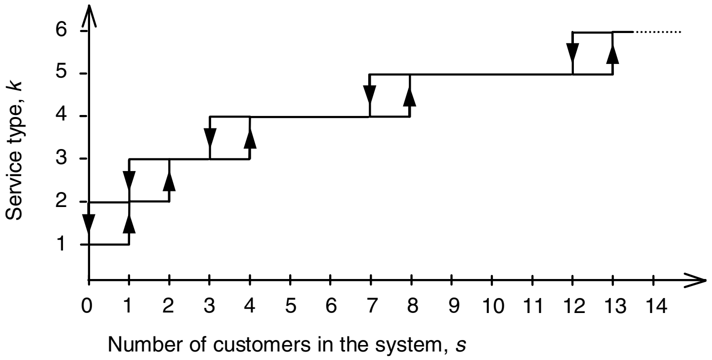


Figure 4.12. The connected increasing policy. (IEEE T Syst Man Cyb B, Vol. 29, p. 510, by Yannis A. Phillis and Runtong Zhang. © 1999 by IEEE. Used with permission.)

4.5 Single Server with Switching Costs

4.5.1 Problem Description

We now introduce switching costs to the system of the previous section. Let $R_{k,l}$ be a fixed nonnegative cost incurred by a change from service type k to service type l, where $l \neq k$. Again, the objective is to find an optimal control policy that minimizes the average cost of the system over an infinite time horizon.

For this problem, Crabill *et al.* (1977) established the optimality of an *increasing hysteretic policy*. This policy generalizes the increasing policy, but the distance between points of optimal switching from k up to $k + 1$ or from $k + 1$ down to k is not necessarily equal to 1. Explicit expressions of the optimal policy have not been found analytically yet.

4.5.2 Fuzzy Controller

The existence of switching costs may lead to a delay in switching the server to another service rate. To include this possibility, we introduce the accumulated holding cost c within the current server state given by Equation (4.4). As previously, the other inputs of the fuzzy controller are the server state s and the current holding cost rate hs. We assume that the service rate of a busy server cannot be changed. Therefore, the decision epochs coincide with the transition epochs of the number of customers s.

Table 4.4. Rule base.

Rules 1–16				*Rules 17–32*				*Rules 33–48*				*Rules 49–64*			
k	hs	c	δk	k	hs	c	δk	k	hs	c	δk	k	hs	c	δk
ZO	ZO	ZO	ZO	ZO	ZO	PS	ZO	ZO	ZO	PM	ZO	ZO	ZO	PB	ZO
PS	ZO	ZO	NS	PS	ZO	PS	NS	PS	ZO	PM	NS	PS	ZO	PB	NS
PM	ZO	ZO	NM	PM	ZO	PS	NM	PM	ZO	PM	NM	PM	ZO	PB	NM
PB	ZO	ZO	NB	PB	ZO	PS	NB	PB	ZO	PM	NB	PB	ZO	PB	NB
ZO	PS	ZO	ZO	ZO	PS	PS	PS	ZO	PS	PM	PS	ZO	PS	PB	PS
PS	PS	ZO	ZO	PS	PS	PS	ZO	PS	PS	PM	ZO	PS	PS	PB	ZO
PM	PS	ZO	NS	PM	PS	PS	NS	PM	PS	PM	NS	PM	PS	PB	NS
PB	PS	ZO	NM	PB	PS	PS	NM	PB	PS	PM	NM	PB	PS	PB	NM
ZO	PM	ZO	ZO	ZO	PM	PS	PS	ZO	PM	PM	PM	ZO	PM	PB	PM
PS	PM	ZO	ZO	PS	PM	PS	PS	PS	PM	PM	PS	PS	PM	PB	PS
PM	PM	ZO	ZO	PM	PM	PS	ZO	PM	PM	PM	ZO	PM	PM	PB	ZO
PB	PM	ZO	NS	PB	PM	PS	NS	ZO	ZO	PM	ZO	PB	PM	PB	NS
ZO	PB	ZO	ZO	ZO	PB	PS	PS	PB	PM	PM	NS	ZO	PB	PB	PB
PS	PB	ZO	ZO	PS	PB	PS	PS	ZO	PB	PM	PM	PS	PB	PB	PM
PM	PB	ZO	ZO	PM	PB	PS	PS	PS	PB	PM	PM	PM	PB	PB	PS
PB	PB	ZO	ZO	PB	PB	PS	ZO	PM	PB	PM	PS	PB	PB	PB	ZO

In general, when the accumulated holding cost c in the current server state is high enough to compete with a value R of switching costs, it is optimal to change the service type. Combining the ideas of Sections 4.2.2 and 4.3.2, we deduce that when the service type is to be switched to a *lower* level, this decision is accomplished immediately. When the service type is to be switched to a *higher* level, we should observe whether the accumulated holding cost has reached a level high enough to compete with the switching cost. Consequently, the higher the c, the easier it is to make the decision to turn the server to a higher service rate.

Following Table 4.3 and the above argument, we construct a new rule base that consists of 64 rules, as shown in Table 4.4. The fuzzy inputs are k, hs, and c, and the output is δk. Their membership functions are shown in Figures 4.8b, 4.2b, 4.3b, and 4.9, respectively. As in Sections 4.2.2 and 4.3.2, PB for hs with membership grade 1.0 is fixed at $r_m - r_1$, whereas PB for c with membership grade 1.0 is fixed at $R_{1,m} + R_{m,1}$.

4.5.3 A Numerical Example

We consider the system of Section 4.4.3 where now we introduce fixed switching costs, $R_{k,k+1} = R_{k+1,k} = 12$.

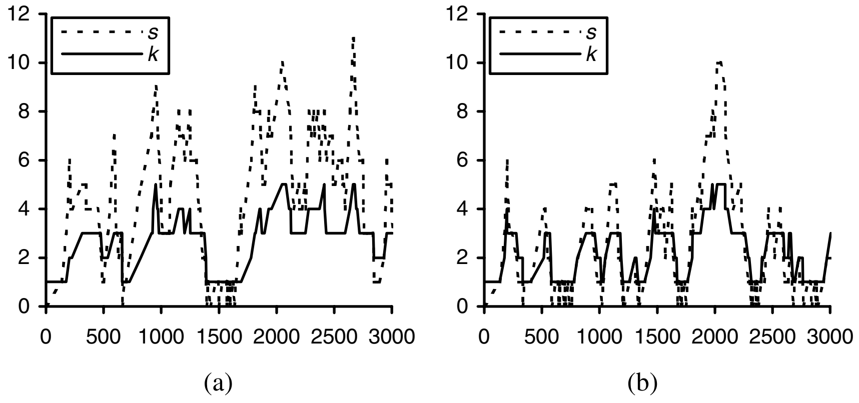


Figure 4.13. Evolution of k and s with (a) and without (b) switching costs.

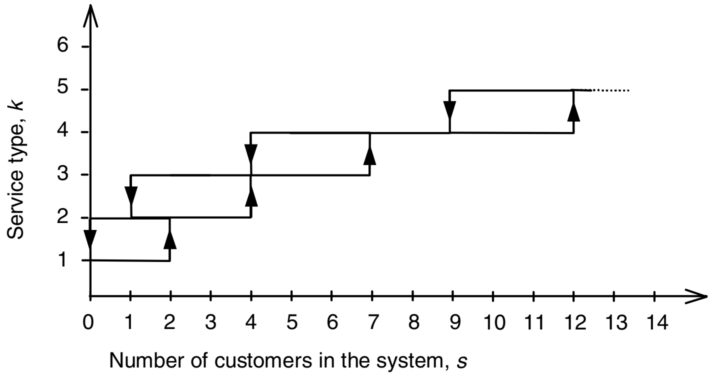


Figure 4.14. The increasing hysteretic policy. (IEEE T Syst Man Cyb B, Vol. 29, p. 511, by Yannis A. Phillis and Runtong Zhang. © 1999 by IEEE. Used with permission.)

The simulation starts from an initial state $(k, s, c) = (1, 0, 0)$. The evolution of k and s for either case, with and without switching costs, during the first 3000 time units is shown in Figure 4.13. We see that because of switching costs, the service type k in Figure 4.13a is less responsive to changes of the queue lengths as compared with Figure 4.13b, and therefore, the average number of customers in the system is kept at a higher level.

By applying the rules of Table 4.4, we obtain the fuzzy control policy shown in Figure 4.14, which is an increasing hysteretic policy. The hysteretic loops are caused by the switching costs. There are no analytical counterparts to these results.

4.6 Tandem Servers without Service Costs

4.6.1 Problem Description

The simplest queuing network is one with two workstations in tandem, as illustrated in Figure 4.15. Each workstation has one exponential server with its own infinite buffer. Customers arrive in station 1 according to a Poisson process with constant rate λ. Upon completion of service in station 1, customers join station 2, which is served by a server with constant rate μ.

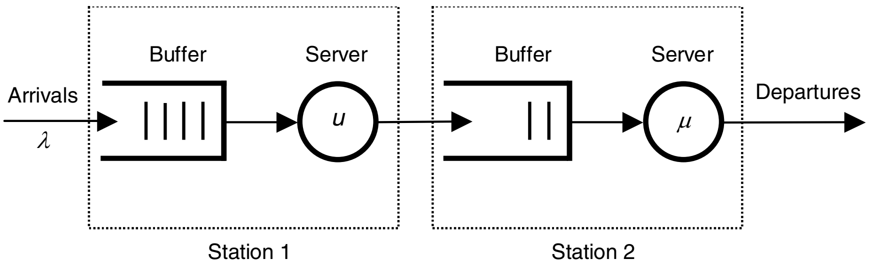


Figure 4.15. Tandem workstations.

The *maximum* service rate of the server in station 1 is a. The system may alter the mean rate u of this server to any value in $[0, a]$. Such decisions are made using information about the state (s_1, s_2), where s_i is the number of customers in station i, $i = 1, 2$. For stability, it is assumed that $\lambda < \min(a, \mu)$.

The holding cost rate per customer in station i is h_i. The objective is to minimize the average cost H over an infinite horizon, which is given by

$$H = \lim_{T\to\infty} \frac{\int_0^T [h_1 s_1(t) + h_2 s_2(t)]\,dt}{T}, \tag{4.10}$$

where $s_i(t)$ is the number of customers in station i at time t.

It is easy to see that if the holding cost rate in station 1 is not less than that in station 2, that is, $h_1 \geq h_2$, then the optimal service rate is $u = a$. Henceforth, it is assumed that $h_1 < h_2$. Rosberg *et al.* (1982) show that because there are no service costs, the optimal policy for this problem is *bang-bang* taking values 0 or a. Specifically, an increasing *switching* function S exists such that $u = a$ is optimal if $s_1 > S(s_2)$ and $u = 0$ is optimal if $s_1 \leq S(s_2)$. However, no explicit method to determine S has been given.

4.6.2 Fuzzy Controller

The state of the system (s_1, s_2) changes upon customer arrivals and service completions. As previously, it is assumed that the service rate of a busy server cannot be changed and the decision epochs coincide with the transition epochs of the state.

As $h_1 < h_2$, it is reasonable to (1) keep customers in the lower cost place (station 1) for the longest possible time and in station 2 for the shortest possible time, if they have to queue somewhere. Conversely, as all customers must eventually enter station 2, we should (2) avoid holding them in station 1 while station 2 is starved. It would be desirable to have always only one customer in service in station 2, unless there are no customers in station 1. However, the stochastic nature of the interarrival and service times makes it impossible to avoid starvation of station 2 while keeping s_2 low.

The fuzzy rule base and the corresponding membership functions are developed by seeking an optimal balance between cases (1) and (2). To achieve this, we develop the rule base using the following criteria:

1. By the bang-bang principle, when h_1s_1 is smaller than h_2s_2, we turn off the server in station 1 to keep s_2 relatively small, and thus, the customers stay in the high-cost station less.
2. If s_1 is relatively large, to avoid situation (2), we turn off the server in station 1 a little later.
3. As $h_1 < h_2$, we should avoid situations where $s_1 < s_2$.

The inputs to the fuzzy logic controller are the numbers of customers in stations 1 and 2, $s_1 \in [0, \infty)$, $s_2 \in [0, \infty)$, and the fuzzy output is the next service rate k of the server in station 1. The fuzzy output k of an optimal control policy has only two values, 0 and a. We henceforth assign two fuzzy sets to the fuzzy output k, ZO and PB, which correspond to the service rates of 0 and a, and four fuzzy sets to each input, ZO, PS, PM, PB. The server in station 1 directly uses the defuzzified crisp output k ($k = 0, 1$) as its next service rate type. A rule base with $4 \times 4 = 16$ rules and in accordance with Criteria (1)–(3) is shown in Table 4.5. Note that if $s_2 = 0$, even when $s_1 = 0$, k is set at the largest value. This is because of the absence of service costs.

Table 4.5. Rule base.

Rules 1–8			*Rules 9–16*		
s_1	s_2	k	s_1	s_2	k
ZO	ZO	PB	PM	ZO	PB
ZO	PS	ZO	PM	PS	PB
ZO	PM	ZO	PM	PM	PB
ZO	PB	ZO	PM	PB	ZO
PS	ZO	PB	PB	ZO	PB
PS	PS	PB	PB	PS	PB
PS	PM	ZO	PB	PM	PB
PS	PB	ZO	PB	PB	PB

Now we determine the membership functions for the various quantities involved in the fuzzy controller. If h_2 is much greater than h_1, the server in station 2 should be starved rarely. The difference $h_2 - h_1$, in a sense, determines how often the server in station 2 will be starved. We adjust s_2 from s_1 to 1 as the difference $h_2 - h_1$ ranges accordingly from 0 to $+\infty$. For example, if $h_1 \approx h_2$, the membership functions for s_1 and s_2 should be similar; if $h_1 << h_2$, PB for s_2 should reach 1 from above in the physical domain. Finally, because of the relationship between the membership functions for s_1 and s_2, the membership functions for s_1 are automatically fixed if the membership functions for s_2 have been specified. Hence, the rule *if s_1 is ZO and s_2 is PS, then k is ZO* in the fuzzy rule base implies that PS for s_2 with membership grade 1.0 is equal to 1.

The membership functions for the fuzzy inputs s_1 and s_2 and the fuzzy output k are shown in Figures 4.2b, 4.3b, and 4.16, respectively.

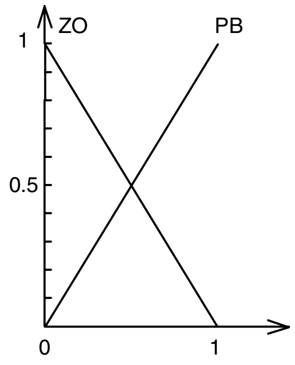


Figure 4.16. Membership function of the output variable k.

4.6.3 A Numerical Example

Consider a two-stage tandem queuing network with arrival rate $\lambda = 1/30$, service rate in station 2 $\mu = 1/20$, service rate in station 1 either 0 or $a = 1/20$, and holding cost rates $h_1 = 0.2$ and $h_2 = 0.4$.

The simulation starts from an initial state $s_1 = s_2 = 0$, and the system performance for the first 5000 time units is shown in Figure 4.17 (see Examples A.6 and A.7 in the Appendix for a description of the simulation algorithm). We see that the number of customers in station 1 is usually kept at a higher level compared with that in station 2. Whenever the number of customers in station 2 is relatively high, the server in station 1 is always turned off.

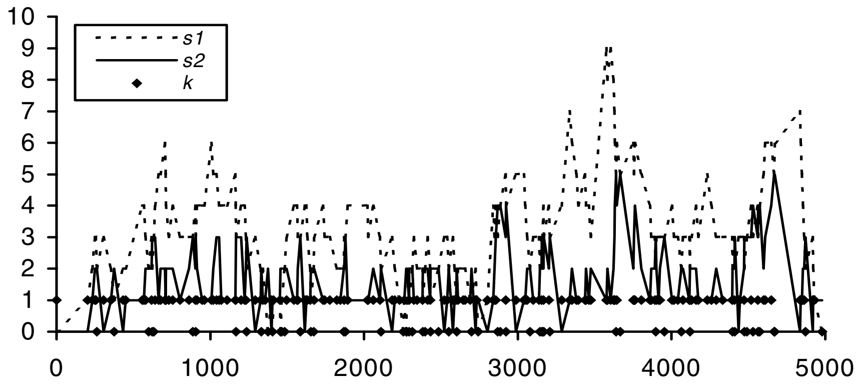


Figure 4.17. Evolution of k and s_i.

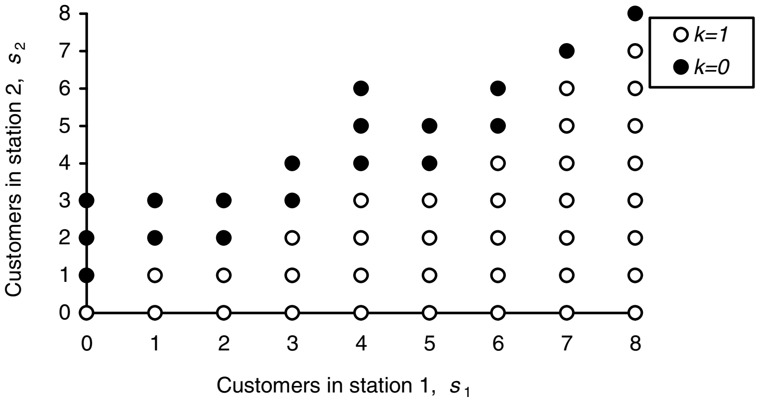


Figure 4.18. Switching policy of the fuzzy controller. (IEEE T Syst Man Cyb B, Vol. 29, p. 513, by Yannis A. Phillis and Runtong Zhang. © 1999 by IEEE. Used with permission.)

Observing the quantitative relationships among s_1, s_2, and k in the first 30,000 time units, we obtain the optimal policy shown in Figure 4.18. There are two regions in the (s_1, s_2) plane, one where s_1 is relatively large and s_2 relatively small and the server in station 1 operates at the highest rate a, and another where the server operates at the lowest rate 0. The solid line that separates these two regions is a switching curve as predicted in Rosberg *et al.* (1982). Given the absence of service costs, the point (0, 0) in the (s_1, s_2) plane is inside the highest rate region ($k = 1$).

4.7 Tandem Servers with Service Costs

4.7.1 Problem Description

We now extend the system of the previous section by introducing service costs whenever the server in station 1 is used.

The service rate u in station 1 is selected from a finite countable set of service rates μ_k, $k = 1, 2, \ldots, m$ where $0 \leq \mu_1 < \ldots < \mu_m < \infty$ and $\mu_m > \lambda$, with corresponding service cost per unit time r_k, where $0 \leq r_1 < \ldots < r_m < \infty$. We assume that $h_1 < h_2$. The system objective is to minimize the average cost over an infinite horizon. This model is new in the literature.

4.7.2 Fuzzy Controller

The state of the system is described by three variables (s_1, s_2, k), where s_1 and s_2 are the contents of stations 1 and 2 and k is the service type in station 1. This is a combination of the tandem system of the previous section and the single-server system without switching costs of Section 4.4, if we view the server in station 1 as a single server with variable service rate.

The fuzzy inputs in the previous two systems, s_1, s_2, k, and h_1s_1 are also the inputs in this problem, where h_1s_1 is the counterpart of hs of Section 4.4. All four fuzzy inputs have the same definitions as well as universes of discourse and membership functions as previously. The variation δk of service rate type is chosen to be the fuzzy output, and its universe of discourse is the standard domain $[-6, 6]$. The server in station 1 amends its type of service rate by simply adding the defuzzified crisp output $\delta k = -(m - 1), \ldots, -1, 0, 1, \ldots, m - 1$ to its current type k.

The membership functions for s_1 are shown in Figure 4.2b, for s_2 and h_1s_1 in Figure 4.3b, for k in Figure 4.8b, and for δk in Figure 4.9. The shape of the membership functions for the fuzzy input s_2 is determined following the ideas of Section 4.6.2. The membership functions for s_1 are automatically fixed according to the membership functions for s_2, where PS for s_2 with membership grade 1.0 is 1. Finally, PB for h_1s_1 with membership grade 1.0 is fixed at $r_m - r_1$.

The controller is developed using the following criteria:

(i) When station 1 is congested compared with station 2, the controller should increase the service rate at station 1. Hence, the service rate of station 1 is an increasing function of $s_1 - s_2$.

(ii) The higher the inventory cost rate at station 1, the easier it is to make a decision to increase the service rate.

A fuzzy rule base that is in line with these criteria is shown in Table 4.6.

Table 4.6. Rule base.

Rules 1–64					*Rules 65–128*					*Rules 129–192*					*Rules 193–256*				
s_1	s_2	k	h_1s_1	δk	s_1	s_2	k	h_1s_1	δk	s_1	s_2	k	h_1s_1	δk	s_1	s_2	k	h_1s_1	δk
ZO	ZO	PB	ZO	NB	ZO	ZO	ZO	PS	ZO	ZO	ZO	ZO	PM	ZO	ZO	ZO	ZO	PB	ZO
PS	ZO	PB	ZO	NB	PS	ZO	ZO	PS	PS	PS	ZO	ZO	PM	PS	PS	ZO	ZO	PB	PS
PM	ZO	PB	ZO	NB	PM	ZO	ZO	PS	PS	PM	ZO	ZO	PM	PM	PM	ZO	ZO	PB	PM
PB	ZO	PB	ZO	NB	PB	ZO	ZO	PS	PS	PB	ZO	ZO	PM	PM	PB	ZO	ZO	PB	PB
ZO	PS	PB	ZO	NB	ZO	PS	ZO	PS	ZO	ZO	PS	ZO	PM	ZO	ZO	PS	ZO	PB	ZO
PS	PS	PB	ZO	NB	PS	PS	ZO	PS	ZO	PS	PS	ZO	PM	ZO	PS	PS	ZO	PB	ZO
PM	PS	PB	ZO	NB	PM	PS	ZO	PS	PS	PM	PS	ZO	PM	PS	PM	PS	ZO	PB	PS
PB	PS	PB	ZO	NB	PB	PS	ZO	PS	PS	PB	PS	ZO	PM	PM	PB	PS	ZO	PB	PM
ZO	PM	PB	ZO	NB	ZO	PM	ZO	PS	ZO	ZO	PM	ZO	PM	ZO	ZO	PM	ZO	PB	ZO
PS	PM	PB	ZO	NB	PS	PM	ZO	PS	ZO	PS	PM	ZO	PM	ZO	PS	PM	ZO	PB	ZO
PM	PM	PB	ZO	NB	PM	PM	ZO	PS	ZO	PM	PM	ZO	PM	ZO	PM	PM	ZO	PB	ZO
PB	PM	PB	ZO	NB	PB	PM	ZO	PS	PS	PB	PM	ZO	PM	PS	PB	PM	ZO	PB	PS
ZO	PB	PB	ZO	NB	ZO	PB	ZO	PS	ZO	ZO	PB	ZO	PM	ZO	ZO	PB	ZO	PB	ZO
PS	PB	PB	ZO	NB	PS	PB	ZO	PS	ZO	PS	PB	ZO	PM	ZO	PS	PB	ZO	PB	ZO
PM	PB	PB	ZO	NB	PM	PB	ZO	PS	ZO	PM	PB	ZO	PM	ZO	PM	PB	ZO	PB	ZO
PB	PB	PB	ZO	NB	PB	PB	ZO	PS	ZO	PB	PB	ZO	PM	ZO	PB	PB	ZO	PB	ZO
ZO	ZO	PB	ZO	NB	ZO	ZO	PS	PS	NS	ZO	ZO	PS	PM	NS	ZO	ZO	PS	PB	NS
PS	ZO	PB	ZO	NB	PS	ZO	PS	PS	ZO	PS	ZO	PS	PM	ZO	PS	ZO	PS	PB	ZO
PM	ZO	PB	ZO	NB	PM	ZO	PS	PS	ZO	PM	ZO	PS	PM	PS	PM	ZO	PS	PB	PS
PB	ZO	PB	ZO	NB	PB	ZO	PS	PS	ZO	PB	ZO	PS	PM	PS	PB	ZO	PS	PB	PM
ZO	PS	PB	ZO	NB	ZO	PS	PS	PS	NS	ZO	PS	PS	PM	NS	ZO	PS	PS	PB	NS
PS	PS	PB	ZO	NB	PS	PS	PS	PS	NS	PS	PS	PS	PM	NS	PS	PS	PS	PB	NS
PM	PS	PB	ZO	NB	PM	PS	PS	PS	ZO	PM	PS	PS	PM	ZO	PM	PS	PS	PB	ZO
PB	PS	PB	ZO	NB	PB	PS	PS	PS	ZO	PB	PS	PS	PM	PS	PB	PS	PS	PB	PS
ZO	PM	PB	ZO	NB	ZO	PM	PS	PS	NS	ZO	PM	PS	PM	NS	ZO	PM	PS	PB	NS
PS	PM	PB	ZO	NB	PS	PM	PS	PS	NS	PS	PM	PS	PM	NS	PS	PM	PS	PB	NS
PM	PM	PB	ZO	NB	PM	PM	PS	PS	NS	PM	PM	PS	PM	NS	PM	PM	PS	PB	NS
PB	PM	PB	ZO	NB	PB	PM	PS	PS	ZO	PB	PM	PS	PM	ZO	PB	PM	PS	PB	ZO
ZO	PB	PB	ZO	NB	ZO	PB	PS	PS	NS	ZO	PB	PS	PM	NS	ZO	PB	PS	PB	NS
PS	PB	PB	ZO	NB	PS	PB	PS	PS	NS	PS	PB	PS	PM	NS	PS	PB	PS	PB	NS
PM	PB	PB	ZO	NB	PM	PB	PS	PS	NS	PM	PB	PS	PM	NS	PM	PB	PS	PB	NS
PB	PB	PB	ZO	NB	PB	PB	PS	PS	NS	PB	PB	PS	PM	NS	PB	PB	PS	PB	NS
ZO	ZO	PB	ZO	NB	ZO	ZO	PM	PS	NM	ZO	ZO	PM	PM	NM	ZO	ZO	PM	PB	NM
PS	ZO	PB	ZO	NB	PS	ZO	PM	PS	NS	PS	ZO	PM	PM	NS	PS	ZO	PM	PB	NS
PM	ZO	PB	ZO	NB	PM	ZO	PM	PS	NS	PM	ZO	PM	PM	ZO	PM	ZO	PM	PB	ZO
PB	ZO	PB	ZO	NB	PB	ZO	PM	PS	NS	PB	ZO	PM	PM	ZO	PB	ZO	PM	PB	PS
ZO	PS	PB	ZO	NB	ZO	PS	PM	PS	NM	ZO	PS	PM	PM	NM	ZO	PS	PM	PB	NM
PS	PS	PB	ZO	NB	PS	PS	PM	PS	NM	PS	PS	PM	PM	NM	PS	PS	PM	PB	NM
PM	PS	PB	ZO	NB	PM	PS	PM	PS	NS	PM	PS	PM	PM	NS	PM	PS	PM	PB	NS
PB	PS	PB	ZO	NB	PB	PS	PM	PS	NS	PB	PS	PM	PM	ZO	PB	PS	PM	PB	ZO
ZO	PM	PB	ZO	NB	ZO	PM	PM	PS	NM	ZO	PM	PM	PM	NM	ZO	PM	PM	PB	NM
PS	PM	PB	ZO	NB	PS	PM	PM	PS	NM	PS	PM	PM	PM	NM	PS	PM	PM	PB	NM
PM	PM	PB	ZO	NB	PM	PM	PM	PS	NM	PM	PM	PM	PM	NM	PM	PM	PM	PB	NM
PB	PM	PB	ZO	NB	PB	PM	PM	PS	NS	PB	PM	PM	PM	NS	PB	PM	PM	PB	NS
ZO	PB	PB	ZO	NB	ZO	PB	PM	PS	NM	ZO	PB	PM	PM	NM	ZO	PB	PM	PB	NM
PS	PB	PB	ZO	NB	PS	PB	PM	PS	NM	PS	PB	PM	PM	NM	PS	PB	PM	PB	NM
PM	PB	PB	ZO	NB	PM	PB	PM	PS	NM	PM	PB	PM	PM	NM	PM	PB	PM	PB	NM
PB	PB	PB	ZO	NB	PB	PB	PM	PS	NM	PB	PB	PM	PM	NM	PB	PB	PM	PB	NM
ZO	ZO	PB	ZO	NB	ZO	ZO	PB	PS	NB	ZO	ZO	PB	PM	NB	ZO	ZO	PB	PB	NB
PS	ZO	PB	ZO	NB	PS	ZO	PB	PS	NM	PS	ZO	PB	PM	NM	PS	ZO	PB	PB	NM
PM	ZO	PB	ZO	NB	PM	ZO	PB	PS	NM	PM	ZO	PB	PM	NS	PM	ZO	PB	PB	NS
PB	ZO	PB	ZO	NB	PB	ZO	PB	PS	NM	PB	ZO	PB	PM	NS	PB	ZO	PB	PB	ZO
ZO	PS	PB	ZO	NB	ZO	PS	PB	PS	NB	ZO	PS	PB	PM	NB	ZO	PS	PB	PB	NB
PS	PS	PB	ZO	NB	PS	PS	PB	PS	NB	PS	PS	PB	PM	NB	PS	PS	PB	PB	NB
PM	PS	PB	ZO	NB	PM	PS	PB	PS	NM	PM	PS	PB	PM	NM	PM	PS	PB	PB	NM
PB	PS	PB	ZO	NB	PB	PS	PB	PS	NM	PB	PS	PB	PM	NS	PB	PS	PB	PB	NS
ZO	PM	PB	ZO	NB	ZO	PM	PB	PS	NB	ZO	PM	PB	PM	NB	ZO	PM	PB	PB	NB
PS	PM	PB	ZO	NB	PS	PM	PB	PS	NB	PS	PM	PB	PM	NB	PS	PM	PB	PB	NB
PM	PM	PB	ZO	NB	PM	PM	PB	PS	NB	PM	PM	PB	PM	NB	PM	PM	PB	PB	NB
PB	PM	PB	ZO	NB	PB	PM	PB	PS	NM	PB	PM	PB	PM	NM	PB	PM	PB	PB	NM
ZO	PB	PB	ZO	NB	ZO	PB	PB	PS	NB	ZO	PB	PB	PM	NB	ZO	PB	PB	PB	NB
PS	PB	PB	ZO	NB	PS	PB	PB	PS	NB	PS	PB	PB	PM	NB	PS	PB	PB	PB	NB
PM	PB	PB	ZO	NB	PM	PB	PB	PS	NB	PM	PB	PB	PM	NB	PM	PB	PB	PB	NB
PB	PB	PB	ZO	NB	PB	PB	PB	PS	NB	PB	PB	PB	PM	NB	PB	PB	PB	PB	NB

4.7.3 A Numerical Example

We examine a two-stage tandem queuing network with parameters $\lambda = 1/30$ and $\mu = 1/20$. The service rate in station 1 can be chosen from a set of four available rates $\mu_k = 1/50$, $1/40$, $1/30$, and $1/20$ with corresponding service costs $r_k = 10$, 20, 30, and 40, $k = 1, 2, 3, 4$. The holding costs per customer per unit time in stations 1 and 2 are $h_1 = 0.2$ and $h_2 = 0.4$.

The simulation is started from an initial state $s_1 = s_2 = 0$ and $k = 1$. The system performance for the first 5000 time units is shown in Figure 4.19. We see that the number of customers in station 1 is kept at a higher level compared with station 2. Also, as expected, k tends to high values as s_1 becomes large or s_2 becomes small. Comparing Figures 4.17 and 4.19, we see that when several rates are available, the curves of s_1 and s_2 are smoother and station 2 is rarely starved.

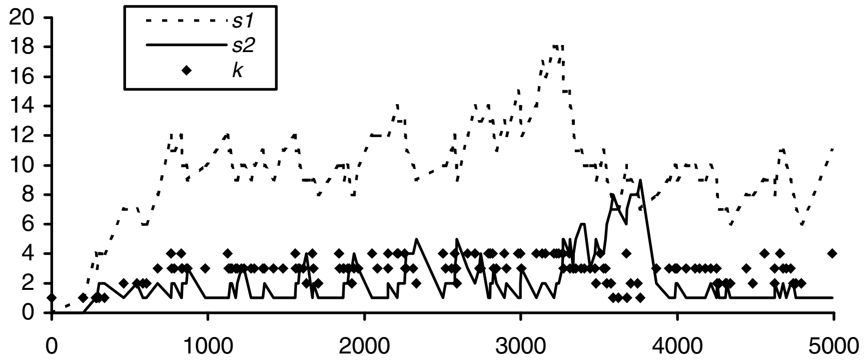


Figure 4.19. Evolution of k and s_i.

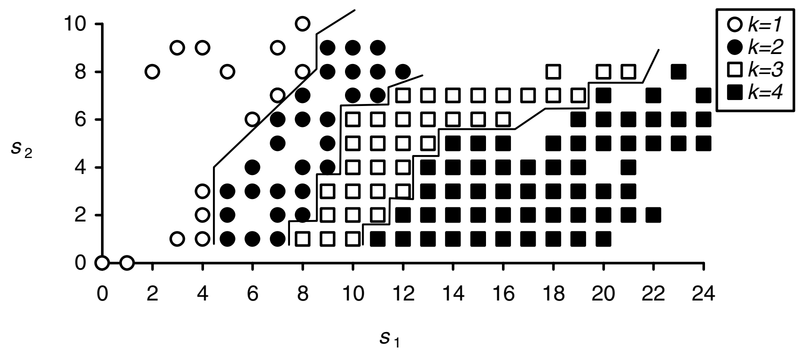


Figure 4.20. Switching policy of the fuzzy controller. (IEEE T Syst Man Cyb B, Vol. 29, p. 513, by Yannis A. Phillis and Runtong Zhang. © 1999 by IEEE. Used with permission.)

Observing the quantitative relationships among s_1, s_2, and k in the first 30,000 time units, we obtain the optimal policy shown in Figure 4.20. There are four regions in the (s_1, s_2) plane corresponding to each of the four available service rates. In general, when s_1 is relatively large and s_2 relatively small, the server in station 1 operates at the highest rate ($k = 4$) and when s_1 is small but s_2 is large, then we use service type $k = 1$.

Unlike Figure 4.18, the point (0, 0) in the (s_1, s_2) plane is inside the lowest rate region (o) because service costs were introduced.

5 Control of the Queue Discipline

5.1 Introduction

In this chapter, we consider the problem of optimal routing of customers in queuing systems with heterogeneous servers in parallel. Five cases are studied in detail:

- Systems in which servers have a common queue with server heterogeneity in service rates, in service functions, and in both service rates and service functions
- Systems with two and three arrival streams and two servers with their own queues

The problems in Sections 5.2 and 5.3 are known and are used as benchmarks to the fuzzy controllers. No analytical solutions are known for the problems in Sections 5.4–5.6. The objective is to assign customers dynamically to idle servers based on the state of the system to minimize the average cost of customer delays. Each problem is solved using fuzzy logic. We use Mamdani implication (Section 2.7) to represent the meaning of "if-then" rules and height defuzzification (Section 3.6) to transform the fuzzy outputs into control actions. Once more, simulation is used to show the details and efficiency of the fuzzy controllers.

5.2 Parallel Servers with Different Service Rates

5.2.1 Problem Description

The simplest routing problem is one in which a buffer of infinite size accommodates a single stream of arriving customers and feeds two parallel servers. The corresponding system is shown in Figure 5.1. Arrivals occur according to a Poisson process with constant rate λ. The buffer is served by two exponential servers with mean service rates μ_i, $i = 1, 2$, where $\mu_1 + \mu_2 > \lambda$. With no loss of generality, it is assumed that $\mu_1 > \mu_2$.

The objective is to dynamically assign queuing customers to idle servers to minimize the mean sojourn time. The sojourn time is the sum of waiting time in queue and service time. By Little's theorem,

$$\text{(mean number of customers in the system)} = \lambda \times \text{(mean sojourn time)},$$

the system objective is equivalent to minimizing the mean number of customers in the system.

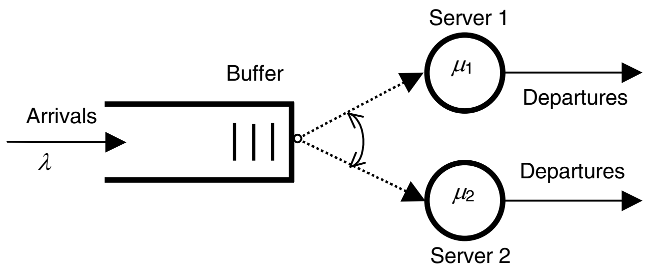


Figure 5.1. A system with two parallel heterogeneous servers.

This process is a continuous time Markov decision process. For this case, Lin and Kumar (1984), Walrand (1984) and Viniotis and Ephremides (1988) prove that it is optimal to use the faster server whenever it becomes available for service and activate the slower server if and only if the total number of customers in the system is *strictly larger* than a critical threshold value n. This policy is of the threshold type and it is called t_n *policy*.

Lin and Kumar (1984) give a method to calculate the value of the optimal threshold. Let $\rho = \lambda/(\mu_1 + \mu_2)$, $a = (\lambda + \mu_1 + \mu_2)$, $b = \lambda/\mu_1$, and

$$b_1 = \frac{a - \sqrt{a^2 - 4\mu_1\lambda}}{2\mu_1},\; b_2 = \frac{a + \sqrt{a^2 - 4\mu_1\lambda}}{2\mu_1},\; c_1 = \frac{1 - b_1}{b_2 - b_1},\; c_2 = \frac{b_2 - 1}{b_2 - b_1}.$$

If J_n is the mean number of customers in the system under a t_n policy, then for $n \geq 3$ and $\lambda \neq \mu_1$,

$$J_n = \frac{\sum_{i=1}^{2} c_i b_i^n \left[\left(\frac{\rho}{1-\rho} - \frac{b_i}{1-b}\right) n + \frac{b_i}{b^n(1-b)^2} + \frac{1}{(1-\rho)^2} - \frac{b_i}{(1-b)^2} - \frac{1}{1-b_i}\right]}{\sum_{i=1}^{2} c_i b_i^n \left[\frac{b_i}{b^{n+1}(1-b)} + \frac{\rho}{1-\rho} - \frac{b_i}{1-b}\right]}. \tag{5.1}$$

For $\lambda = \mu_1$, the limiting result of Equation (5.1) is derived as $b \to 1$ or by adding a very small number to μ_1. A simple algorithm to determine the optimal threshold consists of starting with $n = 3$ and using the above equation to determine n^* such that $J_3 \geq J_4 \geq \ldots \geq J_{n^*}$ and $J_{n^*} < J_{n^*+1}$.

5.2.2 Fuzzy Controller

The state of the system can be described by (x, y_1, y_2), where $x = 0, 1, \ldots$, is the number of customers in the buffer, and $y_i = 0, 1$ indicates whether server i, $i = 1,2$, is idle or busy. Thus, $x + y_1 + y_2$ represents the total number of customers in the system at this state.

The state of the system changes at each customer arrival or service completion. We assume that service is nonpreemptive; that is, a busy server cannot accept a new customer before finishing its current service. In this case, the decision epochs are the times when a customer is at the head of the queue and there is at least one available server. This happens when an arriving customer finds an empty queue and idle servers or a departing customer leaves the system with customers in queue.

As service is nonpreemptive, if there are no customers in the queue ($x = 0$) no matter whether the servers are busy or idle, then no customers are allocated to any server in the system. Furthermore, as $\mu_1 > \mu_2$, it is reasonable to keep the faster server busy when there are customers in the system. Thus, if $x > 0$ and $y_1 = 0$, then a customer is allocated to this server. It then remains to determine the optimal policy when $x > 0$, $y_1 = 1$, and $y_2 = 0$; that is, the slower server is idle and there are customers in queue. We shall attempt to do this using fuzzy logic.

The decision to allocate a customer to server 2 depends on the size of the queue and on the arrival rate. We examine two cases:

(1) When λ is zero, a decision to use the slower server depends solely on the total number of customers in the system. Then, the problem becomes one of finding the smallest buffer level x_0 for which it is optimal to send one and only one customer to server 2. Suppose that at time zero there are $x + 1$ customers in the system (x customers in the buffer *plus* one in server 1) and server 2 is idle. Let $J(x, y)$ denote the expected total cost incurred until the system clears when y out of $x + 1$ customers are sent to server 2 and the rest are sent to the faster server. Then x_0 is the smallest buffer level for which $y = 1$ is optimal; that is,

$$J(x_0, 1) \le J(x_0, 0).$$

The function $J(x, y)$ is the sum of mean sojourn times of the y customers that are sent to server 2 and the $x - y + 1$ customers sent to server 1. The first customer to be sent to server i will stay in the system for $1/\mu_i$ time units on average, the second one $2/\mu_i$, and so on. Therefore,

$$J(x, y) = \frac{1}{\mu_1} + \ldots + \frac{x - y + 1}{\mu_1} + \frac{1}{\mu_2} + \ldots + \frac{y}{\mu_2}$$

$$= \frac{(x - y + 1)(x - y + 2)}{2\mu_1} + \frac{y(y + 1)}{2\mu_2}.$$

By substituting the above into inequality $J(x, 1) \le J(x, 0)$, we obtain

$$\frac{\mu_1}{\mu_2} - 1 \le x.$$

Thus,

$$x \ge x_0 = \left\lceil \frac{\mu_1}{\mu_2} - 1 \right\rceil, \tag{5.2}$$

where $\lceil (.) \rceil$ is the smallest integer greater than or equal to (.). For example, for $\mu_1/\mu_2 = 3.5$ and $\lambda = 0$, we have $x_0 = \lceil 2.5 \rceil = 3$. Hence, when $x = 3$ and the fast server is busy ($y_1 = 1$), it is optimal to send the customer at the head of the queue to server 2 at time zero and assign the remaining ones to server 1.

(2) When λ is positive, it pays more to start using server 2 earlier in anticipation of future arrivals. Thus, the optimal threshold x_λ is bounded from above by x_0.

In view of the above, the rule base and membership functions should be constructed so that when either x or λ or both are relatively large, server 2 will be activated. The fuzzy inputs are the number of queuing customers ($x = 0, 1, \ldots$) and the mean arrival rate of customers $\lambda \in [0, \mu_1 + \mu_2)$. The fuzzy output is the decision $d = 1, 0$ to allocate a customer to the idle server 2. The universes of discourse for the fuzzy inputs x and λ are $[0, \infty)$ and $[0, 6)$, respectively. The universe of discourse for the fuzzy output d is $[0, 1]$. We develop a rule base in Table 5.1, where PVB for the input x indicates "positively very big," which is larger than PB, and YES and NO for the fuzzy output d correspond to 1 and 0. If YES is obtained, a waiting customer is allocated to the idle server 2; otherwise no allocation is made.

Table 5.1. Rule base.

Rules 1–5			*Rules 6–10*			*Rules 11–15*			*Rules 16–20*		
λ	x	d	λ	x	d	λ	x	d	λ	x	d
ZO	ZO	NO	PS	ZO	NO	PM	ZO	NO	PB	ZO	NO
ZO	PS	NO	PS	PS	NO	PM	PS	NO	PB	PS	YES
ZO	PM	NO	PS	PM	NO	PM	PM	YES	PB	PM	YES
ZO	PB	NO	PS	PB	YES	PM	PB	YES	PB	PB	YES
ZO	PVB	YES	PS	PVB	YES	PM	PVB	YES	PB	PVB	YES

The membership functions for the fuzzy inputs x, λ and the fuzzy output d are shown in Figure 5.2a–c, respectively.

The interpretation of Figure 5.2c is that when the defuzzified output d is less than 1.0, the decision is NO. Therefore, in order to use the slow server, the outputs of all rules must be YES. When λ is zero, this happens if x is assigned *only* the largest linguistic value, PVB, as shown in Table 5.1. The situation we are considering is equivalent to the rule *if λ is ZO and x is PVB, then d is YES.*

Now we determine the fuzzy membership functions of x. In view of Equation (5.2), the smallest value for which x is declared merely as PVB when $\lambda = 0$ is $\mu_1/\mu_2 - 1$. From Figure 5.2a, this corresponds to the normalized value 9. The membership functions of linguistic values ZO, PS, PM, and PB of x have triangular forms and become denser as we reach the higher values of x to indicate that the sojourn times increase fast with x. Indeed, when $x = 1$ and a new customer arrives, its sojourn time will be twice as long as that of the previous customer. In general, an arriving customer that finds x others ahead will experience an average delay of $(x + 1)$ times the delay of the first customer in the queue. Therefore, the sum of sojourn times increases with the number x in the queue as 1, 3, 6, …, which is given by $1 + 2 + \ldots + (x + 1)$.

The membership functions of λ are represented by triangles that are equally distributed along the line segment $[0, 6)$, as shown in Figure 5.2b. As $\lambda \in [0, \mu_1 + \mu_2)$, a value $\mu_1 + \mu_2$ for λ is declared PB with membership grade 1.0 and is assigned the normalized value 6.

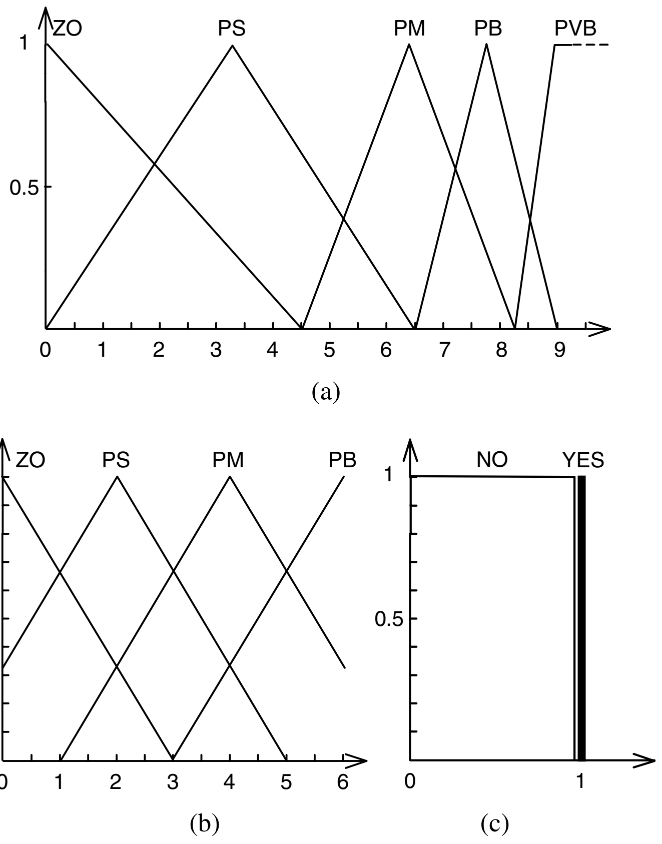


Figure 5.2. Membership functions: (a) x; (b) λ; (c) d.

5.2.3 A Numerical Example

We consider a system with $\lambda = 0.1$, $\mu_1 = 0.1$, and $\mu_2 = 0.02$. We already know that $\mu_1/\mu_2 - 1 = 4$ corresponds to 9 in Figure 5.2a and $\mu_1 + \mu_2 = 0.12$ corresponds to 6 in Figure 5.2b. Hence, the scaling factor is $9/4 = 2.25$ for x and $6/0.12 = 50$ for λ. Therefore, $\lambda = 0.1$ is scaled up to 5, which is interpreted either as *PM with grade 0.67* or as *PB with grade 0.67*.

The fuzzy control procedure is briefly illustrated via an example. Assume that the current number of queuing customers is $x = 5$. This value is scaled up to $5 \times 2.25 = 11.25$, which from Figure 5.2a corresponds to PVB with grade 1. According to the fuzzy rule base (Table 5.1) and Mamdani implication, the fuzzy decisions d are formulated as follows:

If λ is PM with grade 0.67 and x is PVB with grade 1, then d is YES with grade min (0.67, 1) = 0.67.

If λ is PB with grade 0.67 and x is PVB with grade 1, then d is YES with grade min (0.67, 1) = 0.67.

Both fuzzy outputs d are YES. Therefore, the decision is "Yes."

Working similarly, we get the following:

Fuzzy control policy: (1) If $x < 3$ and server 2 is idle, then it is kept idle. (2) If $x \geq 3$, server 1 is busy, and server 2 is idle, then server 2 starts serving the customer at the head of the queue; in which case, the buffer level drops to $x - 1$ immediately.

Clearly, this is a threshold type policy whereby an idle server 2 is used if and only if the number of customers in the system is greater than or equal to 4. Using the notation introduced in Section 5.2.1, this is a t_n threshold policy with $n = 3$.

The evolution of the number of customers in the system and in server 2 for the first 10,000 time units is shown in Figure 5.3. As the fast server is busy when there are customers in the system, the decision d actually corresponds to the state y_2 of server 2. Thus, for clarity, we only record two key variables of the system, $x + y_1 + y_2$ and y_2. We see from the figure that when server 2 is available, a queuing customer is allocated to it only if there are at least four customers in the system. This is a threshold policy with threshold value $n = 3$.

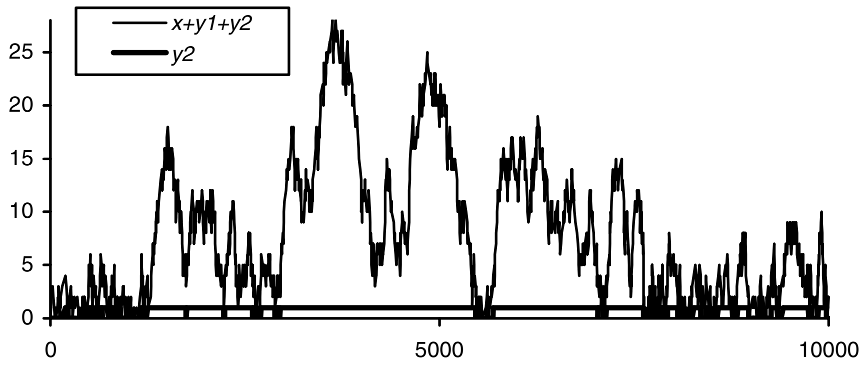


Figure 5.3. Evolution of state variables under the fuzzy control policy.

Table 5.2. Performance measure versus threshold, n.

n	1	2	3	4	5	6
J_n – Equation (5.1)	na	na	5.57	5.75	6.01	6.32
J_n – simulation	5.70	**5.56**	**5.56**	5.70	5.95	6.13

To verify the optimality of this policy we use simulation because Equation (5.1) is valid only for $n \geq 3$. We simulate this system starting from an initial state $(x, y_1, y_2) = (0, 0, 0)$ for 30,000 time units. The mean number of customers in the system for various values of n is shown in Table 5.1. From this table, we see that the optimal threshold value n is either 2 or 3, which indicates that the performance of the fuzzy controller is very close (if not equal) to the optimal.

5.3 Parallel Servers with Heterogeneity in Service Functions

5.3.1 Problem Description

We consider the queuing system of Figure 5.4 with two workstations, $i = 1, 2$, and two classes of customers, $j = 1, 2$. Station i comprises m_i exponential servers having mean processing rates μ. Class j customers arrive at queue j according to a Poisson process with rate λ_j. Customers 1 can be only served by any of the m_1 servers in station 1. Customers 2 can be served by any of the servers at either station. The order of service is irrelevant.

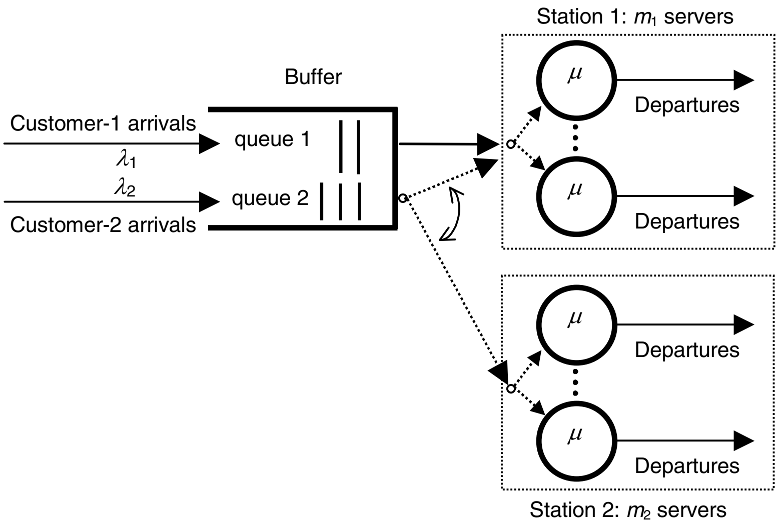


Figure 5.4. A system with two parallel heterogeneous servers and two types of customers.

The holding cost of a customer j per time unit is h_j, $j = 1, 2$. We assume that $h_1 \geq h_2$. To ensure the stability of the system, we assume that $\lambda_1 < m_1\mu$ and $\lambda_1 + \lambda_2 < (m_1 + m_2)\mu$. The objective is to determine the optimal policy, which dynamically assigns customers to idle servers and minimizes the average cost of holding customers. This is a continuous time Markov decision problem.

For this system, Xu *et al.* (1992) have proved the following:

Optimal policy: (1) Whenever possible, assign class i customers to idle servers of station i. (2) If queue 1 is empty and the length of queue 2 exceeds a critical number, then allocate a class 2 customer to an idle server in station 1; the critical number is increasing in the number of busy servers in station 1.

This is a policy of the threshold type .

5.3.2 Fuzzy Controller

The state of the system is described by (x_1, x_2, y_1, y_2, z), where x_i is the number of customers in queue i, $i = 1, 2$, y_i is the number of class i customers receiving service in station i, and z is the number of class 2 customers receiving service in station 1. The total number of customers in the system is $x_1 + x_2 + y_1 + y_2 + z$. All state variables are nonnegative integers.

As in the previous section, we assume that service is nonpreemptive. Consequently, we may restrict the decision epochs to the transition epochs of the state of the system. The following observations eliminate certain trivial situations.

- If there are idle servers in both stations, we need only consider the assignment of customers to stations rather than to individual servers, because all servers have identically distributed processing times.
- If there are customers in queue i, then we allocate as many of them as possible to idle servers of station i, if any.
- Henceforth we will only focus on the optimal assignment of class 2 customers to station 1 when station 2 is full. In this case, there are idle servers in station 1, queue 1 is empty, there are customers in queue 2, and no idle servers in station 2 exist. Symbolically, $y_1 + z < m_1$, $x_1 = 0$, $x_2 > 0$, and $y_2 = m_2$.

We use the following four fuzzy inputs:

- number of class 2 customers in queue $x_2 = 0, 1, \ldots$
- arrival rates λ_1 and λ_2
- number of busy servers in station 1, $b = y_1 + z$, where $b \in [0, m_1]$

The fuzzy output is the decision $d = 1, 0$ of allocating a class 2 customer to an idle server in station 1.

The fuzzy input x_2 is represented by six linguistic values ZO, PS, PM, PB, PVB, and PEB. PEB indicates "positively extremely big," which is larger than PVB. To keep the size of the fuzzy rule base as small as possible, we use only three linguistic values for the remaining input variables ZO, PS, and PB.

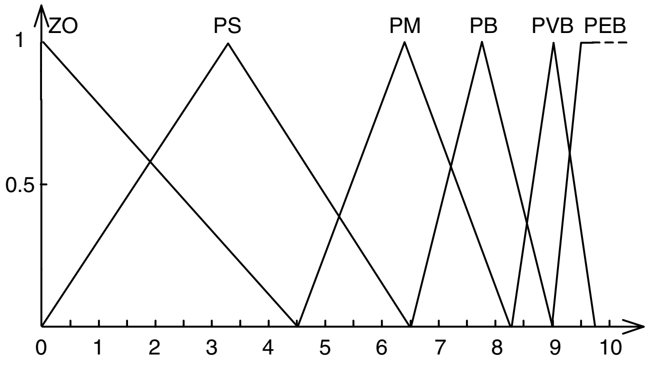
The rule base is constructed by seeking an optimal balance between the following cases: (1) As more customers of class 2 arrive, it pays to forward them to station 1 to avoid the corresponding holding cost. (2) As more customers of class 2 arrive, we must reduce the burden of station 2 because of class 1 customers. Therefore, the larger the λ_2 or x_2, the easier it is to decide "Yes," and the larger the λ_1 or b, the easier it is to decide "No." The fuzzy rule base has 162 rules, and it is shown in Table 5.3.

Table 5.3. Rule base.

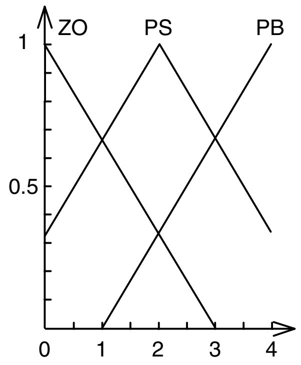
Rules 1–41					*Rules 42–81*					*Rules 82–122*					*Rules 123–162*				
λ_2	x_2	λ_1	b	d	λ_2	x_2	λ_1	b	d	λ_2	x_2	λ_1	b	d	λ_2	X_2	λ_1	b	d
ZO	ZO	ZO	ZO	NO	PB	PS	PB	ZO	NO	ZO	PB	PS	PS	NO	PB	PVB	ZO	PB	NO
PS	ZO	ZO	ZO	NO	ZO	PM	PB	ZO	NO	PS	PB	PS	PS	YES	ZO	PEB	ZO	PB	NO
PB	ZO	ZO	ZO	NO	PS	PM	PB	ZO	NO	PB	PB	PS	PS	YES	PS	PEB	ZO	PB	NO
ZO	PS	ZO	ZO	NO	PB	PM	PB	ZO	YES	ZO	PVB	PS	PS	YES	PB	PEB	ZO	PB	NO
PS	PS	ZO	ZO	YES	ZO	PB	PB	ZO	NO	PS	PVB	PS	PS	YES	ZO	ZO	PS	PB	NO
PB	PS	ZO	ZO	YES	PS	PB	PB	ZO	YES	PB	PVB	PS	PS	YES	PS	ZO	PS	PB	NO
ZO	PM	ZO	ZO	YES	PB	PB	PB	ZO	YES	ZO	PEB	PS	PS	YES	PB	ZO	PS	PB	NO
PS	PM	ZO	ZO	YES	ZO	PVB	PB	ZO	YES	PS	PEB	PS	PS	YES	ZO	PS	PS	PB	NO
PB	PM	ZO	ZO	YES	PS	PVB	PB	ZO	YES	PB	PEB	PS	PS	YES	PS	PS	PS	PB	NO
ZO	**PB**	**ZO**	**ZO**	**YES**	PB	PVB	PB	ZO	YES	ZO	ZO	PB	PS	NO	PB	PS	PS	PB	NO
PS	PB	ZO	ZO	YES	ZO	PEB	PB	ZO	YES	PS	ZO	PB	PS	NO	ZO	PM	PS	PB	NO
PB	PB	ZO	ZO	YES	PS	PEB	PB	ZO	YES	PB	ZO	PB	PS	NO	PS	PM	PS	PB	NO
ZO	PVB	ZO	ZO	YES	PB	PEB	PB	ZO	YES	ZO	PS	PB	PS	NO	PB	PM	PS	PB	NO
PS	PVB	ZO	ZO	YES	ZO	ZO	ZO	PS	NO	PS	PS	PB	PS	NO	ZO	PB	PS	PB	NO
PB	PVB	ZO	ZO	YES	PS	ZO	ZO	PS	NO	PB	PS	PB	PS	NO	PS	PB	PS	PB	NO
ZO	PEB	ZO	ZO	YES	PB	ZO	ZO	PS	NO	ZO	PM	PB	PS	NO	PB	PB	PS	PB	NO
PS	PEB	ZO	ZO	YES	ZO	PS	ZO	PS	NO	PS	PM	PB	PS	NO	ZO	PVB	PS	PB	NO
PB	PEB	ZO	ZO	YES	PS	PS	ZO	PS	NO	PB	PM	PB	PS	NO	PS	PVB	PS	PB	NO
ZO	ZO	PS	ZO	NO	PB	PS	ZO	PS	YES	ZO	PB	PB	PS	NO	PB	PVB	PS	PB	NO
PS	ZO	PS	ZO	NO	ZO	PM	ZO	PS	NO	PS	PB	PB	PS	NO	ZO	PEB	PS	PB	NO
PB	ZO	PS	ZO	NO	PS	PM	ZO	PS	YES	PB	PB	PB	PS	YES	PS	PEB	PS	PB	NO
ZO	PS	PS	ZO	NO	PB	PM	ZO	PS	YES	ZO	PVB	PB	PS	NO	PB	PEB	PS	PB	NO
PS	PS	PS	ZO	NO	**ZO**	**PB**	**ZO**	**PS**	**YES**	PS	PVB	PB	PS	YES	ZO	ZO	PB	PB	NO
PB	PS	PS	ZO	YES	PS	PB	ZO	PS	YES	PB	PVB	PB	PS	YES	PS	ZO	PB	PB	NO
ZO	PM	PS	ZO	NO	PB	PB	ZO	PS	YES	ZO	PEB	PB	PS	YES	PB	ZO	PB	PB	NO
PS	PM	PS	ZO	YES	ZO	PVB	ZO	PS	YES	PS	PEB	PB	PS	YES	ZO	PS	PB	PB	NO
PB	PM	PS	ZO	YES	PS	PVB	ZO	PS	YES	PB	PEB	PB	PS	YES	PS	PS	PB	PB	NO
ZO	PB	PS	ZO	YES	PB	PVB	ZO	PS	YES	ZO	ZO	ZO	PB	NO	PB	PS	PB	PB	NO
PS	PB	PS	ZO	YES	ZO	PEB	ZO	PS	YES	PS	ZO	ZO	PB	NO	ZO	PM	PB	PB	NO
PB	PB	PS	ZO	YES	PS	PEB	ZO	PS	YES	PB	ZO	ZO	PB	NO	PS	PM	PB	PB	NO
ZO	PVB	PS	ZO	YES	PB	PEB	ZO	PS	YES	ZO	PS	ZO	PB	NO	PB	PM	PB	PB	NO
PS	PVB	PS	ZO	YES	ZO	ZO	PS	PS	NO	PS	PS	ZO	PB	NO	ZO	PB	PB	PB	NO
PB	PVB	PS	ZO	YES	PS	ZO	PS	PS	NO	PB	PS	ZO	PB	NO	PS	PB	PB	PB	NO
ZO	PEB	PS	ZO	YES	PB	ZO	PS	PS	NO	ZO	PM	ZO	PB	NO	PB	PB	PB	PB	NO
PS	PEB	PS	ZO	YES	ZO	PS	PS	PS	NO	PS	PM	ZO	PB	NO	ZO	PVB	PB	PB	NO
PB	PEB	PS	ZO	YES	PS	PS	PS	PS	NO	PB	PM	ZO	PB	NO	PS	PVB	PB	PB	NO
ZO	ZO	PB	ZO	NO	PB	PS	PS	PS	NO	ZO	PB	ZO	PB	NO	PB	PVB	PB	PB	NO
PS	ZO	PB	ZO	NO	ZO	PM	PS	PS	NO	PS	PB	ZO	PB	NO	ZO	PEB	PB	PB	NO
PB	ZO	PB	ZO	NO	PS	PM	PS	PS	NO	PB	PB	ZO	PB	NO	PS	PEB	PB	PB	NO
ZO	PS	PB	ZO	NO	PB	PM	PS	PS	YES	ZO	PVB	ZO	PB	NO	PB	PEB	PB	PB	NO
PS	PS	PB	ZO	NO						PS	PVB	ZO	PB	NO					

The universe of discourse for the fuzzy input x_2 is $[0, \infty)$, for λ_1 and λ_2 it is $[0, 4)$, and for b it is $[0, 4]$. The $[0, 4]$ domain is a convenient and frequently used standard domain when we have three fuzzy sets. The universe of discourse for the fuzzy output d is $[0, 1]$; its linguistic values are YES and NO, and the corresponding membership functions are shown in Figure 5.2c. As previously, a decision to allocate a class 2 customer from the queue to an idle server in station 1 is made only if the defuzzified value of d is 1, which means that all rules must give YES.

The membership functions for the fuzzy input x_2 are shown in Figure 5.5a, and for the fuzzy inputs λ_1, λ_2, and b, they are shown in Figure 5.5b.



(a)



(b)

Figure 5.5. Membership functions: (a) x_2 and (b) λ_1, λ_2, and b.

A value m_1 for the input variable b corresponds to the fuzzy set PB with membership grade 1.0. Also, by the stability condition $\lambda_1 < m_1\mu$, $\lambda_1 = m_1\mu$ is declared PB with grade 1.0.

The rationale for the fuzzy membership functions for x_2 and λ_2 is the same as that for x and λ in Section 5.2.2. The scaling factor of x_2 is determined as follows. Consider the rules, which are set in bold on Table 5.3,

- *if λ_1 is ZO and x_2 is PB and λ_2 is ZO and b is ZO, then d is YES*
- *if λ_1 is ZO and x_2 is PB and λ_2 is ZO and b is PS, then d is YES*

According to these rules, $\lambda_1 \approx 0$, $\lambda_2 \approx 0$, and b is less than m_1. Assume that $x_1 = 0$, $x_2 > 0$, and $b = m_1 - 1$; i.e., we want to clear a system while there is exactly one idle server in station 1. If the size of queue 2 is greater than the ratio h_1/h_2, then a decision "No" would incur a holding cost rate h_2x_2 that is greater than h_1. The latter can be interpreted as the additional cost rate if a class 1 customer arrives right after a decision "Yes" is made and finds all servers of station 1 busy. In this case, the decision "Yes" is optimal. Therefore, the value h_1/h_2 corresponds to PB for x_2 with membership grade 1.0.

5.3.3 A Numerical Example

For a queuing system as described in Figure 5.4, we have the following parameters: $\lambda_1 = \lambda_2 = 0.15$, $\mu = 0.15$, $m_1 = 2$, $m_2 = 2$, $h_1 = 0.4$, and $h_2 = 0.1$.

The simulation starts from an initial state (0, 0, 0, 0, 0), employing fuzzy control at each decision epoch. The system performance for the first 1000 units is shown in Figure 5.6.

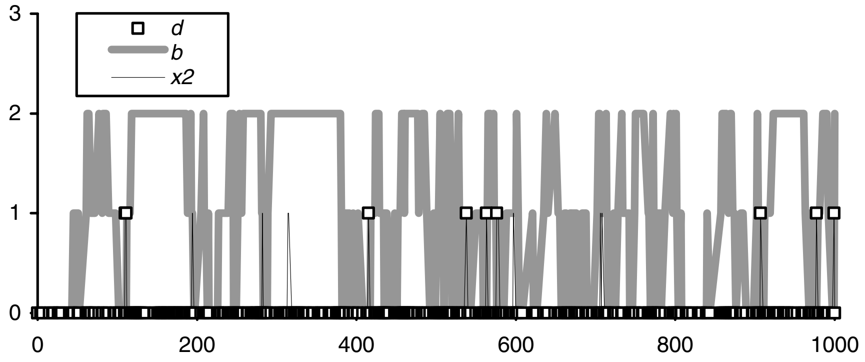


Figure 5.6. Evolution of state variables under the fuzzy control policy.

From the simulation, we observe the following:

(a) Whenever both servers in station 1 are available, a waiting class 2 customer will be allocated to station 1 as long as there are class 2 customers in queue, i.e., $x_2 > 0$. Hence, when $b = 0$, the threshold for x_2 is $n_0 = 0$.

(b) Whenever there is only one idle server in station 1, a waiting class 2 customer is allocated to this server only after queue 2 exceeds 2. In this case, $b = m_1 - 1 = 1$ and the threshold for x_2 is $n_1 = 2$.

This result is identical to that obtained by the traditional value iteration method.

5.4 Parallel Servers with Different Service Rates and Service Functions

5.4.1 Problem Description

In this section, we combine the last two cases by considering parallel-server queuing systems with heterogeneity in both service rates and different service functions. Such systems are new in the literature.

An illustrative figure of such a queuing system could be Figure 5.4, except that the service rates of servers in stations 1 and 2 should be μ_1 and μ_2.

Again, $h_1 \geq h_2$ and the task is to determine the optimal policy, which dynamically assigns queuing customers to idle servers to minimize the expected holding cost. From the fuzzy logic point of view, this problem is an extension of the previous two, and therefore, we shall combine the corresponding results to construct a composite fuzzy controller.

5.4.2 Fuzzy Controller

The state of the system is represented by (x_1, x_2, y_1, y_2, z), where all the components have the same meaning as in Section 5.3.2. The decision epochs are the instants when a customer arrives in the system with idle servers or leaves the system with queuing customers. As previously, we only focus on the optimal assignment of class 2 customers to station 1 when station 2 is full. Because only one main parameter x_2 governs this decision, we intuitively know that the optimal policy is of the threshold type. However, because of the heterogeneity of the service rates μ_1 and μ_2 in the two service stations, we consider two possibilities $\mu_1 \leq \mu_2$ and $\mu_1 > \mu_2$, which will be called Case 1 and Case 2, respectively.

Case 1: As class 1 customers can only be served in station 1, conditions $\mu_1 \leq \mu_2$ and $h_1 \geq h_2$ impede the decision to allocate class 2 customers to this station. However, the arguments of Section 5.3.2 on how the parameters of the two customer classes affect the decisions of customer assignment are valid here. Hence, the fuzzy rules and the fuzzy variables as well as the corresponding membership functions are identical in both cases. It then remains to specify the effect of condition $\mu_1 \leq \mu_2$ on the range of the fuzzy set for x_2. Following the ideas of the previous sections, we examine a specific situation where $\lambda_1 \approx 0$, $\lambda_2 \approx 0$, $x_1 = 0$, and $x_2 > 0$. We want to clear the system while there is exactly one idle server in station 1, i.e., $b = m_1 - 1$, and one new class 1 customer will arrive after the decision has been made. For the problem of Section 5.2, we showed that when $\mu_1 > \mu_2$, the optimal threshold value for the queue size $x + 1$ is $(\mu_1/\mu_2 - 1) + 1 = \mu_1/\mu_2$. When the inequality is reversed, i.e., $\mu_1 \leq \mu_2$, the threshold is adjusted to μ_2/μ_1. For the problem of Section 5.3, we used a threshold value h_1/h_2. For the problem herein, we use the sum of the previous two thresholds. Therefore, the value $(h_1/h_2 + \mu_2/\mu_1)$ is PB for x_2 with membership grade 1.0.

Case 2: Now the service rate μ_1 in station 1 is greater than μ_2 in station 2, which supports the decision of allocating class 2 customers to station 1. Yet, we should avoid delaying class 1 customers because they incur a higher holding cost and only station 1 can serve them. Hence, the optimal policy is a tradeoff between these two antagonistic criteria, which have been used in Sections 5.2 and 5.3. In the first case, a customer is optimally allocated to the slower server only when the queue size is higher than $\mu_1/\mu_2 - 1$, and in the latter, a customer with lower holding cost is optimally allocated to a server only when the number of customers with higher holding cost in the queue exceeds h_1/h_2. These results will be used here. However, to deal with conflicting criteria, we take the difference of the corresponding thresholds. We examine two possibilities:

(a) If $h_1/h_2 \geq \mu_1/\mu_2 - 1$, then the second criterion prevails and class 2 customers will be allocated to station 2 whenever possible.

(b) If $\mu_1/\mu_2 - 1 > h_1/h_2$, class 2 customers will be allocated to station 1 whenever possible.

In Case 2a, we should assign class i customers to idle servers in station i, $i = 1, 2$, whenever possible and allocate a class 2 customer to an idle server in station 1 only if there are idle servers in station 1; that is, queue 1 is empty, and the length of queue 2 exceeds a certain number. Hence, the fuzzy controller is same as that in Section 5.3, and following similar arguments, we set fuzzy set PB for x_2 with grade 1.0 at the value $(h_1/h_2 - \mu_1/\mu_2 + 1)$.

In Case 2b, where the rate of server 1 is very high, i.e., $\mu_1/\mu_2 - 1 > h_1/h_2$, it may be economical to allocate a waiting class 2 customer to an idle server in station 1 even when there are idle type-2 servers. The fuzzy controller should assign both classes of customers to station 1 whenever possible and allocate a waiting class 2 customer to an idle server in station 2 only if the length of queue 2 exceeds a certain number. Henceforth, we will only focus on the optimal assignment of queuing class 2 customers to station 2 when station 1 is full, but there are idle type-2 servers and there are class 2 customers in queue. The fuzzy controller will be developed accordingly using those ideas of Sections 5.2 and 5.3, which are still valid. The queue size x_2 and the arrival rate λ_2 of class 2 customers are still positive factors for the decision of allocating a waiting class 2 customer to an idle type-2 server. However, unlike in the previous two cases, the arrival rate λ_1 of class 1 customers becomes a positive factor for d to become "Yes" and the number of busy servers in station 1 or 2 does not bear any effect on the fuzzy rule base.

We choose as fuzzy inputs the number of class 2 customers in queue $x_2 = 0, 1, \ldots$ and the arrival rates λ_1 and λ_2. Again, the fuzzy output is the decision $d = 1, 0$. The universe of discourse for the fuzzy input x_2 is $[0, \infty)$, for λ_i it is $[0, 4)$, and for the fuzzy output d it is $[0, 1]$. We assign to each of the fuzzy inputs λ_1 and λ_2 three fuzzy sets and to x_2 four fuzzy sets. The complete rule base, which is shown in Table 5.4, has a total of 36 rules and is compiled using previous arguments.

Table 5.4. Rule base for Case 2b.

Rules 1–9				*Rules 10–18*				*Rules 19–27*				*Rules 28–36*			
λ_2	x_2	λ_1	d	λ_2	x_2	λ_1	d	λ_2	x_2	λ_1	d	λ_2	x_2	λ_1	d
ZO	ZO	ZO	NO	ZO	PS	ZO	NO	ZO	PM	ZO	NO	ZO	PB	ZO	YES
ZO	ZO	PS	NO	ZO	PS	PS	NO	ZO	PM	PS	YES	ZO	PB	PS	YES
ZO	ZO	PB	NO	ZO	PS	PB	YES	ZO	PM	PB	YES	ZO	PB	PB	YES
PS	ZO	ZO	NO	PS	PS	ZO	NO	PS	PM	ZO	YES	PS	PB	ZO	YES
PS	ZO	PS	NO	PS	PS	PS	YES	PS	PM	PS	YES	PS	PB	PS	YES
PS	ZO	PB	NO	PS	PS	PB	YES	PS	PM	PB	YES	PS	PB	PB	YES
PB	ZO	ZO	NO	PB	PS	ZO	YES	PB	PM	ZO	YES	PB	PB	ZO	YES
PB	ZO	PS	NO	PB	PS	PS	YES	PB	PM	PS	YES	PB	PB	PS	YES
PB	ZO	PB	NO	PB	PS	PB	YES	PB	PM	PB	YES	PB	PB	PB	YES

The membership functions for the fuzzy inputs λ_1 and λ_2 are shown in Figure 5.5b, and for the fuzzy output d they are shown in Figure 5.2c. The fuzzy input x_2 is represented by four linguistic values, whose membership functions are shown in Figure 5.7. The value $\mu_1/\mu_2 - 1 - h_1/h_2$ for x_2 is PB with grade 1.0.

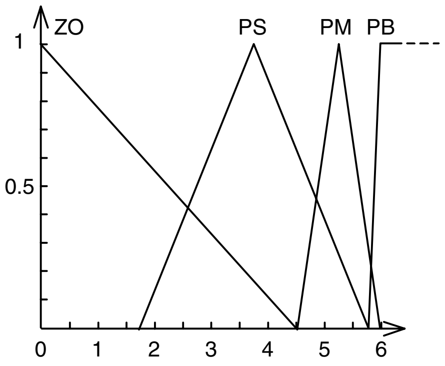


Figure 5.7. Membership functions of the fuzzy input x_2.

The arrival rates of both classes of customers are positive factors to decide "Yes"; therefore, the optimal threshold here is bounded from above by $\mu_1/\mu_2 - 1 - h_1/h_2$.

5.4.3 A Numerical Example

A queuing system with server heterogeneity in both service rates and functions has the following characteristics: $m_1 = 2$ and $m_2 = 2$, $h_1 = 0.4$ and $h_2 = 0.1$, $\mu_1 = 0.05$ and $\mu_2 = 0.2$, and $\lambda_1 = 0.05$ and $\lambda_2 = 0.2$.

For this system, $\mu_1 > \mu_2$ and $h_1/h_2 - \mu_1/\mu_2 + 1 = 2.5 > 0$; hence, we use the fuzzy controller of Case 2a. The simulation under this controller is started from an initial state $(x_1, x_2, y_1, y_2, z) = (0, 0, 0, 0, 0)$, and the system performance for the first 5000 time units is shown in Figure 5.8.

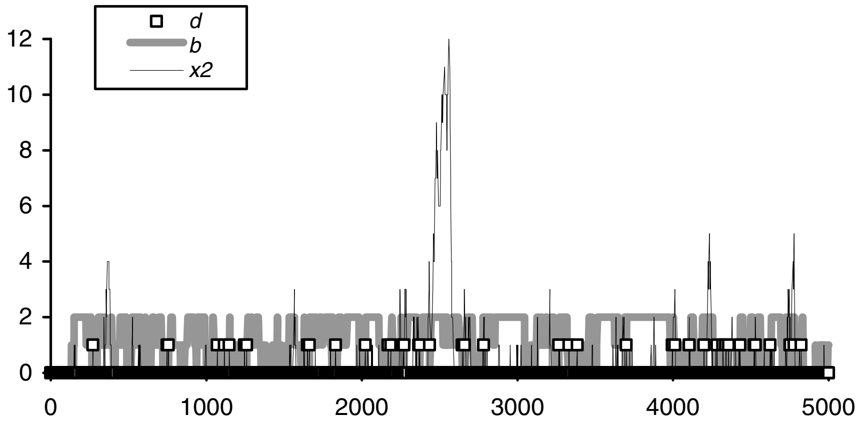


Figure 5.8. Evolution of state variables under the fuzzy control policy.

From the simulation, we observe the following:

(1) Whenever both servers in station 1 are available, a waiting class 2 customer will be allocated to station 1 as long as there are class 2 customers in queue, i.e., $x_2 > 0$. Hence, as in Section 5.3.3, the threshold for x_2 when $b = 0$ is $n_0 = 0$.
(2) Whenever there is only one idle server in station 1, a waiting class 2 customer is allocated to this server after the length of queue 2 exceeds 4. In this case, $b = m_1 - 1 = 1$ and the threshold for x_2 is $n_1 = 4$. This situation occurs only once at time 4778 in Figure 5.8.

5.5 Queuing System with Heterogeneous Servers

5.5.1 Problem Description

The system shown in Figure 5.9 consists of two heterogeneous servers in parallel, called server i, $i = 1, 2$, each with its own buffer of unlimited capacity. There are two classes of customers. Customers of class 2 arrive at queue 2 to be served by server 2. However, a customer of class 1, upon arrival, can be transferred to queue 2 by paying an assignment cost C, or the customer can join queue 1 at no cost. The system incurs a holding cost h_i per customer per time unit in queue i. Customers of class i arrive according to a Poisson process with rate λ_i, and the processing times of server i are exponentially distributed with mean $1/\mu_i$, $i = 1, 2$. We assume that these random variables are mutually independent and $\lambda_i < \mu_i$.

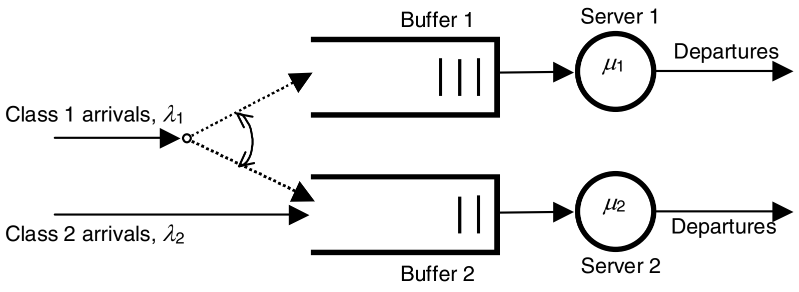


Figure 5.9. Queuing system with heterogeneous servers.

Suppose that the state of the system is observable. The task is to determine how to assign class 1 customers so that the average cost is minimized. This process is a semi-Markov decision process. Koyanagi and Kawai (1995) have shown that the optimal policy is of a switchover type. However, the explicit determination of the optimal policy remains unknown. In the next section, we present a solution to this problem using fuzzy logic.

5.5.2 Fuzzy Controller

As in previous problems, the decision epochs coincide with the arrival times of class 1 customers. The state of the system at the decision epochs is described by (s_1, s_2), where $s_i = 0, 1, 2, \ldots$ is the number of customers in queue i. Let $d = 1, 0$ be the decision of whether to transfer an arriving class 1 customer to queue 2. To develop the fuzzy rule base, we consider three special cases of this problem.

Case (1). Suppose we want to clear a system in which an arriving class 1 customer finds s_i customers in queue i, $i = 1, 2$, and there are no further arrivals pending, i.e., $\lambda_1 = \lambda_2 = 0$. If the customer joins queue 1, the system incurs an additional cost $h_1 s_1/\mu_1$ on average, whereas if the customer is transferred to queue 2, the corresponding cost will be $h_2 s_2/\mu_2 + C$). The condition under which the second decision is optimal is $s \geq C$, where

$$s \triangleq h_1 \frac{s_1}{\mu_1} - h_2 \frac{s_2}{\mu_2}.$$

Hence, the greater the difference s between the customer's expected holding costs in both queues, the easier it is to decide $d = 1$ for an arriving customer.

Case (2). A high arrival rate of class 1 customers strengthens the decision $d = 1$. Indeed, if λ_1 is high, we should transfer the present customer to queue 2 in anticipation of future arrivals, which would increase the length of queue 1 as well as s.

Case (3). In a dual fashion, a higher arrival rate of class 2 customers strengthens the decision $d = 0$.

Table 5.5. Rule base.

Rules 1–16				*Rules 17–32*				*Rules 33–48*				*Rules 49–64*			
s	λ_1	λ_2	d	s	λ_1	λ_2	d	s	λ_1	λ_2	d	s	λ_1	λ_2	d
ZO	ZO	ZO	NO	ZO	ZO	PS	NO	ZO	ZO	PM	NO	ZO	ZO	PB	NO
PS	ZO	ZO	YES	PS	ZO	PS	NO	PS	ZO	PM	NO	PS	ZO	PB	NO
PM	ZO	ZO	YES	PM	ZO	PS	YES	PM	ZO	PM	NO	PM	ZO	PB	NO
PB	ZO	ZO	YES	PB	ZO	PS	YES	PB	ZO	PM	YES	PB	ZO	PB	NO
ZO	PS	ZO	YES	ZO	PS	PS	NO	ZO	PS	PM	NO	ZO	PS	PB	NO
PS	PS	ZO	YES	PS	PS	PS	YES	PS	PS	PM	NO	PS	PS	PB	NO
PM	PS	ZO	YES	PM	PS	PS	YES	PM	PS	PM	YES	PM	PS	PB	NO
PB	PS	ZO	YES	PB	PS	PS	YES	PB	PS	PM	YES	PB	PS	PB	YES
ZO	PM	ZO	YES	ZO	PM	PS	YES	ZO	PM	PM	NO	ZO	PM	PB	NO
PS	PM	ZO	YES	PS	PM	PS	YES	PS	PM	PM	YES	PS	PM	PB	NO
PM	PM	ZO	YES	PM	PM	PS	YES	PM	PM	PM	YES	PM	PM	PB	YES
PB	PM	ZO	YES	PB	PM	PS	YES	PB	PM	PM	YES	PB	PM	PB	YES
ZO	PB	ZO	YES	ZO	PB	PS	YES	ZO	PB	PM	YES	ZO	PB	PB	NO
PS	PB	ZO	YES	PS	PB	PS	YES	PS	PB	PM	YES	PS	PB	PB	YES
PM	PB	ZO	YES	PM	PB	PS	YES	PM	PB	PM	YES	PM	PB	PB	YES
PB	PB	ZO	YES	PB	PB	PS	YES	PB	PB	PM	YES	PB	PB	PB	YES

Based on the above arguments, we construct a rule base as shown in Table 5.5. We choose the following parameters as fuzzy inputs: the difference s between the expected holding costs in both queues, $s \in (-\infty, \infty)$, and the customer arrival rates λ_i

of class i, $\lambda_i \in [0, \mu_i)$, $i = 1, 2$. Each input is assigned four linguistic values, and the rule base consists of $4^3 = 64$ rules.

The membership functions for the fuzzy inputs s and λ_i and the fuzzy output d are shown in Figures 5.10a–c. The universe of discourse for the fuzzy output d is [0, 1]. As $\lambda_i < \mu_i$, we use a scaling factor $6/\mu_i$ to normalize λ_i over the standard domain [0, 6].

The universe of discourse for the fuzzy input s is $[0, \infty)$. Its scaling factor is determined using the rule base in Table 5.5. Consider Rule 2, *if s is PS and λ_1 is ZO and λ_2 is ZO, then d is YES*. From Figure 5.10a, we see that PS with membership grade 1.0 corresponds to a normalized value 2.0 for s. Also, in Case (1) we have shown that if $s \geq C$ then $d = 1$ is optimal. According to Figure 5.10c, the decision $d = 1$ corresponds to the linguistic value YES with membership grade 1.0. Hence, a value C for s is scaled to 2.0 and the scaling factor for s is $2/C$.

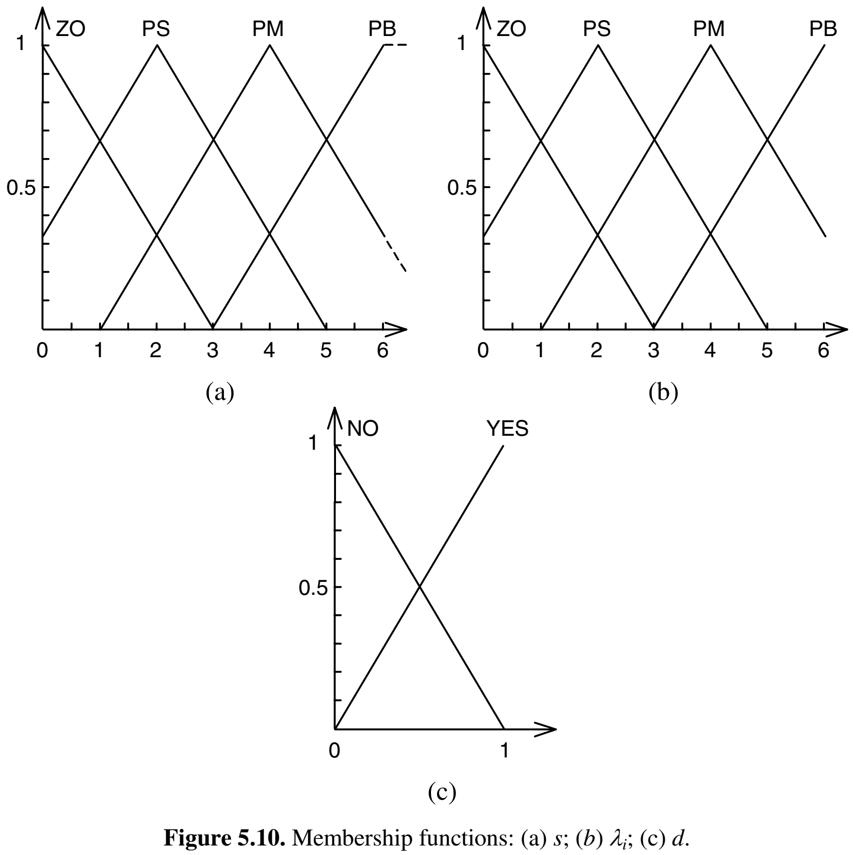


Figure 5.10. Membership functions: (a) s; (*b*) λ_i; (c) d.

5.5.3 A Numerical Example

We examine the system shown in Figure 5.9 with the parameters $\lambda_1 = 0.8$, $\lambda_2 = 0.5$, $\mu_1 = \mu_2 = 1$, $C = 5$, $h_1 = 2$, and $h_2 = 1$.

The optimal policy for d can be determined from the fuzzy logic controller as follows:

1. According to the given information, we determine the scaling factors for the fuzzy inputs s, λ_1, and λ_2. As $C = 5$, the scaling factor for s is $2/C = 0.4$, and from Figure 5.10a, PS for s with membership grade 1.0 is 2 in the universe of discourse. The scaling factor for λ_i is $6/\mu_i = 6$. According to Figure 5.10b, $\lambda_1 = 0.8$ corresponds to PS with grade 0.06, PM with grade 0.73, and PB with grade 0.6. Finally, $\lambda_2 = 0.5$ corresponds to PS with grade 0.67 and PM with grade 0.67.
2. We arbitrarily define two numbers M and N as the limits for s_1 and s_2, respectively, where we stop the search. If these limits are too small, we run the risk of missing the switching points, whereas if they are too large, we waste computational time. The proper size of these quantities is a matter of experience and experimentation.
3. We start the algorithm from an initial state $s_1 = s_2 = 0$.
4. According to the current s_1 and s_2, the fuzzy input s is first calculated.
5. Using the current s as well as the given λ_1 and λ_2 as crisp inputs, we determine the decision d via fuzzification, fuzzy inference, and defuzzification.
6. We plot the decision d in the two dimensional plane of s_1 and s_2,
7. If the current $s_2 = M$, we go to step (8); otherwise we let $s_2 = s_2 + 1$ and go to (4).
8. If the current $s_1 = N$, then we stop; otherwise we let $s_1 = s_1 + 1$ and go to (4).

For example, let us assume that the current numbers of queuing customers of class 1 and 2 are $s_1 = 3$ and $s_2 = 2$. Then,

$$s = h_1 \frac{s_1}{\mu_1} - h_2 \frac{s_2}{\mu_2} = 2.0 \times \frac{3}{1.0} - 1.0 \times \frac{2}{1.0} = 4,$$

which is scaled down to $4 \times 0.4 = 1.6$. We see from Figure 5.10a that s corresponds to ZO with grade 0.47, PS with grade 0.87, and PM with grade 0.2. According to the fuzzy rule base of Table 5.5, the fuzzy decisions d are formulated as follows:

If s is ZO with grade 0.46 and λ_1 is PS with grade 0.06 and λ_2 is PS with grade 0.67, then d is 0 with grade 0.06.

If s is PS with grade 0.86 and λ_1 is PS with grade 0.06 and λ_2 is PS with grade 0.67, then d is 1 with grade 0.06.

If s is PM with grade 0.2 and λ_1 is PS with grade 0.06 and λ_2 is PS with grade 0.67, then d is 1 with grade 0.06.

If s is ZO with grade 0.46 and λ_1 is PM with grade 0.73 and λ_2 is PS with grade 0.67, then d is 1 with grade 0.46.

$\vdots$

If s is PM with grade 0.2 and λ_1 is PM with grade 0.73 and λ_2 is PM with grade 0.67, then d is 1 with grade 0.2.
If s is ZO with grade 0.46 and λ_1 is PB with grade 0.6 and λ_2 is PM with grade 0.67, then d is 1 with grade 0.46.
If s is PS with grade 0.86 and λ_1 is PB with grade 0.6 and λ_2 is PM with grade 0.67, then d is 1 with grade 0.6.
If s is PM with grade 0.2 and λ_1 is PB with grade 0.6 and λ_2 is PM with grade 0.67, then d is 1 with grade 0.2.

Each of the two inputs s and λ_1 has three fuzzy sets, whereas λ_2 has two fuzzy sets; hence, $3 \times 3 \times 2 = 18$ fuzzy decisions are fired. The peak value of the decision $d = 1$ is 1.0 and that of $d = 0$ is 0.0. By the height method of defuzzification, the crisp output d^* is given by

$$d^* = \frac{\sum_{i=1}^{18} e_i f_i}{\sum_{i=1}^{18} f_i} = 0.86,$$

where i is the ith firing rule and e_i and f_i the peak value and membership grade of the corresponding fuzzy decision. As $d^* > 0.5$, the crisp decision is $d = 1$, which means that the arriving class 1 customer should be transferred to queue 2.

Repeating the above procedure for all possible pairs (s_1, s_2), we obtain the fuzzy control policy shown in Figure 5.11. This policy has a switchover structure in the two-dimensional state space of s_1 and s_2.

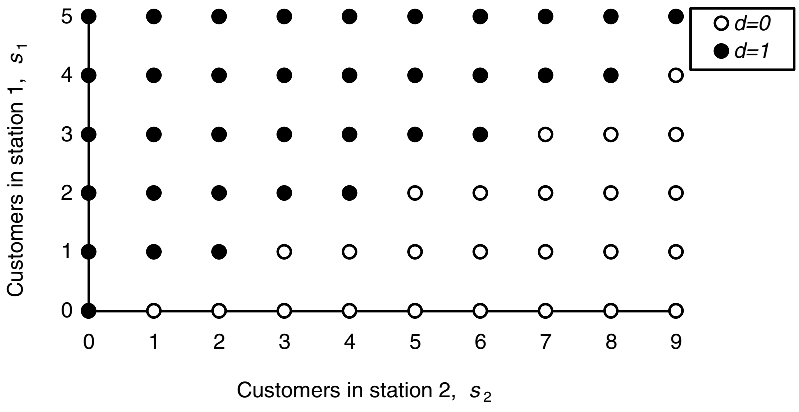


Figure 5.11. Switchover policy of the fuzzy controller. (J Intell Fuzzy Syst, Vol. 11, p. 168, by Runtong Zhang and Yannis A. Phillis. © 2001 by IOS Press. Used with permission.)

The existence of assignment costs when class 1 customers are routed to queue 2 is a negative factor for such decisions. However, we see from Figure 5.11 that an arriving class 1 customer can be transferred to queue 2 even when the length of queue 1 is smaller than the length of queue 2. In this example, this is because the holding cost per customer per unit time in queue 1 is greater than that in queue 2 and the arrival rate of class 1 customers is greater than the corresponding rate of class 2 customers.

5.6 Parallel Servers with Two Uncontrolled Arrival Streams

5.6.1 Problem Description

We now proceed with a queuing system similar to the one studied in the previous section but with three classes of customers, as shown in Figure 5.12. Customers of class j, $j = 1, 2$, join queue j waiting for service by server j. The length of queue j is observable. Class 3 customers can be served by either server and therefore, upon arrival, decisions must be made for scheduling.

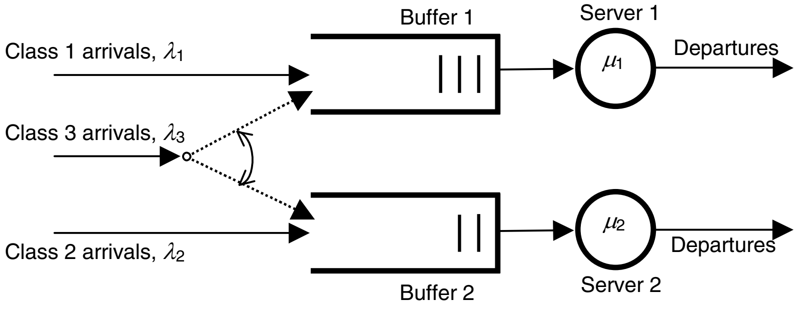

Buffer 1
Server 1
Class 1 arrivals, λ_1
μ_1
Departures
Class 3 arrivals, λ_3
μ_2
Departures
Class 2 arrivals, λ_2
Buffer 2
Server 2

Figure 5.12. Queuing system with two uncontrolled arrival streams.

As previously, we assume that the interarrival and service times are independent exponentially distributed random variables with corresponding rates as shown in Figure 5.12. A holding cost h_j is paid per customer per unit time in queue j and in server j. There are no costs for assigning class 3 customers to either queue. To ensure stability, it is assumed that $\lambda_j < \mu_j$, $j = 1, 2$, and $\lambda_1 + \lambda_2 + \lambda_3 < \mu_1 + \mu_2$.

We wish to decide the optimal scheduling policy, based on the state of the system, which minimizes the overall average holding cost.

5.6.2 Fuzzy Controller

The state of the system at the decision epochs can be described by the number of customers in queue j, $j = 1, 2$, $s_j = 0, 1, 2, \ldots$

As there are no assignment costs, a new class 3 customer that joins queue j incurs an expected holding cost $h_j(s_j + 1)/\mu_j$. From the viewpoint of this individual customer, it is optimal to join the queue with the smallest expected holding cost. Furthermore, if there are no additional arrivals of customers seeking service, the individually optimal behavior is also socially optimal in the sense that it minimizes the overall holding cost of the system. When additional arrivals are anticipated, however, the individually optimal behavior generally does not lead to an optimal policy for the whole system, because of the decrease in utility imposed on future customers. For example, if an arriving class 3 customer chooses queue 1, other customers of class 1 will have to wait longer and a higher holding cost for the whole system might be incurred. The arrival rates of all three classes play a role in bridging individual and social optimization criteria.

Let d be the decision to transfer an arriving class 3 customer to queue d, $d \in \{0, 1, 2\}$, where the value 0 for the decision d indicates indifference and 1 or 2 are chosen randomly. The establishment of the fuzzy rule base relies on similar arguments as in Section 5.5.2.

Case (1). Suppose $\lambda_i = 0$ for all $i = 1, 2, 3$. We want to schedule a class 3 customer in a system with s_j customers already assigned to queue j, $j = 1, 2$. From the previous discussion, the socially optimal decision is also individually optimal, thus

$$d = \begin{cases} 0 & \text{if } s = 0, \\ 1 & \text{if } s < 0, \\ 2 & \text{if } s > 0, \end{cases}$$

where

$$s \triangleq h_1 \frac{s_1 + 1}{\mu_1} - h_2 \frac{s_2 + 1}{\mu_2} .$$

Hence, the smaller the difference s between the customer's expected holding costs in both queues, the easier it is to decide $d = 1$ for a class 3 customer.

Case (2). A higher arrival rate of class 2 customers indicates that the length of queue 2 and the corresponding holding cost increase fast. In this case, the manager of the system would prefer to assign a class 3 customer to queue 1; hence, $d = 1$. In a dual fashion, a higher arrival rate of class 1 customers strengthens the decision $d = 2$.

Case (3). When s is negative (positive) and λ_2 (λ_1) is relatively high, the previous arguments strengthen the decision $d = 1(2)$. Otherwise, we have a conflict between individual and social criteria and the decision should be based on their relative strength. A higher arrival rate of class 3 customers tips the balance in favor of the individually optimal criterion. Thus, for example, if s is negative but close to zero and λ_1 is a little larger than λ_2, then we have two conflicting situations of equal strength. Thus, $d = 0$. However, if λ_3 is high, then $d = 1$.

The inputs of the fuzzy controller are the difference s between the expected holding costs in both queues, $s \in (-\infty, \infty)$, and the customer arrival rates λ_i, $i = 1, 2, 3$. We use three linguistic values for the arrival rates and five linguistic values for s. Based on the above arguments, we construct the rule base shown in Table 5.6, which consists of $5 \times 3^3 = 135$ rules.

The membership functions for λ_i, d, and s are shown in Figure 5.13. The physical domain of λ_j, $j = 1, 2$, is $[0, \mu_j)$ and for λ_3 it is $[0, \mu_1 + \mu_2 - \lambda_1 - \lambda_2)$. The universe of discourse for λ_i is $[0, 4)$, $i = 1, 2, 3$. The universe of discourse for the fuzzy output d is $[0, 1]$.

Table 5.6. Rule base.

Rules 1–45					*Rules 46–90*					*Rules 91–135*				
s	λ_1	λ_2	λ_3	d	s	λ_1	λ_2	λ_3	d	s	λ_1	λ_2	λ_3	d
NB	ZO	ZO	ZO	1	NB	ZO	ZO	PS	1	NB	ZO	ZO	PB	1
NS	ZO	ZO	ZO	1	NS	ZO	ZO	PS	1	NS	ZO	ZO	PB	1
ZO	ZO	ZO	ZO	0	ZO	ZO	ZO	PS	0	ZO	ZO	ZO	PB	0
PS	ZO	ZO	ZO	2	PS	ZO	ZO	PS	2	PS	ZO	ZO	PB	2
PB	ZO	ZO	ZO	2	PB	ZO	ZO	PS	2	PB	ZO	ZO	PB	2
NB	PS	ZO	ZO	1	NB	PS	ZO	PS	1	NB	PS	ZO	PB	1
NS	PS	ZO	ZO	0	NS	PS	ZO	PS	1	NS	PS	ZO	PB	1
ZO	PS	ZO	ZO	2	ZO	PS	ZO	PS	2	ZO	PS	ZO	PB	2
PS	PS	ZO	ZO	2	PS	PS	ZO	PS	2	PS	PS	ZO	PB	2
PB	PS	ZO	ZO	2	PB	PS	ZO	PS	2	PB	PS	ZO	PB	2
NB	PB	ZO	ZO	0	NB	PB	ZO	PS	1	NB	PB	ZO	PB	1
NS	PB	ZO	ZO	2	NS	PB	ZO	PS	2	NS	PB	ZO	PB	2
ZO	PB	ZO	ZO	2	ZO	PB	ZO	PS	2	ZO	PB	ZO	PB	2
PS	PB	ZO	ZO	2	PS	PB	ZO	PS	2	PS	PB	ZO	PB	2
PB	PB	ZO	ZO	2	PB	PB	ZO	PS	2	PB	PB	ZO	PB	2
NB	ZO	PS	ZO	1	NB	ZO	PS	PS	1	NB	ZO	PS	PB	1
NS	ZO	PS	ZO	1	NS	ZO	PS	PS	1	NS	ZO	PS	PB	1
ZO	ZO	PS	ZO	1	ZO	ZO	PS	PS	1	ZO	ZO	PS	PB	1
PS	ZO	PS	ZO	0	PS	ZO	PS	PS	2	PS	ZO	PS	PB	2
PB	ZO	PS	ZO	2	PB	ZO	PS	PS	2	PB	ZO	PS	PB	2
NB	PS	PS	ZO	1	NB	PS	PS	PS	1	NB	PS	PS	PB	1
NS	PS	PS	ZO	1	NS	PS	PS	PS	1	NS	PS	PS	PB	1
ZO	PS	PS	ZO	0	ZO	PS	PS	PS	0	ZO	PS	PS	PB	0
PS	PS	PS	ZO	2	PS	PS	PS	PS	2	PS	PS	PS	PB	2
PB	PS	PS	ZO	2	PB	PS	PS	PS	2	PB	PS	PS	PB	2
NB	PB	PS	ZO	2	NB	PB	PS	PS	2	NB	PB	PS	PB	2
NS	PB	PS	ZO	2	NS	PB	PS	PS	2	NS	PB	PS	PB	2
ZO	PB	PS	ZO	2	ZO	PB	PS	PS	2	ZO	PB	PS	PB	2
PS	PB	PS	ZO	0	PS	PB	PS	PS	2	PS	PB	PS	PB	2
PB	PB	PS	ZO	1	PB	PB	PS	PS	2	PB	PB	PS	PB	2
NB	ZO	PB	ZO	1	NB	ZO	PB	PS	1	NB	ZO	PB	PB	1
NS	ZO	PB	ZO	1	NS	ZO	PB	PS	1	NS	ZO	PB	PB	1
ZO	ZO	PB	ZO	1	ZO	ZO	PB	PS	1	ZO	ZO	PB	PB	1
PS	ZO	PB	ZO	1	PS	ZO	PB	PS	1	PS	ZO	PB	PB	1
PB	ZO	PB	ZO	0	PB	ZO	PB	PS	2	PB	ZO	PB	PB	2
NB	PS	PB	ZO	1	NB	PS	PB	PS	1	NB	PS	PB	PB	1
NS	PS	PB	ZO	1	NS	PS	PB	PS	1	NS	PS	PB	PB	1
ZO	PS	PB	ZO	1	ZO	PS	PB	PS	1	ZO	PS	PB	PB	1
PS	PS	PB	ZO	0	PS	PS	PB	PS	2	PS	PS	PB	PB	2
PB	PS	PB	ZO	2	PB	PS	PB	PS	2	PB	PS	PB	PB	2
NB	PB	PB	ZO	1	NB	PB	PB	PS	1	NB	PB	PB	PB	1
NS	PB	PB	ZO	1	NS	PB	PB	PS	1	NS	PB	PB	PB	1
ZO	PB	PB	ZO	0	ZO	PB	PB	PS	0	ZO	PB	PB	PB	0
PS	PB	PB	ZO	2	PS	PB	PB	PS	2	PS	PB	PB	PB	2
PB	PB	PB	ZO	2	PB	PB	PB	PS	2	PB	PB	PB	PB	2

The physical domain as well as the universe of discourse for the input variable s are both $(-\infty, \infty)$. The corresponding scaling factor is determined with the aid of the rule base. If a class 3 customer arrives and no additional customers of any classes are expected, i.e., $\lambda_1 = \lambda_2 = \lambda_3 = 0$, then, according to Rule 4 of Table 5.6, the customer will join queue 2 if s is PS. The problem then is to find a threshold value for s that is positive enough to tip the balance in favor of $d = 2$. As s is decreasing in s_2, this threshold value should also be positive enough to keep s nonnegative even when s_2 is increased by one; in which case, s is reduced by h_2/μ_2. Hence, PS for the input s with membership grade 1.0 is fixed at h_2/μ_2 with normalized value 2.0. Similarly, from Rule 2 of Table 5.6, we fix NS for s with grade 1.0 at $-h_1/\mu_1$ with normalized value −2.0.

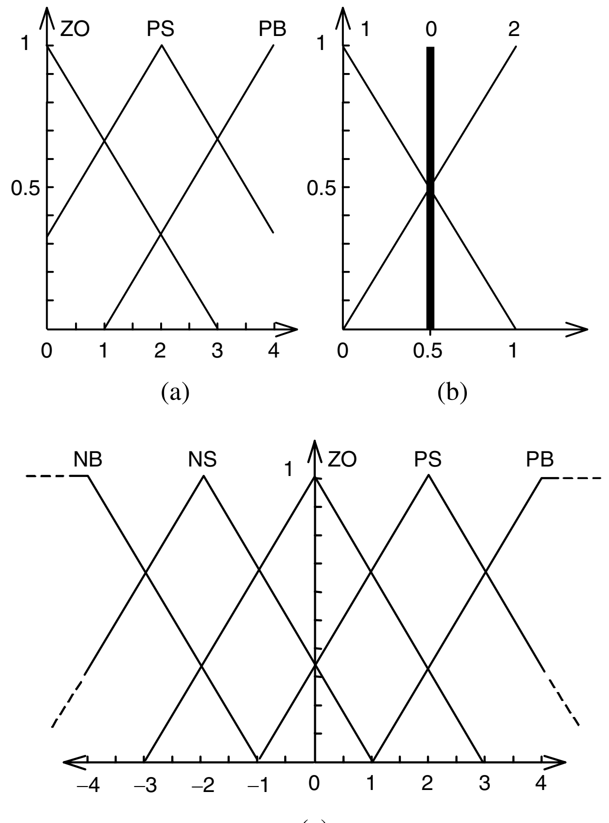


Figure 5.13. Membership functions: (a) λ_i; (b) d; (c) s.

5.6.3 A Numerical Example

We examine the system shown in Figure 5.12 with parameters $\lambda_1 = 0.5$, $\lambda_1 = 0.3$, $\lambda_1 = 0.6$, $\mu_1 = \mu_2 = 1$, $h_1 = 2$, and $h_2 = 1$. Following the same algorithm as in Section 5.5.3, we obtain the fuzzy control policy shown in Figure 5.14, which is of a switchover structure in the two-dimensional state space of s_1 and s_2.

We see from Figure 5.14 that an arriving class 3 customer is often scheduled to queue 2, although the length of queue 1 is shorter than that of queue 2. This is because the holding cost per customer per unit time in queue 1 is greater than that in queue 2 and the arrival rate of class 1 customers is also greater than that of class 2 customers.

Under this control policy, the average holding cost is 6.83 per time unit.

No analytical solution is known for this problem. For the special case of equal service rates and equal holding cost rates, Weber (1978) and Winston (1977) show that the policy of joining the shortest queue minimizes the average cost.

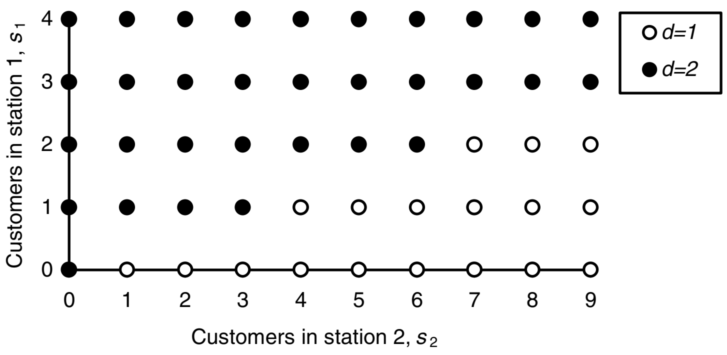


Figure 5.14. Switchover policy of the fuzzy controller. (IEEE T Fuzzy Syst, Vol. 9, p. 312, by Runtong Zhang and Yannis A. Phillis. © 2001 by IEEE. Used with permission.)

To study the efficiency of the method, we examine three control policies:

- *Fuzzy Control*
- *Shortest Queue*, where the customer is sent to the server with the shortest queue
- *Round Robin*, where the first customer is sent to queue 1, the next one to queue 2, and so on

We simulate the system for each control policy separately under varying class 3 arrival rates. The average cost versus λ_3 for each policy is shown in Figure 5.15. We see that the fuzzy controller performs better than the round robin and the shortest queue policies, especially when the load becomes heavy.

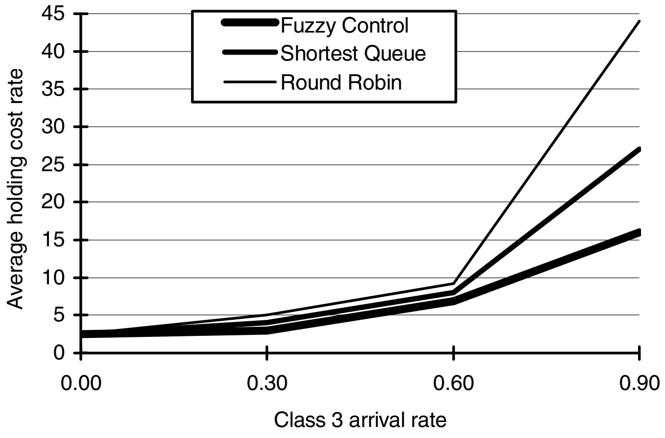


Figure 5.15. A comparison of policies under varying load conditions. (IEEE T Fuzzy Syst, Vol. 9, p. 312, by Runtong Zhang and Yannis A. Phillis. © 2001 by IEEE. Used with permission.)

6 Control of the Admission of Customers

6.1 Introduction

In this chapter, we examine queuing systems in which arriving customers may be accepted or rejected. The objective is to determine admission policies that maximize the net profit (reward minus cost) over an infinite horizon. Four problems are studied in detail:

- A single server with one class of customers
- Parallel servers with one class of customers
- Parallel servers with two classes of customers
- Serial queuing networks with two classes of customers

Only the first problem of the above list has an analytic solution.

To determine admission policies, one may use individual or social optimality criteria. Individual optimality is often easier to be handled mathematically than social optimality. Naturally, these two policies do not necessarily coincide as the first stresses optimality of the point of view of each customer, and the second views the system as a whole entity. One may extend the discussion to optimality in the context of social or economic systems where human values play a decisive role. In this chapter, we shall use social optimality in the framework of fuzzy logic. Once again, the caveats about optimality raised in Section 1.5 hold. However, as in Chapters 4 and 5, whenever analytical solutions exist, they coincide with the fuzzy ones. This is as good an indication as we can get about the validity of the fuzzy controllers.

6.2 Single Server with One Arrival Stream

6.2.1 Problem Description

We consider an M/M/1 queuing system with arrival control, as shown in Figure 6.1. Customers arrive into the system according to a Poisson process with mean rate λ, and they are served by one exponential server with mean rate μ. The system receives a fixed reward w for each accepted customer and incurs a holding cost h per customer per unit time in the system. A controller, located at the entrance of the system, has the options of permitting or denying admission to each arriving customer depending on the state of the system. The controller's task is to decide the admission policy that maximizes the average profit rate (reward minus holding cost). Because the customer arrivals are controlled, it is *not* necessary to assume that $\lambda < \mu$ in order to guarantee stability.

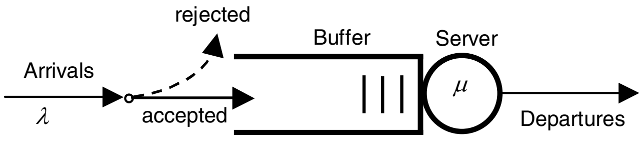


Figure 6.1. A system with one server and one class of customers.

This process is a semi-Markov decision process. Naor (1969) has proved that the individually and socially optimal policies are of the threshold type. Specifically, the system admits a customer if and only if the number x of customers in the system is less than a threshold value. The individually optimal threshold x_i is given by

$$x_i \le \frac{w\mu}{h} < x_i + 1, \tag{6.1}$$

whereas the socially optimal threshold x_s is given by

$$\frac{x_s(1-\rho)-\rho(1-\rho^{x_s})}{(1-\rho)^2} \le \frac{w\mu}{h} < \frac{(x_s+1)(1-\rho)-\rho(1-\rho^{x_s+1})}{(1-\rho)^2}, \tag{6.2}$$

where $\rho = \lambda/\mu$. It turns out that $x_s \le x_i$, which means that a socially optimal policy admits fewer customers than an individually optimal policy. We shall use these formulas to compare our results with the analytical ones.

To avoid the trivial situation where an arriving customer is immediately denied entrance even when the system is empty ($x_i = x_s = 0$), we assume that $w > h/\mu$. Hence, both x_s and x_i are greater than zero and an arriving customer who finds the server idle ($x = 0$) is always accepted. Henceforth we will only focus on the optimal admission of customers when the server is busy or $x \ge 1$.

6.2.2 Fuzzy Controller

The decision epochs as always coincide with the customer arrival times in which the server is busy. The state of the system is described by x, $x = 1, 2, \ldots$, or, equivalently, by $s = x - 1$, where s is the number of customers in queue, $s = 0, 1, \ldots$.

The number of customers in queue has a negative effect on the decision to admit a customer, and together with the arrival rate λ determines the fuzzy control policy. The output of the fuzzy controller is the decision $d = 1, 0$ of admitting an arriving customer into the system.

Table 6.1 shows a fuzzy rule base that relies on the following arguments.

1. As s increases, the holding cost becomes greater than the reward, and therefore, admission should be denied.

2. A higher arrival rate of customers implies that the server is rarely starved. As a result, the cost of holding customers in the system increases, and this strengthens the decision $d = 0$.

Table 6.1. Rules whose output is YES.

s	λ	d	s	λ	d
ZO	ZO	YES	PS	ZO	YES
ZO	PS	YES	PS	PS	YES
ZO	PM	YES	PM	ZO	YES

For brevity, Table 6.1 records only the rules whose output is YES. All other combinations of s and λ lead to the decision NO.

The universes of discourse for the fuzzy inputs λ and s are both $[0, \infty)$. The universe of discourse for the fuzzy output d is $[0, 1]$. The corresponding membership functions are shown in Figure 6.2.

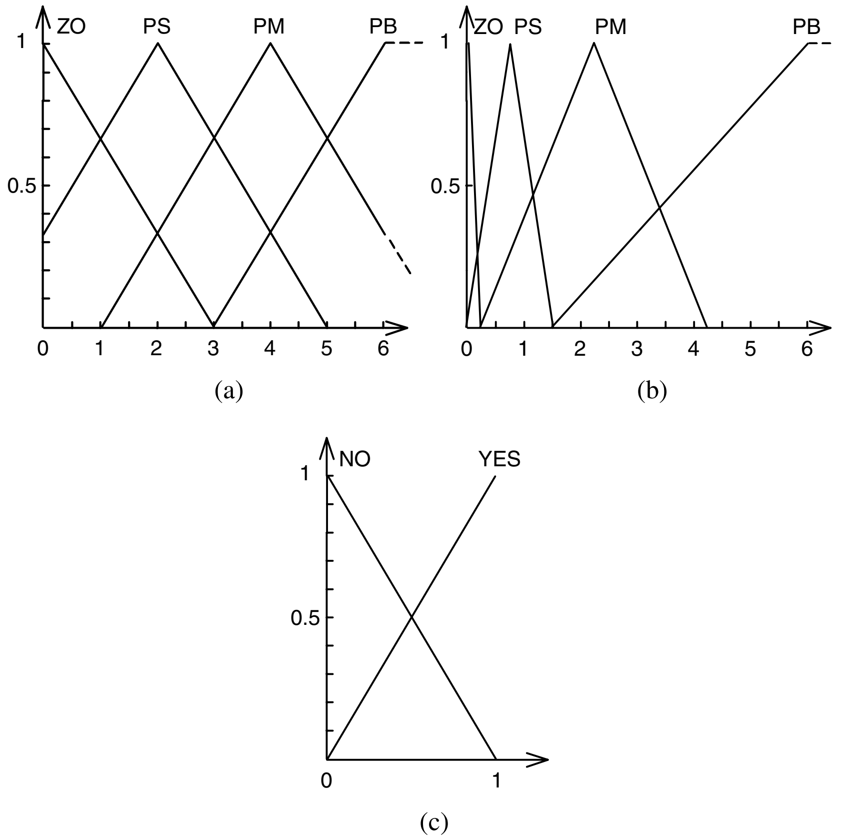


Figure 6.2. Membership functions: (a) λ; (b) s; (c) d.

The membership functions of the input variable λ are evenly spaced, as shown in Figure 6.2a. After solving a few test problems, we discovered that the membership functions for the number of customers in queue should be denser for small values of s, as shown in Figure 6.2b. In addition, we have noticed that the decision of admitting a customer is beneficial whenever one fuzzy output is YES. In other words, a decision “No” is made only when all the fuzzy outputs are NO, whereby we obtain the fuzzy membership functions for d as in Figure 6.2c.

We now determine the quantitative relationships among the fuzzy output d and the two fuzzy inputs s and λ. For the special case $\lambda \approx 0$, there is a minimum value $x_{\lambda \approx 0}$ for the number x of customers in the system beyond which we should reject all new customers requesting admission. This value is such that the reward from the admission of a new customer, $(1 + x_{\lambda \approx 0})$, is less than the corresponding expected holding cost. Hence,

$$x_{\lambda \approx 0} = \min x \colon w < h\frac{x+1}{\mu}$$

It turns out that $x_{\lambda \approx 0}$ is the individually optimal threshold $x_i = \lfloor w\mu/h \rfloor$, which is computed from inequalities (6.1). This special situation gives rise to the rule: *If λ is ZO and s is PB, then d is NO*. The value $w\mu/h$, therefore, corresponds to the linguistic value PB of s with membership grade 1.0.

Regarding the fuzzy input λ, we consider the fuzzy rule *if λ is PB and s is ZO, then d is NO*. A threshold value $\lambda_{s=0}$ for λ should be such that, for any arrival rate $\lambda \ge \lambda_{s=0}$, it is more profitable to reject incoming customers when the server is busy, i.e., $s = 0$ and $x = 1$. We have then reduced the original system to an M/M/1/1 queuing system. Let $J(K)$ denote the mean profit rate of an M/M/1/K system in which all arrivals that find K customers already in the system are rejected. Then, rejecting all arrivals when the server is busy is optimal if the corresponding profit $J(1)$ is greater than or equal to $J(2)$, which is the mean profit rate when one more customer is allowed in the system. From standard queuing formulas (Kleinrock 1975), we obtain

$$J(K) = w\times(\text{throughput}) - h\times(\text{mean number in the system})$$

$$= w\mu\left(1 - \frac{1-\rho}{1-\rho^{K+1}}\right) - h\frac{\rho}{1-\rho}\frac{1-(K+1)\rho^K + K\rho^{K+1}}{1-\rho^{K+1}},$$

where $\rho \triangleq \lambda/\mu$. From the above, the condition $J(1) \ge J(2)$ after a little algebra yields

$$1 + \rho \ge \frac{w\mu - h}{h}$$

or, equivalently,

$$\lambda \ge \lambda_{s=0} \triangleq \frac{\mu(w\mu - 2h)}{h}. \tag{6.3}$$

Hence, PB for λ with membership grade 1.0 is fixed at $\lambda_{s=0}$.

6.2.3 A Numerical Example

An M/M/1 system with admission control has parameters $\lambda = 0.3$, $\mu = 0.3$, $w = 50$, and $h = 2$.

From the previous discussion and Figure 6.2, the scaling factors for the fuzzy inputs λ and s are $6/\lambda_{s=0} = 3.64$ and $6/(w\mu/h) = 0.8$, respectively. Then $\lambda = 0.3$ is scaled down to $0.3 \times 3.64 = 1.1$ which is interpreted as ZO with grade 0.63, PS with grade 0.70, or PM with grade 0.03.

The fuzzy inference procedure is briefly illustrated as follows. At each decision epoch, the fuzzy logic controller fuzzifies the current number of queuing customers s into suitable linguistic values. Based on the fuzzy rules, corresponding fuzzy decisions are fired. A decision "No" is made only if all the fuzzy outputs are NO. For example, let us assume that the current number of queuing customers is $s = 1$. This value is scaled down to $1 \times 0.8 = 0.8$, which from Figure 6.2b corresponds to PS with grade 0.91 or PM with grade 0.27. According to the fuzzy rule base (Table 6.1) and Mamdani implication, the fuzzy decisions d are formulated as follows:

If λ is ZO with grade 0.63 and s is PS with grade 0.91, then d is YES with grade 0.63.
If λ is PS with grade 0.70 and s is PS with grade 0.91, then d is YES with grade 0.70.
If λ is PM with grade 0.03 and s is PS with grade 0.91, then d is YES with grade 0.03.
If λ is ZO with grade 0.63 and s is PM with grade 0.27, then d is YES with grade 0.27.
If λ is PS with grade 0.70 and s is PM with grade 0.27, then d is NO with grade 0.27.
If λ is PM with grade 0.03 and s is PM with grade 0.27, then d is NO with grade 0.03.

From Figure 6.2c, the peak values and heights of the fuzzy decisions d are $e_1 = e_2 = e_3 = e_4 = 1$, $e_5 = e_6 = 0$, and $f_1 = 0.63$, $f_2 = 0.70$, $f_3 = 0.03$, $f_4 = 0.27$, $f_5 = 0.27$, $f_6 = 0.03$. By the height method of defuzzification, the crisp output d^* is given by

$$d^* = \frac{\sum_{i=1}^{6} e_i f_i}{\sum_{i=1}^{6} f_i} = 0.845.$$

As $d^* > 0.5$, the decision is "Yes."

The simulation in the fuzzy logic control environment is started from an initial state $x = 0$, and the system performance for the first 500 time units is shown in Figure 6.3.

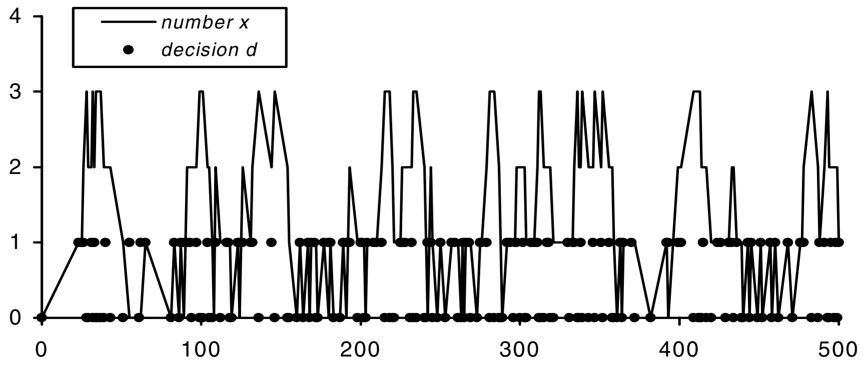


Figure 6.3. Evolution of x and d under the fuzzy control policy.

Note that the number x of customers in the system is never greater than 3. When x is less than 3, the controller always admits an arriving customer to the system ($d = 1$); otherwise a decision of rejecting an arriving customer ($d = 0$) is made. This is a threshold policy with threshold $x_s = 3$, which is the same as that found by formula (6.2).

6.3 Parallel Servers with One Arrival Stream

6.3.1 Problem Description

Now we add parallel identical servers to the previous system and obtain an M/M/m queuing system with controlled arrivals, as shown in Figure 6.4. Again, we wish to decide the optimal admission policy so that the average profit is maximized.

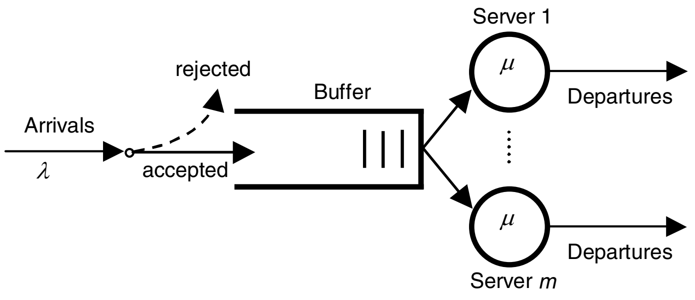


Figure 6.4. A parallel-server queue with arrival control.

Knudsen (1972) and Yechiali (1972) have extended Naor's results by establishing the existence of an optimal policy of the threshold type. Furthermore, it is suggested in Knudsen (1972) that this result is valid for GI/M/m systems, whereas Yechiali (1972) uses a variable reward and a nonlinear customer holding cost. However, no explicit calculation of the threshold has been provided up to now.

6.3.2 Fuzzy Controller

As in the previous model, the decision epochs are restricted to the arrival times of new customers. The state of the system is represented by the total number x of customers in queue and in service. To avoid the trivial situation where an arriving customer is immediately denied entrance even when there are idle servers, we assume that $w > h/\mu$. Hence, if there are idle servers in the system, then an arriving customer is admitted.

Henceforth we shall be concerned with the optimal admission of customers when all m servers are busy, or $x \geq m$. Then, the control policy is governed by the number of customers in the buffer, $s = x - m$.

The development of the fuzzy controller relies on previous arguments if we view the system with m parallel servers as a single-server system with rate $m\mu$. The fuzzy input s has the value PB for membership grade 1.0 at $wm\mu/h$. Next we rewrite inequality (6.3) as

$$\lambda \geq \lambda_{s=0}^{(m)} \triangleq \frac{m\mu(wm\mu - 2h)}{h}. \tag{6.4}$$

Consequently, PB for λ with membership grade 1.0 is fixed at $\lambda_{s=0}^{(m)}$.

6.3.3 A Numerical Example

An M/M/m queuing system with controlled arrivals has parameters $m = 10$, $\lambda = 1$, $\mu = 0.1$, $w = 50$, and $h = 2$.

We determine the fuzzy control policy as follows:

1. We start from an initial state $x = m$ ($s = 0$).
2. Using the current number s of queuing customers and the given arrival rate λ as crisp inputs, we determine the decision via fuzzification, fuzzy inference and defuzzification.
3. If the decision is "No," then we stop and record the value of the current state x as the threshold of the fuzzy controller. Otherwise, we add one to the current state x and repeat the procedure from step (2).

Following this algorithm, we find that the threshold is $x^* = 17$. It turns out from this algorithm that the CPU time is constant with respect to the number of servers m and linear with respect to the threshold x^*. This is a significant improvement over the computational time of classical algorithms such as policy iteration where the CPU time increases as the mth power of 2 and the third of an upper bound to the socially optimal threshold $\overline{x}_s$. For the M/M/10 system of our example, the optimal threshold is $\overline{x}_s = 17$. Thus,

$$\frac{\text{CPU time of the fuzzy control approach}}{\text{CPU time of the policy iteration method}} \approx \frac{17}{17^3 \times 2^{10}},$$

which is an enormous computational saving and one of the great advantages of our approach. Conventional policy iteration chooses the best actions by eliminating nonoptimal ones, and fuzzy logic control does this by means of direct determination of the best action. The larger the system scale, the greater this advantage becomes.

6.4 Parallel Servers with Two Arrival Streams

6.4.1 Problem Description

We now assume that the previous system serves two classes of arriving customers as shown in Figure 6.5. Customers of class i, $i = 1, 2$, arrive in a Poisson manner with rate λ_i. Class 1 customers upon arrival are either permitted or prohibited to enter the buffer, and class 2 customers enter the buffer without restriction. As previously, the system receives a fixed reward w for each accepted customer and pays a cost h per customer per time unit in the system. We wish to decide the optimal admission policy so that the average profit is maximized.

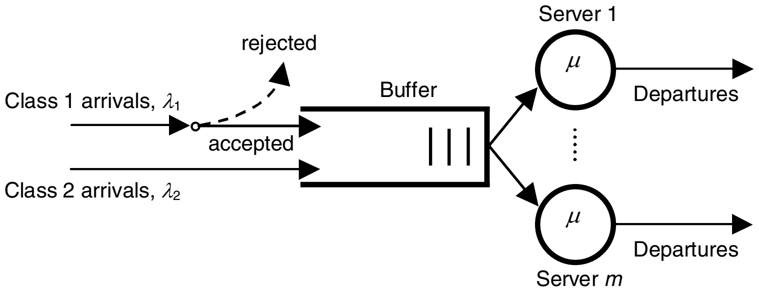


Figure 6.5. A parallel-server queue with two classes of customers.

Blanc *et al.* (1992), using dynamic programming, considered a similar system with more general reward functions but no holding costs and proved the existence of a threshold policy .

6.4.2 Fuzzy Controller

As previously, the decision epochs at which the arriving customers are controlled are restricted to the times of arrival of new class 1 customers. The state of the system at the decision epochs is described by the number of class 1 and class 2 customers in the system including the customers in service (if any), $x = 0, 1, 2, \ldots$

As only class 2 customer arrivals are uncontrolled, we assume that $\lambda_2 < m\mu$. Any admission policy ensuring a finite cost rate will also ensure boundedness of class 1 customers in the system regardless of whether the conditions $\lambda_1 < m\mu$ or $\lambda_1 + \lambda_2 < m\mu$ hold.

As previously, we assume that $w > h/\mu$, from which we write a crisp rule for the state $x < m$, *if there are idle servers in the system, then an arriving class 1 customer* *is admitted*. We are now interested in the optimal admission of class 1 customers when all m servers are busy or $x \geq m$.

We choose as fuzzy inputs the number of customers in the buffer $s = 0, 1, 2, \ldots$, and the arrival rates $\lambda_1 \in [0, \infty)$ and $\lambda_2 \in [0, m\mu)$. The fuzzy output is the decision $d = 1, 0$ as to whether an arriving class 1 customer is admitted.

The development of the fuzzy rule base relies on previous ideas. The larger the arrival rates or the number of customers in the queue, the more difficult it becomes to admit an arriving class 1 customer. The rule base is built as follows. To each linguistic value, we assign an integer weight, ZO→0, PS→1, PM→2, and PB→3. For each rule, we compute the sum of the weights associated with the corresponding linguistic values of its inputs. We decide *d is YES*, if the sum is less than or equal to 2; otherwise we decide *d is NO*. Thus, for example, the rule with antecedent part *if s is ZO and λ_1 is PS and λ_2 is PS* has a total weight $0 + 1 + 1 = 2$, and therefore, its consequent part is *d is YES*. For brevity, Table 6.2 records only the rules whose output is YES. All other combinations of s, λ_1, and λ_2 lead to the decision NO.

Table 6.2. Rules whose output is YES.

s	λ_1	λ_2	d	s	λ_1	λ_2	d
ZO	ZO	ZO	YES	ZO	PM	ZO	YES
ZO	ZO	PS	YES	PS	ZO	ZO	YES
ZO	ZO	PM	YES	PS	ZO	PS	YES
ZO	PS	ZO	YES	PS	PS	ZO	YES
ZO	PS	PS	YES	PM	ZO	ZO	YES

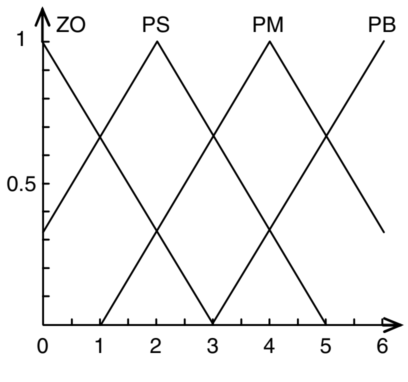


Figure 6.6. Membership functions of λ_2.

The membership functions for the fuzzy inputs λ_1, s, and λ_2 are shown in Figures 6.2a,b and 6.6, respectively, and for the fuzzy output d, they are shown in Figure 6.2c. The universe of discourse for s and λ_1 is $[0, \infty)$, for λ_2 it is $[0, 6)$, and for d it is $[0, 1]$.

When $\lambda_2 = 0$, the problem reduces to the previous one and λ_1 coincides with λ. Hence, PB for s with membership grade 1.0 corresponds to the value $wm\mu/h$ and PB for λ_1 with membership grade 1.0 is fixed at $\lambda_{s=0}^{(m)}$, which is computed from

Equation (6.4). Finally, as $\lambda_2 < m\mu$, a value $m\mu$ for λ_2 is declared PB with grade 1.0.

6.4.3 A Numerical Example

We apply the fuzzy controller to an M/M/2 queuing system serving two arrival streams. The system parameters are $\lambda_1 = 0.25$, $\lambda_2 = 0.05$, $\mu = 0.15$, $w = 50$, and $h = 2$.

The simulation of the system is started from an initial state $x = 0$, and the system performance for the first 500 time units is shown in Figure 6.7. From the system performance, we see that an arriving class 1 customer is admitted when $x < 3$, but if $x \geq 3$, the incoming customer is rejected. Hence, we obtain the threshold $x^* = 3$. This policy is not the same as in the single-server example of Section 6.2.3 because in the present case, an arriving class 1 customer is admitted only when the buffer is empty, whereas in that example an arriving customer is admitted even when there is already a customer in the buffer.

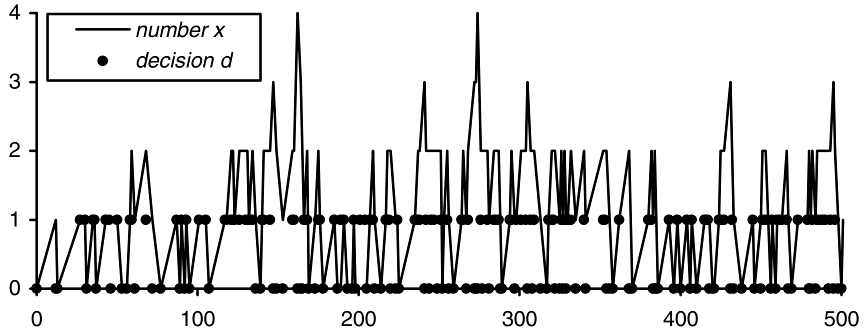


Figure 6.7. Evolution of x and d under the fuzzy control policy.

By modeling the system as a Markov chain, we compute the mean profit rate $J(x^*)$ under different threshold values x^*. The results are $J(2) = 5.64$, $J(3) = 2.38$, $J(4) = 0.09$, and $J(5) = -1.75$. We see that the fuzzy controller imposes a threshold that is very close to the optimal one.

6.5 Two Stations in Tandem with Their Own Arrival Streams

6.5.1 Problem Description

We now consider two queues in tandem, each of which has its own input of arriving customers that may either be accepted or rejected. This queuing network is depicted in Figure 6.8.

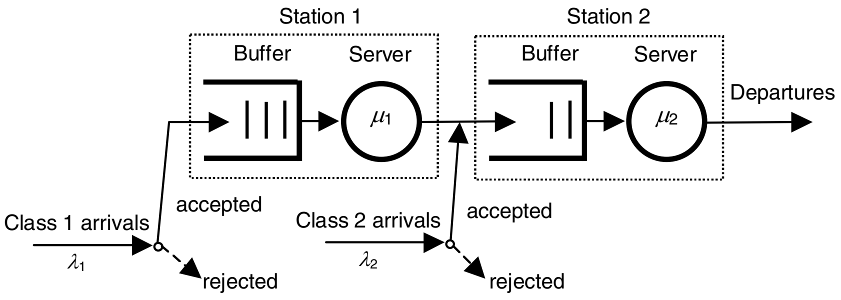


Figure 6.8. Tandem workstations with controlled arrivals.

Each station has an exponential server and an unlimited queuing capacity. Class 1 customers seek admission to station 1 and then go to station 2 after being served in station 1. Class 2 customers request admission to station 2 directly. All customers leave the system after being served in station 2. Class i arrivals occur in a Poisson stream with constant rate λ_i, $i = 1, 2$. Successive services in each station j are independent and exponentially distributed, with mean $1/\mu_j$ in station j, $j = 1, 2$, regardless of the class of customer. An arriving customer may either be permitted or prohibited to enter the system.

Suppose that the system receives a fixed reward w_i for each accepted customer of class i and pays a holding cost h_j per customer per time unit in station j. We wish to decide the optimal admission policy, based on the state of the system, so that the average profit (reward minus cost) is maximized. This is a semi-Markov decision process. Indeed, although the state transitions depend on the current state rather than on the history of the process, the times between two successive admissions are no longer exponential random variables because of the possibility of rejecting certain intermediate arrivals.

For this system, Ghoneim and Stidham (1985) have proved that the optimal policy is of the monotonic type, whereby more customers in the system strengthen the decision to reject an incoming customer.

6.5.2 Fuzzy Controller

Fuzzy Inputs and Fuzzy Rule Base

The state of the system is described by (x_1, x_2), where x_i is the number of customers in station i (queue plus service), $i = 1, 2$. To avoid the trivial situation where an arriving class 1 customer is rejected even when the system is empty ($x_1 = x_2 = 0$) or an arriving class 2 customer is rejected even when station 2 is empty ($x_2 = 0$), we assume that the reward at each station per customer is greater than the corresponding expected holding cost. Hence,

$$w_1 > \frac{h_1}{\mu_1} + \frac{h_2}{\mu_2} \tag{6.5}$$

and

$$w_2 > \frac{h_2}{\mu_2}. \tag{6.6}$$

Hereafter we shall only consider the decision to control a class 1 arrival when $x_1 + x_2 > 0$ or a class 2 arrival when $x_2 > 0$.

The system receives a reward for accepting customers and incurs a cost for holding customers. Under conditions (6.5) and (6.6), the best behavior for the system is to keep the server in station 2 busy, as long as customers are arriving, and never hold any customers in queue. However, this state cannot be maintained indefinitely because of the memoryless property of the exponential distribution whereby neither the interarrival nor the service times can be conditioned on the present observable state. If the system receives a high reward for accepting and incurs a low cost for holding class 1 customers, then the system should accept class 1 customers easily; otherwise it should not. The fuzzy logic controller specifies the control policies by taking into account such observations.

The fuzzy inputs are the numbers $s_i = 0, 1, \ldots, i = 1, 2$ of customers in the buffer of station i and the customer arrival rates $\lambda_j \in [0, \infty), j = 1, 2$ of class j. The decisions $d_j = 1, 0$ as to whether an arriving class j customer is admitted into the system are the fuzzy outputs. Both the number of queuing customers at each station and the customer arrival rates weaken the decision to accept an arriving customer regardless of class.

Based on the above argument, we develop a fuzzy rule base for d_j in Table 6.3, where YES means that an arriving customer of class j is admitted into station j. We choose four fuzzy sets for each of the four inputs, and the complete rule base consists of $4^4 = 256$ rules. In the table, we record only the rules whose output is YES for brevity. All other combinations of s_1, s_2, λ_1, and λ_2 lead to the decision NO.

The membership functions for the fuzzy inputs s_i, $i = 1, 2$, and λ_j, $j = 1, 2$, are shown in Figure 6.2a, and for the fuzzy outputs d_j they are shown in Figure 6.2c. The universes of discourse for the fuzzy inputs s_i and λ_j are all $[0, \infty)$, and for the fuzzy output d_j they are $[0, 1]$.

Table 6.3. Rules whose output is YES.

s_1	s_2	λ_1	λ_2	d_1 or d_2	s_1	s_2	λ_1	λ_2	d_1 or d_2
ZO	ZO	ZO	ZO	YES	ZO	PS	PS	ZO	YES
ZO	ZO	ZO	PS	YES	ZO	PM	ZO	ZO	YES
ZO	ZO	ZO	PM	YES	PS	ZO	ZO	ZO	YES
ZO	ZO	PS	ZO	YES	PS	ZO	ZO	PS	YES
ZO	ZO	PS	PS	YES	PS	ZO	PS	ZO	YES
ZO	ZO	PM	ZO	YES	PS	PS	ZO	ZO	YES
ZO	PS	ZO	ZO	YES	PM	ZO	ZO	ZO	YES
ZO	PS	ZO	PS	YES					

Although the rule bases for d_1 and d_2 are identical, their control policies are different because the scaling factors for the corresponding fuzzy inputs in the two rule bases are also different. For example, λ_1 might be regarded as PM when we control class 1 arrivals or PB when we control class 2 arrivals. The scaling factors are derived in the next two subsections.

Membership Functions for the Control of Class 1 Arrivals

To determine the quantitative relationships between the fuzzy output d_1 and the fuzzy inputs s_i and λ_j, we assume that $x_1 + x_2 > 0$.

We start with s_1 and consider the special case given by rule *if s_1 is PB and s_2 is ZO and λ_1 is ZO and λ_2 is ZO, then d_1 is NO*. We need to decide whether to admit the *last* class 1 customer when there are $s_1 \geq 0$ customers already in queue 1, one customer is in server 1, and all other input variables are zero, i.e., $s_2 = 0$, $\lambda_1 = 0$, and $\lambda_2 = 0$. Let E_1 be the mean holding cost the last customer incurs in queue 1 and E_2 the mean holding cost from the time service starts at server 1 until exit from server 2. This customer is accepted only if his reward compensates his expected holding cost; that is,

$$w_1 \geq E_1 + E_2. \tag{6.7}$$

As there are already $s_1 + 1$ customers in station 1, E_1 is $h_1(s_1 + 1)/\mu_1$. The second cost component, E_2, is computed as follows. Server 1 begins serving the customer just when server 2 begins serving the previous customer. Then the state of the system is $(x_1, x_2) = (1, 1)$. From this state, the system moves to state (0, 2) if server 1 completes service earlier than server 2, or to (1, 0) otherwise. The system moves from state (0, 2) or (1, 0) to state (0, 1) and eventually to (0, 0). These transitions are shown in Figure 6.9.

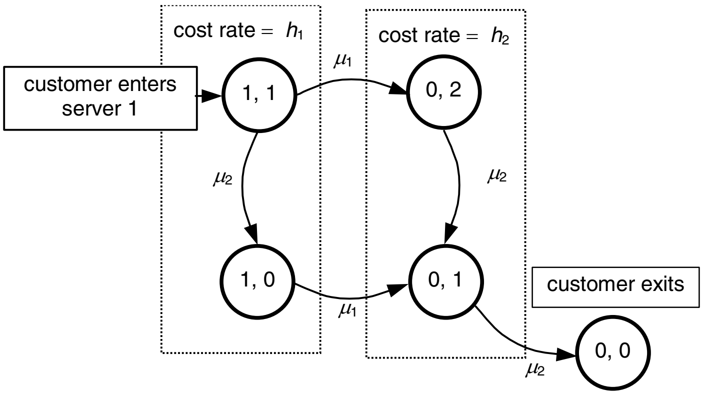


Figure 6.9. State transitions from the beginning of service at server 1 until exit from the system.

From Figure 6.9, we see that the sojourn time in state (1, 1) is a random variable with exponential distribution and mean $1/(\mu_1 + \mu_2)$. Hence, the mean holding cost from server 1 to exit becomes

$$E_2 = \frac{h_1}{\mu_1 + \mu_2}$$

$$+\frac{\mu_1}{\mu_1+\mu_2}\times \text{(mean cost from (0, 2) to exit)}$$

$$+\frac{\mu_2}{\mu_1+\mu_2}\times \text{ (mean cost from (1, 0) to exit)}$$

$$=\frac{h_1}{\mu_1+\mu_2}+\frac{\mu_1}{\mu_1+\mu_2}\frac{2h_2}{\mu_2}+\frac{\mu_2}{\mu_1+\mu_2}\left(\frac{h_1}{\mu_1}+\frac{h_2}{\mu_2}\right)$$

$$=\frac{h_1\left(1+\frac{\mu_2}{\mu_1}\right)+h_2\left(1+\frac{2\mu_1}{\mu_2}\right)}{\mu_1+\mu_2}.$$

Substituting $E_1 = h_1(s_1 + 1)/\mu_1$ and the above into inequality (6.7) and solving for s_1 yields

$$s_1 \le s_{1,d1} \triangleq \frac{\mu_1}{h_1}\left(w_1-\frac{h_1}{\mu_1}-\frac{h_2}{\mu_2}\right)-\frac{\mu_1}{h_1}\frac{h_2}{\mu_2}\frac{\mu_1}{\mu_1+\mu_2}-1. \tag{6.8}$$

Conversely, if the number of customers in queue 1 is greater than the threshold $s_{1,d1}$ defined above, then the decision is rejection. In other words, the fuzzy set PB for s_1 with membership grade 1.0 in the fuzzy rule base for d_1 is fixed at $s_{1,d1}$ and the corresponding scaling factor is $\zeta_{s1,d1} = 6/s_{1,d1}$.

We now consider the special case described by the rule *if s_1 is ZO and s_2 is PB and λ_1 is ZO and λ_2 is ZO, then d_1 is NO*. We want to determine a condition for s_2 under which we must reject a class 1 customer when $s_1 = 0$, $\lambda_1 = 0$, $\lambda_2 = 0$, and the number of customers in station 2 is $x_2 = s_2 + 1$. The class 1 customer will incur a cost $h_1/\mu_1 + h_2/\mu_2$ plus an additional queuing cost at station 2, if the customer, upon departure from server 1, finds server 2 busy. This may happen in a number of different ways. For example, the customer will depart from server 1 before any other customer departs from server 2 with probability $\mu_1/(\mu_1 + \mu_2)$. Hence, the customer will find $x_2 + 1$ customers ahead in queue 2 and the corresponding mean queuing cost in station 2 will be $h_2(x_2 + 1)/\mu_2$. Similarly, the customer will find x_2 customers ahead in queue 2 with probability $\mu_1\mu_2/(\mu_1 + \mu_2)^2$, thus incurring a queuing cost h_2x_2/μ_2, and so on. Finally, if all $s_2 + 1$ customers depart from station 2 earlier than the class 1 customer departs from server 1, then no additional cost will be incurred. This happens with probability $[\mu_2/(\mu_1 + \mu_2)]^{s2+1}$. Let $\beta = \mu_2/(\mu_1 + \mu_2)$ and E the mean holding cost the customer incurs from the beginning of service at server 1 until exit from server 2. From the above observations, we write E as follows:

$$E=\frac{h_1}{\mu_1}+\frac{h_2}{\mu_2}+\frac{\mu_1}{\mu_1+\mu_2}\frac{h_2(s_2+1)}{\mu_2}+\frac{\mu_1\mu_2}{(\mu_1+\mu_2)^2}\frac{h_2 s_2}{\mu_2}$$

$$+\ldots+\frac{\mu_1\mu_2^{s2}}{(\mu_1+\mu_2)^{s2+1}}\frac{h_2}{\mu_2}+\frac{\mu_2^{s2+1}}{(\mu_1+\mu_2)^{s2+1}}0$$

$$= \frac{h_1}{\mu_1} + \frac{h_2}{\mu_2} + \frac{\mu_1 h_2}{(\mu_1 + \mu_2)\mu_2}\left[(s_2+1)\beta^0 + s_2\beta^1 + \ldots + 1\beta^{s2}\right]$$

$$= \frac{h_1}{\mu_1} + \frac{h_2}{\mu_2}$$

$$+ \frac{\mu_1 h_2}{(\mu_1 + \mu_2)\mu_2}\left[(s_2+2)(\beta^0 + \beta^1 + \ldots + \beta^{s2}) - [\beta^0 + 2\beta^1 + \ldots + (s_2+1)\beta^{s2}]\right],$$

and upon evaluation of the geometric sums,

$$E = \frac{h_1}{\mu_1} + \frac{h_2}{\mu_2} + \frac{h_2}{\mu_1}\left[\left(\frac{\mu_2}{\mu_1 + \mu_2}\right)^{s2+1} - 1\right] + \frac{h_2(s_2+1)}{\mu_2}.$$

The condition under which the class 1 customer should be rejected is $w_1 < E$ or

$$w_1 < \frac{h_1}{\mu_1} + \frac{h_2}{\mu_2} + \frac{h_2}{\mu_1}\left[\left(\frac{\mu_2}{\mu_1 + \mu_2}\right)^{s2+1} - 1\right] + \frac{h_2(s_2+1)}{\mu_2}. \tag{6.9}$$

The above inequality is solved for s_2 by trial and error. A lower bound for s_2 is found by strengthening (6.9); thus,

$$w_1 < \frac{h_1}{\mu_1} + \frac{h_2}{\mu_2} + \frac{h_2}{\mu_1}\left[\left(\frac{\mu_2}{\mu_1 + \mu_2}\right)^{s2+1} - 1\right] + \frac{h_2(s_2+1)}{\mu_2}$$

$$\le \frac{h_1}{\mu_1} + \frac{h_2}{\mu_2} + \frac{h_2}{\mu_1}\left[\left(\frac{\mu_2}{\mu_1 + \mu_2}\right)^{1} - 1\right] + \frac{h_2(s_2+1)}{\mu_2}$$

$$= \frac{h_1}{\mu_1} + \frac{h_2}{\mu_2} - \frac{h_2}{\mu_1 + \mu_2} + \frac{h_2(s_2+1)}{\mu_2}.$$

Then, s_2 is bounded from below as follows:

$$s_2 \ge s_{2,d1} > \frac{\mu_2}{h_2}\left(w_1 - \frac{h_1}{\mu_1} - \frac{h_2}{\mu_2}\right) - \frac{\mu_1}{\mu_1 + \mu_2}, \tag{6.10}$$

where $s_{2,d1}$ is the smallest value (not necessarily an integer) for which s_2 should be interpreted as PB with grade 1.0. The corresponding scaling factor is $\zeta_{s1,d1} = 6/s_{2,d1}$.

Regarding the fuzzy input λ_1, we consider the rule *if s_1 is ZO and s_2 is ZO and λ_1 is PB and λ_2 is ZO, then d_1 is NO*. We have then reduced the original system to a serial, two-node queuing network with a single arrival stream. In this case, the rule dictates that an arriving class 1 customer is rejected, whereas there is only one customer in the system; that is, $x_1 + x_2 = 1$. This is equivalent to imposing an upper bound $C = 1$ to the number of customers in the system. The decision to reject the arriving customer is optimal if the mean profit rate of the system when $C = 1$ is greater than the mean profit rate when $C = 2$. For any fixed C, the mean profit rate J_C of the system is given by

$$J_C = w_1\lambda_1 \, P(\text{acceptance}) - \sum_{i=1}^{2} h_i \begin{pmatrix} \text{mean number of} \\ \text{customers in station } i \end{pmatrix}$$

$$= w_1\lambda_1 P_C(x_1 + x_2 < C) - \sum_{x_1=0}^{C} \sum_{x_2=0}^{C-x_1} (h_1 x_1 + h_2 x_2) P_C(x_1, x_2)\,, \qquad (6.11)$$

where $P_C(x_1, x_2)$ is the equilibrium probability of state (x_1, x_2) given that $x_1 + x_2 \le C$. The state transitions for $C = 1$ and 2 are shown in Figure 6.10. The transitions for $C = 1$ are a subset of those for $C = 2$ and are marked by heavy arrows.

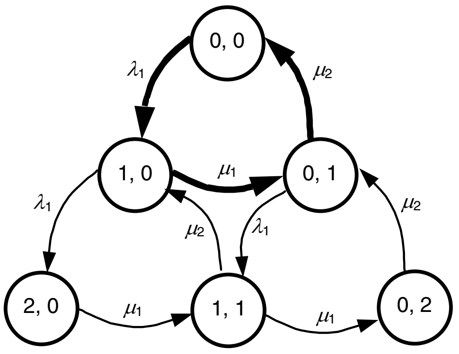


Figure 6.10. State transitions for $C = 1$ (in bold) and for $C = 2$.

By analyzing the Markov chains of Figure 6.10, we compute the corresponding equilibrium probabilities and from these we find J_1 and J_2. For $C = 1$, the equilibrium probabilities $P_1(x_1, x_2)$ satisfy the Chapman–Kolmogorov equations

$$\lambda_1 P_1(0, 0) = \mu_2 P_1(0, 1),$$

$$\mu_1 P_1(1, 0) = \lambda_2 P_1(0, 0),$$

$$\mu_2 P_1(0, 1) = \mu_1 P_1(1, 0),$$

and the normalization equation $P_1(0, 0) + P_1(0, 1) + P_1(1, 0) = 1$. From these equations, we obtain

$$P_1(0, 0) = \frac{1}{1 + \dfrac{\lambda_1}{\mu_2} + \dfrac{\lambda_2}{\mu_1}}\,,$$

$$P_1(0, 1) = \frac{\lambda_1}{\mu_2} \frac{1}{1 + \dfrac{\lambda_1}{\mu_2} + \dfrac{\lambda_2}{\mu_1}}\,,$$

$$P_1(1,0) = \frac{\lambda_2}{\mu_1} \frac{1}{1+\frac{\lambda_1}{\mu_2}+\frac{\lambda_2}{\mu_1}}.$$

From Equation (6.11), the mean profit rate is given by

$$J_1 = w_1\lambda_1 P_1(0,0) - h_1 P_1(1,0) - h_2 P_2(0,1).$$

The mean profit rate J_2 for $C = 2$ is computed similarly from Equation (6.11) and the corresponding equilibrium probabilities $P_2(0, 0)$, …, $P_2(2, 0)$, $P_2(1, 1)$, and $P_2(0, 2)$, but the solutions to the Chapman–Kolmogorov equations are lengthy and will be omitted. After a little algebra, the condition $J_1 > J_2$ under which it is optimal to reject an arrival when $x_1 + x_2 = 1$ is equivalent to

$$\begin{aligned}&\lambda_1\left[\frac{h_1}{\mu_1}\left(\frac{\mu_2}{\mu_1}+2\right)+\frac{h_2}{\mu_2}\left(\frac{\mu_1}{\mu_2}+2\right)-w_1\right] > \\ &\left[w_1(\mu_1+\mu_2)-h_1-h_2-\frac{2\mu_2 h_1}{\mu_1}-\frac{2\mu_1 h_2}{\mu_2}\right].\end{aligned} \tag{6.12}$$

The range of values of λ_1 for which the above inequality is satisfied depends on the signs of the terms inside the brackets.

- If inequality (6.12) is not satisfied for any nonnegative value λ_1, then we should always admit class 1 customers when $s_1 = 0$, $s_2 = 0$, and $\lambda_2 = 0$. Hence, the rule *if s_1 is ZO and s_2 is ZO and λ_1 is PB and λ_2 is ZO, then d_1 is NO* should never fire. This implies that the fuzzy input λ_1 should never be declared PB for any crisp value of λ_1. Hence, we use a scaling factor $\zeta_{\lambda 1, d1} = 0$ for λ_1.
- If inequality (6.12) is satisfied for *any* nonnegative value λ_1, then we should always reject class 1 customers when $s_1 = 0$, $s_2 = 0$, and $\lambda_2 = 0$. Hence, the rule *if s_1 is ZO and s_2 is ZO and λ_1 is PB and λ_2 is ZO, then d_1 is NO* should fire even when $\lambda_1 = 0$. This in turn implies that the fuzzy input λ_1 should always be declared as PB for any crisp value of λ_1; in which case, we use a scaling factor ∞ for λ_1 or, equivalently, we scale λ_1 up to 6.
- If inequality (6.12) is satisfied for any λ_1 greater than or equal to a positive number $\lambda_{1,d1}$, then the fuzzy set PB for λ_1 with membership grade 1.0 in the fuzzy rule base for d_1 is fixed at $\lambda_{1,d1}$.

Finally, for the fuzzy input λ_2, we consider the rule *if s_1 is ZO and s_2 is ZO and λ_1 is ZO and λ_2 is PB, then d_1 is NO*. This rule reflects the situation in which the last arriving class 1 customer is rejected while there is only one customer (in service) in station 2, and no additional class 1 customers are expected to arrive, or $x_1 = 0$, $x_2 = 1$, $s_1 = 0$, $s_2 = 0$, $\lambda_1 = 0$, and $\lambda_2 \geq 0$. The problem here is to find a critical value $\lambda_{2,d1}$ such that the decision to accept a customer is optimal for all $\lambda_2 \leq \lambda_{2,d1}$. To solve this problem rigorously, we should employ dynamic programming. Here we derive a threshold using heuristic arguments and intuition.

If the class 1 customer is rejected while there is another one in server 2, then the profit per unit time J^{reject} will be given by $w_2 - h_2/\mu_2$ divided by the expected time until a next arrival. This time is equal to the expected service time $1/\mu_2$ plus the expected time until the next event (departure or arrival) in the system, which is given by $1/(\lambda_2 + \mu_2)$. Thus, the profit rate is

$$J^{\text{reject}} = \frac{w_2 - \dfrac{h_2}{\mu_2}}{\dfrac{1}{\mu_2} + \dfrac{1}{\lambda_2 + \mu_2}}.$$

If, on the other hand, the customer is admitted, the system earns a net reward $w_1 - h_1/\mu_1 - h_2/\mu_2$ over the period of service $1/\mu_1 + 1/\mu_2$. Thus, the profit rate becomes

$$J^{\text{accept}} = \frac{w_1 - \dfrac{h_1}{\mu_1} - \dfrac{h_2}{\mu_2}}{\dfrac{1}{\mu_1} + \dfrac{1}{\mu_2}}.$$

The condition under which it pays more to accept the customer is $J^{\text{accept}} \geq J^{\text{reject}}$, which is equivalent to

$$\lambda_2 \leq \lambda_{2,d1} \triangleq \frac{2w_1\mu_1\mu_2 - w_2\mu_2(\mu_1 + \mu_2) - 2h_1\mu_2 + h_2(\mu_2 - \mu_1)}{w_2(\mu_1 + \mu_2) - w_1\mu_1 + h_1 - h_2}. \tag{6.13}$$

Conversely, if $\lambda_2 > \lambda_{2,d1}$ then the decision is rejection. In the rule base, a value $\lambda_{2,d1}$ for λ_2 is declared PB with membership grade 1.0.

Membership Functions for the Control of Class 2 Arrivals

The rule base for the fuzzy output d_2 is identical to that for d_1. We now proceed as before to specify the numerical settings of s_i and λ_j for the fuzzy output d_2. We assume that $x_2 > 0$.

We start with s_1 by considering the special case described by rule *if s_1 is PB and s_2 is ZO and λ_1 is ZO and λ_2 is ZO, then d_2 is NO*. We want to determine a condition for s_1 under which we must reject a class 2 customer when $\lambda_1 = 0$, $\lambda_2 = 0$, $s_2 = 0$, and because we have assumed $x_2 > 0$, there is one customer in server 2. If we accept the customer, then the state of the system at time zero will be $(s_1 + 1, 2)$, otherwise the system will start from state $(s_1 + 1, 1)$. Our objective is to maximize the expected total profit until the system clears, that is, until it reaches state $(0, 0)$. Let $F(i, j)$ be the total expected holding cost from state (i, j) to state $(0, 0)$. If both i and j are greater than 0, then the system moves from state (i, j) to $(i - 1, j + 1)$ with probability $\mu_1/(\mu_1 + \mu_2)$ or to state $(i, j - 1)$ with the complementary probability. When $i = 0$ ($j = 0$), the system will visit state $(0, j - 1)$ [respectively $(i - 1, 1)$] with probability 1. Therefore, $F(i, j)$ can be computed recursively from

$$F(i,j)=\begin{cases} j\dfrac{h_2}{\mu_2}+F(0,j-1) & \text{if } i=0 \\ i\dfrac{h_1}{\mu_1}+F(i-1,1) & \text{if } j=0 \\ \dfrac{ih_1+jh_2+\mu_1 F(i-1,j+1)+\mu_2 F(i,j-1)}{\mu_1+\mu_2} & \text{if } i,j>0 \end{cases}.$$

The condition under which the class 2 customer should be rejected is

$$w_2 - F(s_1+1, 2) < -F(s_1+1, 1).$$

It can be verified by induction that $F(s_1 + 1, 2) - F(s_1 + 1, 1)$ is increasing in s_1. Therefore, if the above condition holds for some value $s_{1,d2}$, then it also holds for every $s_1 \geq s_{1,d2}$. Consequently, the fuzzy set PB for s_1 with membership grade 1.0 in the fuzzy rule base for d_2 is fixed at

$$s_{1,d2} = \min\{s_1: F(s_1+1, 2) - F(s_1+1, 1) > w_2\}. \tag{6.14}$$

We now consider the special case described by the rule *if s_1 is ZO and s_2 is PB and λ_1 is ZO and λ_2 is ZO, then d_2 is NO*. We want to determine a condition for s_2 under which we must reject a class 2 customer when $x_1 = s_1 = 0$, $\lambda_1 = 0$, $\lambda_2 = 0$, and the number of customers already in station 2 is $x_2 = s_2 + 1$. This decision is optimal if the expected holding cost for the new customer is greater than the corresponding reward, or $w_2 < (s_2 + 2)h_2/\mu_2$. Therefore, the fuzzy set PB for s_2 with membership grade 1.0 in the fuzzy rule base for d_2 is fixed at

$$s_{2,d2} \triangleq \frac{\mu_2}{h_2} w_2 - 2. \tag{6.15}$$

Regarding the fuzzy input λ_1, we consider the rule *if s_1 is ZO and s_2 is ZO and λ_1 is PB and λ_2 is ZO, then d_2 is NO*. In this case, an arriving class 2 customer is rejected, and $x_i = 1$, $s_i = 0$, $i = 1, 2$, and only class 1 arrivals are allowed. As previously, we shall determine the optimal rejection threshold for λ_1 using heuristic arguments.

If the class 2 customer is rejected while there is one in server 1, then the profit per unit time J^{reject} will be given by $w_1 - h_1/(\mu_1 + \mu_2) - h_2/\mu_2$ divided by the expected time until a next arrival. This time is equal to the expected service time $1/\mu_2$ plus the expected time until the next event (departure or arrival) related to class 1 customers, which is given by $1/(\lambda_2 + \mu_2)$. Thus, the profit rate is

$$J^{\text{reject}} = \frac{w_1 - \dfrac{h_1}{\lambda_1+\mu_1} - \dfrac{h_2}{\mu_2}}{\dfrac{1}{\mu_2} + \dfrac{1}{\lambda_1+\mu_1}}.$$

If, on the other hand, the customer is admitted, the system earns a net reward $w_2 - h_2[2/\mu_2 - 1/(\lambda_2 + \mu_2)]$ over the period of service $1/\mu_2$. Thus, the profit rate becomes

$$J^{\text{accept}} = \frac{w_2 - h_2\left(\dfrac{2}{\mu_2} - \dfrac{1}{\lambda_2 + \mu_2}\right)}{\dfrac{1}{\mu_2}}.$$

The condition under which it pays more to reject the customer is $J^{\text{reject}} \geq J^{\text{accept}}$, which is equivalent to

$$\lambda_1 > \lambda_{1,d2} = \frac{w_1\mu_2 + h_1 - h_2}{w_1 - w_2} - \mu_1 . \tag{6.16}$$

Therefore, we fix PB for λ_1 in the rule base for the decision d_2 with membership grade 1.0 at $\lambda_{1,d2}$.

Lastly, the rule *if s_1 is ZO and s_2 is ZO and λ_1 is ZO and λ_2 is PB, then d_2 is NO* corresponds to the case where an arriving class 2 customer is rejected when there is one and only one customer (in service) in station 2 and no class 1 customers admitted, or $x_1 = s_1 = 0$, $x_2 = 1$, $s_2 = 0$, and $\lambda_1 = 0$. This case is identical to the one described in Section 6.2 if we neglect class 1 customers. A condition under which it is optimal to reject the arriving customer is obtained as follows. We know from Section 6.2 that the decision to reject is socially optimal if x_2 is greater than or equal to a threshold value x_s, which is the unique solution to inequalities (6.2). In our case, rejection should be optimal for all $x_2 \geq 1$. Therefore, setting $x_s = 1$ and $\rho = \lambda_2/\mu_2$ in inequalities (6.2), we obtain

$$\frac{(1-\rho)-\rho(1-\rho)}{(1-\rho)^2} \leq \frac{w_2\mu_2}{h_2} < \frac{2(1-\rho)-\rho(1-\rho^2)}{(1-\rho)^2}$$

or

$$1 \leq \frac{w_2\mu_2}{h_2} < \frac{2-\rho(1+\rho)}{1-\rho}.$$

The inequality on the left side is true by condition (6.6). It can be verified that the inequality on the right side is true if and only if $\rho > w_2\mu_2/h_2 - 2$ or

$$\lambda_2 > \lambda_{2,d2} \triangleq \mu_2\left(\frac{w_2\mu_2}{h_2} - 2\right). \tag{6.17}$$

Therefore, we fix PB for λ_2 in the rule base for the decision d_2 with membership grade 1.0 at $\lambda_{2,d2}$.If $\lambda_{2,d2}$ is negative, then λ_2 is declared PB with grade 1.0.

6.5.3 A Numerical Example

We examine the network shown in Figure 6.8 with parameters $\lambda_1 = \lambda_2 = 0.5$, $\mu_1 = 1$, $\mu_2 = 1.5$, $w_1 = 10$, $w_2 = 6$, and $h_1 = h_2 = 1$.

We determine the optimal policy for d_1 from the architecture of the fuzzy logic controller shown in Table 6.3. The algorithm is outlined as follows:

1. According to expressions (6.7)–(6.13) and the given information, we determine the scaling factors for the fuzzy inputs s_1, s_2, λ_1, and λ_2 in the rule base for d_1. For this example, we have $s_{1,\,d1} = 7.07$, which corresponds to PB for s_1 with membership grade 1.0 in the rule base for d_1. From Figure 6.2a, PB for s_1 with membership grade 1.0 is 6 in the universe of discourse. Hence, we obtain the scaling factor $\zeta_{s1,\,d1} = 0.85$ for s_1 in the rule base for d_1. Similarly, we obtain $\zeta_{s2,\,d1} = 0.46$, $\zeta_{\lambda 1,\,d1} = 0$, and $\zeta_{\lambda 2,\,d1} = 6$.
2. We start the algorithm from an initial state $s_1 = s_2 = 0$.
3. Using the current s_1 and s_2, as well as the given λ_1 and λ_2 as crisp inputs, we determine the decision via fuzzification, fuzzy inference, and defuzzification.
4. We plot the decision d_1 in the two-dimensional plane of s_1 and s_2.
5. If $d_1 = 0$, we go to step (6); otherwise we set $s_2 = s_2 + 1$ and go to (3),
6. If $d_1 = 0$, the calculations stop; otherwise we set $s_1 = s_1 + 1$ and go to (3).

We describe step (3) of the above algorithm via an example. Suppose that $s_1 = 2$ and $s_2 = 2$, which should be scaled down to 1.7 and 0.92, respectively. We see from Figure 6.2a that s_1 corresponds to ZO with grade 0.43, PS with grade 0.90 and PM with grade 0.23, and s_2 corresponds to ZO with grade 0.69 and PS with grade 0.64. In addition, $\lambda_1 = 0.5$ is ZO with grade 1.0 and $\lambda_2 = 0.5$ is PS with grade 0.67 and PM with grade 0.67. The four inputs s_1, s_2, λ_1, and λ_2 have 3, 2, 1, and 2 fuzzy sets, respectively; and hence, $3 \times 2 \times 1 \times 2 = 12$ fuzzy decisions are fired for d_1. According to the fuzzy rule base and Mamdani implication, which is associated with the min-operation, the fuzzy decisions d_1 are formulated as follows:

If s_1 is ZO with grade 0.43 and s_2 is ZO with grade 0.69 and λ_1 is ZO with grade 1.00 and λ_2 is PS with grade 0.67, then d_1 is YES with grade 0.43.

If s_1 is ZO with grade 0.43 and s_2 is ZO with grade 0.69 and λ_1 is ZO with grade 1.00 and λ_2 is PM with grade 0.67, then d_1 is YES with grade 0.43.

If s_1 is ZO with grade 0.43 and s_2 is PS with grade 0.64 and λ_1 is ZO with grade 1.00 and λ_2 is PS with grade 0.67, then d_1 is YES with grade 0.43.

If s_1 is ZO with grade 0.43 and s_2 is PS with grade 0.64 and λ_1 is ZO with grade 1.00 and λ_2 is PM with grade 0.67, then d_1 is NO with grade 0.43.

If s_1 is PS with grade 0.90 and s_2 is ZO with grade 0.69 and λ_1 is ZO with grade 1.00 and λ_2 is PS with grade 0.67, then d_1 is YES with grade 0.67.

If s_1 is PS with grade 0.90 and s_2 is ZO with grade 0.69 and λ_1 is ZO with grade 1.00 and λ_2 is PM with grade 0.67, then d_1 is NO with grade 0.67.

If s_1 is PS with grade 0.90 and s_2 is PS with grade 0.64 and λ_1 is ZO with grade 1.00 and λ_2 is PS with grade 0.67, then d_1 is NO with grade 0.64.

If s_1 is PS with grade 0.90 and s_2 is PS with grade 0.64 and λ_1 is ZO with grade 1.00 and λ_2 is PM with grade 0.67, then d_1 is NO with grade 0.64.

If s_1 is PM with grade 0.23 and s_2 is ZO with grade 0.69 and λ_1 is ZO with grade 1.00 and λ_2 is PS with grade 0.67, then d_1 is NO with grade 0.23.

If s_1 is PM with grade 0.23 and s_2 is ZO with grade 0.69 and λ_1 is ZO with grade 1.00 and λ_2 is PM with grade 0.67, then d_1 is NO with grade 0. 23.

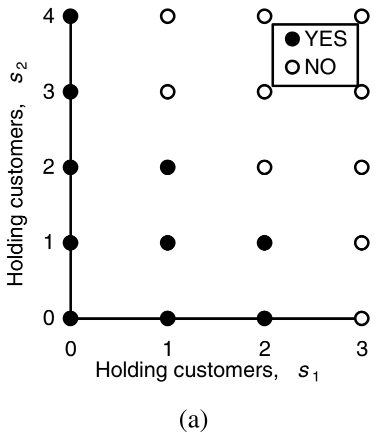
If s_1 is PM with grade 0.23 and s_2 is PS with grade 0.64 and λ_1 is ZO with grade 1.00 and λ_2 is PS with grade 0.67, then d_1 is NO with grade 0. 23.

If s_1 is PM with grade 0.23 and s_2 is PS with grade 0.64 and λ_1 is ZO with grade 1.00 and λ_2 is PM with grade 0.67, then d_1 is NO with grade 0. 23.

From Figure 6.2a, the peak values and heights of the fuzzy decisions d are $e_1 = e_2 = e_3 = e_5 = 1$, $e_4 = e_6 = \ldots = e_{12} = 0$, and $f_1 = \ldots = f_4 = 0.43$, $f_5 = f_6 = 0.67$, $f_7 = f_8 = 0.64$, $f_9 = \ldots = f_{12} = 0.23$. By the height method of defuzzification, the crisp output d^* is given by

$$d^* = \frac{\sum_{i=1}^{12} e_i f_i}{\sum_{i=1}^{12} f_i} = 0.373.$$

As $d^* < 0.5$, the decision d_1 is "No," which means that the server at station 1 should not receive an arriving class 1 customer.



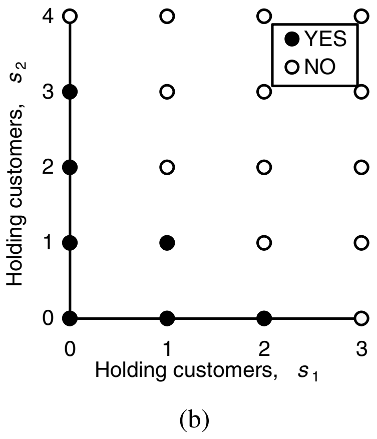


Figure 6.11. Decisions (a) d_1 and (b) d_2.

Following the previous algorithm, we obtain the fuzzy control policies for d_1 and d_2 shown in Figure 6.11. We see that although the rule bases for d_1 and d_2 are identical, their control policies are not. This is because the scaling factors for the corresponding fuzzy inputs in the two rule bases are different.

7 Coordinating Multiple Control Policies

7.1 Introduction

In the previous three chapters, we studied various queuing control problems concerning service rate selection, routing, or admission of customers separately. In practice, however, it is possible to have the option to either reject or schedule an arriving customer among different workstations and at the same time to select the service rate in each station.

Queuing models involving multiple decisions are of interest in manufacturing systems producing a variety of parts on demand, multiprocessor systems, and virtual-circuit-switching communication networks. Whenever control is formulated as a Markov or semi-Markov decision process, the techniques of dynamic programming can in principle characterize the structure of an optimal policy or numerically solve for its parameters. However, when general distributions are considered or when the queuing networks are complex, there is no effective way to tackle these problems.

In this chapter, we develop fuzzy controllers for the following four queuing models:

- Two-station tandem network with two classes of customers
- Two-station tandem network with two classes of customers and service costs
- Three-station network with two classes of customers
- Three-station network with both controlled and uncontrolled arrivals

None of these problems has a known analytical solution. We assume that deterministic stationary policies exist, which will be specified by means of the fuzzy logic. As in the previous chapters, the specific features of each controller are explored by mimicking a human operator who learns from examples.

7.2 Two Stations in Tandem with Two Arrival Streams

7.2.1 Problem Description

Our first model involving multiple decisions is a queuing network with two stations in tandem as shown in Figure 7.1. Class i, $i = 1, 2$, customers arrive at station 1 according to independent Poisson processes with rates λ_i. Class 1 customers visit station 1 only, and class 2 customers visit both stations 1 and 2 in series. There is one exponential server i in each station i, $i = 1, 2$. The service rate of server 1 can be selected to be any number u in $[0, a]$. The service in server 1 is nonpreemptive. Once a customer initiates service in station 1, its mean service rate u remains fixed until the customer completes service. In other words, the service rate of a busy server 1 and the customer being served cannot be changed. The service rate of server 2 is a constant μ. The buffers in either station have unlimited capacity, and the order of service is irrelevant.

The state of the system is described by (u, s_{11}, s_{12}, s_2), where s_{1i}, $i = 1, 2$, is the number of class i customers in station 1 including the one in service (if any) and s_2 is the number of class 2 customers in station 2 including the one in service (if any). The system incurs an instantaneous cost $h_1(s_{11} + s_{12}) + h_2 s_2$, where $h_1 > 0$ and $h_2 > 0$ are the holding costs per customer per time unit in stations 1 and 2, respectively. To ensure stability in stations 1 and 2, it is assumed that $\lambda_1 + \lambda_2 < a$ and $\lambda_2 < \mu$. The objective is to dynamically schedule server 1 nonpreemptively between the two classes of customers and determine its service rate to minimize the average cost of the system.

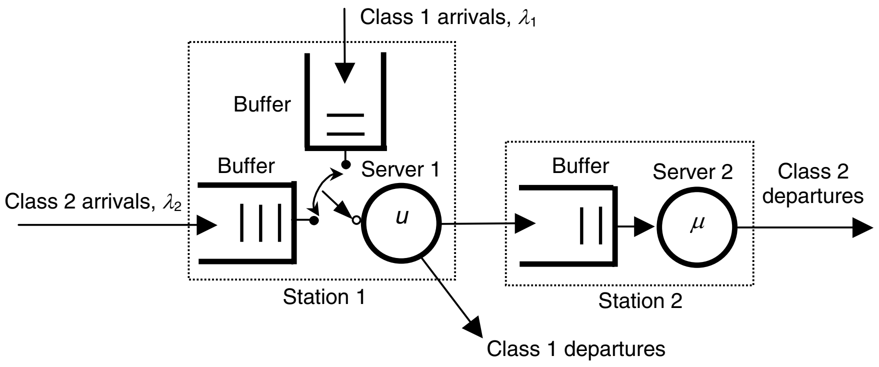


Figure 7.1. Queuing network with server scheduling and service rate selection.

This system is perhaps the simplest queuing network simultaneously dealing with both dynamic determination of service rate and scheduling of a server among different classes of customers. Harrison and Wein (1990) and Wein (1990) have provided a partial characterization of the optimal control policy for this problem. Chen *et al.* (1994) suggested and, later, Bäuerle *et al.* (1998) showed that the optimal policy has a switching-curve structure in the sense that the server in station 1 should serve a class 2 customer if $s_2 < f(s_{11}, s_{12})$ and otherwise a class 1 customer whenever available.

7.2.2 Fuzzy Controller

The state of the system, (u, s_{11}, s_{12}, s_2), changes whenever an arrival or a departure in either station occurs. Decisions are made at the times the service rate is controlled or server 1 is scheduled. Because the service is nonpreemptive and all processing and interarrival times are exponential random variables, we restrict the decision epochs to the transition epochs of state. Hence, the decision epochs are the times when a customer arrives at an empty station 1 or departs from station i, $i = 1$, 2. As there are no service costs involved, the service rate u does not bear any effects on the course of the fuzzy decision inference.

In the next subsections, we consider two different cases, $h_1 \geq h_2$ and $h_1 < h_2$.

Case A: $h_1 \geq h_2$

When the holding cost rate in station 1 is greater than that in station 2, because there are no service costs, the optimal service rate for server 1 is trivially $u \equiv a$. This policy guarantees that class 1 customers are served as fast as possible and that class 2 customers stay in the higher cost place (station 1) for the shortest possible time. The problem now is which class should an idle server 1 select when customers of both classes are available.

If we remove station 2, then the optimal policy is the so-called *$c\mu$ rule* whereby server 1 always serves the customer class that yields the largest inventory cost reduction rate. In our case, because the reduction cost rate for class 1 customers is $h_1 a$ and for class 2 customers $(h_1 - h_2)a$, it seems that class 1 customers should have preemptive priority over those of class 2. When the number s_{11} of class 1 customers in station 1 is large or the difference $s_{11} - s_{12}$ is large, this rule is the basic decision criterion. However, because of the existence of station 2, an extra condition should be attached to this priority, that is, an optimal policy ought to keep server 2 busy as long as there are class 2 customers in station 1. This condition is strengthened further by the fact that it is more beneficial for the system to keep class 2 customers in station 2 than in station 1 because $h_1 \geq h_2$.

From the above observations, we obtain the following heuristic decision criteria:

Criterion (1): The larger the number s_{11} of class 1 customers in station 1, the easier it is for server 1 to receive a class 1 customer to achieve a maximum cost reduction rate.

Criterion (2): The larger the difference $s_{11} - s_{12}$ in station 1, the easier it is for server 1 to receive a class 1 customer, again to achieve a maximum cost reduction rate.

Criterion (3): The smaller the number s_2 of class 2 customers in server 2, the easier it is for server 1 to receive a class 2 customer in order to avoid starvation of server 2 and unnecessary delays of class 2 customers in the system.

The fuzzy controller uses the numbers of customers s_{11}, s_{12}, and s_2, whose physical domain is $[0, \infty)$, as fuzzy inputs. The fuzzy output is the decision $d = 1, 2$, which indicates that server 1 should serve a class d customer. The fuzzy rule base is shown in Table 7.1. We choose four linguistic values for each input, and therefore, the rule base has a total of $4^3 = 64$ rules. From the table, we see that when $s_{11} = 0$ and $s_{12} > 0$, server 1 serves class 2 customers exclusively (Rules 4–16) and when $s_{11} > 0$ and $s_{12} = 0$, the server receives class 1 customers (Rules 17–20, 33–36, and 49–52). The remaining rules are selected according to the previous heuristic criteria. In summary, the decision $d = 2$ is strengthened when s_{11}, $s_{11} - s_{12}$, and s_2 are small and weakened otherwise. When there are no customers in station 1, i.e.,

$s_{11} = s_{12} = 0$, the decisions d in the rule base are meaningless and server 1 simply idles.

Table 7.1. Rule base for Case A.

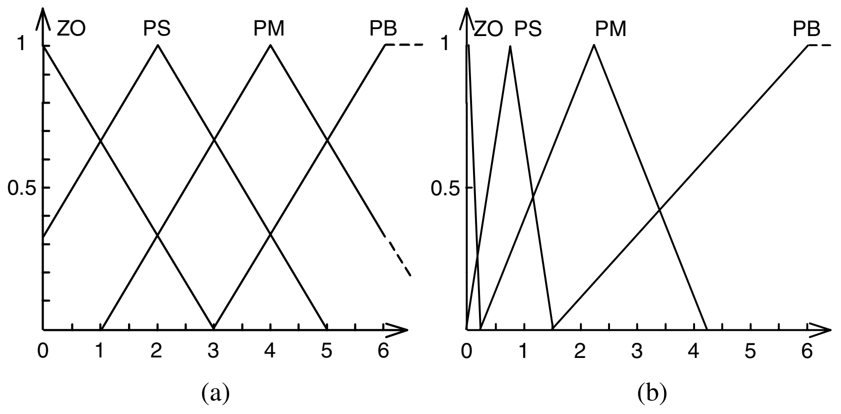
Rules 1–16				*Rules 17–32*				*Rules 33–48*				*Rules 49–64*			
s_{11}	s_{12}	s_2	d	s_{11}	s_{12}	s_2	d	s_{11}	s_{12}	s_2	d	s_{11}	s_{12}	s_2	d
ZO	ZO	ZO	2	PS	ZO	ZO	1	PM	ZO	ZO	1	PB	ZO	ZO	1
ZO	ZO	PS	1	PS	ZO	PS	1	PM	ZO	PS	1	PB	ZO	PS	1
ZO	ZO	PM	1	PS	ZO	PM	1	PM	ZO	PM	1	PB	ZO	PM	1
ZO	ZO	PB	1	PS	ZO	PB	1	PM	ZO	PB	1	PB	ZO	PB	1
ZO	PS	ZO	2	PS	PS	ZO	2	PM	PS	ZO	2	PB	PS	ZO	2
ZO	PS	PS	2	PS	PS	PS	1	PM	PS	PS	1	PB	PS	PS	1
ZO	PS	PM	2	PS	PS	PM	1	PM	PS	PM	1	PB	PS	PM	1
ZO	PS	PB	2	PS	PS	PB	1	PM	PS	PB	1	PB	PS	PB	1
ZO	PM	ZO	2	PS	PM	ZO	2	PM	PM	ZO	2	PB	PM	ZO	2
ZO	PM	PS	2	PS	PM	PS	2	PM	PM	PS	1	PB	PM	PS	1
ZO	PM	PM	2	PS	PM	PM	1	PM	PM	PM	1	PB	PM	PM	1
ZO	PM	PB	2	PS	PM	PB	1	PM	PM	PB	1	PB	PM	PB	1
ZO	PB	ZO	2	PS	PB	ZO	2	PM	PB	ZO	2	PB	PB	ZO	2
ZO	PB	PS	2	PS	PB	PS	2	PM	PB	PS	2	PB	PB	PS	1
ZO	PB	PM	2	PS	PB	PM	2	PM	PB	PM	1	PB	PB	PM	1
ZO	PB	PB	2	PS	PB	PB	1	PM	PB	PB	1	PB	PB	PB	1

The membership functions for s_{1i}, $i = 1, 2$, are shown in Figure 7.2a, and for d they are shown in Figure 7.2c. The membership functions for s_2 are shown in Figure 7.2a,b and will be explained shortly.

We now determine the relationships between the fuzzy output d and the three fuzzy inputs.

The scaling factors for all three fuzzy inputs are set at 0.75. It then remains to determine the membership functions of s_2 according to the system parameters h_1, h_2, a, and μ. As the arrival rates are equal for both classes of customers, they have no effect on the optimal policy and, therefore, on the choice of membership functions.

The input variables s_{11}, s_{12}, and s_2 determine the cost rate of the system at any time instant. Moreover, the processing rates a and μ affect the cost reduction rate. If $h_1 \approx h_2$ and $a \approx \mu$, the queuing dynamics in both stations are similar, and therefore, the membership functions for s_{11} and s_{12} as well as for s_2 should be similar, as shown in Figure 7.2a. However, if $h_1 >> h_2$, we should avoid keeping class 2 customers in station 1 or starving server 2. Hence, the priority dictated by the $c\mu$ rule is weakened. This observation leads to the membership functions for s_2 in Figure 7.2b. In a similar way, we examine the relationship between service rates μ and a. If $a << \mu$, PB for s_2 should reach 1 from above in the physical domain because it is profitable to keep s_2 as small as possible, whereas if $a >> \mu$, PS for s_2 should reach ∞ from below in the physical domain. The precise location and shape of each membership function for s_2 is determined by carrying out a few simulations with various membership functions and choosing the ones that yield the smallest cost.



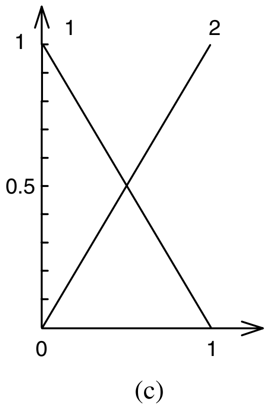


Figure 7.2. Membership functions: (a) s_{11}, s_{12}, and s_2 (if $h_1 \approx h_2$ and $a \approx \mu$); (b) s_2 (if $h_1 >> h_2$); (c) d.

Case B: $h_1 < h_2$

As in Case A, to reduce the inventory cost, whenever class 1 customers are in queue, server 1 should always serve customers, irrespective of the choice of class. This, however, does not always apply to class 2 customers because, after completion of service in server 1, they continue into station 2 where they incur higher inventory costs. It would be desirable to have always only one customer in service in station 2 unless there are no class 2 customers in the system. Such a policy ensures that class 2 customers stay in the high-cost place (station 2) for the shortest possible time. Therefore, when $s_{11} = 0$ and $s_{12} > 0$, the decision to serve class 2 customers and the selection of the corresponding service rate are based on the number of class 2 customers already in station 2. If s_2 is very low or zero, to avoid unnecessary starvation of server 2, server 1 should serve a class 2 customer as fast as possible.

As there are no service costs, the optimal control of the service rate of server 1 is of the bang-bang type. The service rate is set at the maximal value a when $s_{11} > 0$ or when $s_{11} = 0$ but class 2 customers are present ($s_{12} > 0$) and we decide that the server becomes operative; we set the service rate at the minimal value 0 if the server should idle with class 2 customers present.

When $\lambda_1 = 0$, the system becomes a tandem queuing network with a single arrival stream, which has been studied in Section 4.6. As $\lambda_1 = 0$ implies $s_{11} = 0$, the rules for which s_{11} is ZO coincide with those of Table 4.5 in Section 4.6. In this case, the control policy is dictated by a switching curve in the two-dimensional state space of s_{12} and s_2. However, the presence of class 1 customers destroys the switching structure of the policy and makes the problem more involved. As s_{11} increases, the priority of class 1 customers is gradually strengthened as long as it does not cause unnecessary starving of server 2.

Based on the above observations, we develop the fuzzy rule base of Table 7.2. We set $d = 2$ when server 1 serves class 2 customers and $d = 1$ when the server serves class 1 customers or idles, albeit with class 2 customers present. As in Case A when no customers are present in station 1, the decisions d in the rule base are meaningless and server 1 then simply idles.

Table 7.2. Rule base for Case B.

Rules 1–16				*Rules 17–32*				*Rules 33–48*				*Rules 49–64*			
s_{11}	s_{12}	s_2	d	s_{11}	s_{12}	s_2	d	s_{11}	s_{12}	s_2	d	s_{11}	s_{12}	s_2	d
ZO	ZO	ZO	2	PS	ZO	ZO	1	PM	ZO	ZO	1	PB	ZO	ZO	1
ZO	ZO	PS	1	PS	ZO	PS	1	PM	ZO	PS	1	PB	ZO	PS	1
ZO	ZO	PM	1	PS	ZO	PM	1	PM	ZO	PM	1	PB	ZO	PM	1
ZO	ZO	PB	1	PS	ZO	PB	1	PM	ZO	PB	1	PB	ZO	PB	1
ZO	PS	ZO	2	PS	PS	ZO	2	PM	PS	ZO	1	PB	PS	ZO	1
ZO	PS	PS	2	PS	PS	PS	1	PM	PS	PS	1	PB	PS	PS	1
ZO	PS	PM	1	PS	PS	PM	1	PM	PS	PM	1	PB	PS	PM	1
ZO	PS	PB	1	PS	PS	PB	1	PM	PS	PB	1	PB	PS	PB	1
ZO	PM	ZO	2	PS	PM	ZO	2	PM	PM	ZO	2	PB	PM	ZO	1
ZO	PM	PS	2	PS	PM	PS	2	PM	PM	PS	1	PB	PM	PS	1
ZO	PM	PM	2	PS	PM	PM	1	PM	PM	PM	1	PB	PM	PM	1
ZO	PM	PB	1	PS	PM	PB	1	PM	PM	PB	1	PB	PM	PB	1
ZO	PB	ZO	2	PS	PB	ZO	2	PM	PB	ZO	2	PB	PB	ZO	2
ZO	PB	PS	2	PS	PB	PS	2	PM	PB	PS	2	PB	PB	PS	1
ZO	PB	PM	2	PS	PB	PM	2	PM	PB	PM	1	PB	PB	PM	1
ZO	PB	PB	2	PS	PB	PB	1	PM	PB	PB	1	PB	PB	PB	1

An example is in order to explain the rule base. When station 2 is empty, we should serve class 2 customers (if any) first in station 1 to avoid unnecessary starving of server 2. This gives rise to a number of rules in Table 7.2 for which s_2 is ZO and $d = 2$. On the contrary, when there are a few class 2 customers in both stations, we would let class 2 customers in station 1 wait a little and serve class 1 customers (if any) first. Indeed, serving a class 1 customer first offers the highest possible rate of inventory cost reduction at a small risk of starving server 2. This gives rise to the rule: *If s_{11} is PS and s_{12} is PS and s_2 is PS, then d is 1* in Table 7.2. In a similar way, we obtain the remaining rule base.

Following the ideas of Case A, we set the membership functions for both s_{11} and s_{12} as in Figure 7.2a, for s_2 as in Figure 7.2b, and for d as in Figure 7.2c.

7.2.3 A Numerical Example

Suppose the queuing network shown in Figure 7.1 has parameters $a = \mu = 0.1$, $h_1 = 2.8$, and $h_2 = 1.4$. This example corresponds to Case A.

We determine the optimal policy directly from the architecture of the fuzzy controller as follows:

1. We arbitrarily define three numbers S_{11}, S_{12}, and S_2 as the upper limits of s_{11}, s_{12}, and s_2, respectively, and we set $s_{11} = s_{12} = s_2 = 0$.
2. Using the current s_{11}, s_{12}, and s_2 as crisp inputs, we determine the decision d via fuzzification, fuzzy inference based on the rule base in Table 7.1 or Table 7.2, whichever applies, and defuzzification.
3. We plot the decision d in the three-dimensional space of s_{11}, s_{12}, and s_2.
4. If $s_2 < S_2$, we set $s_2 = s_2 + 1$ and go to (2); otherwise we set $s_2 = 0$ and go to (5).
5. If $s_{12} < S_{12}$, we set $s_{12} = s_{12} + 1$ and go to (2); otherwise we set $s_{12} = 0$ and go to (6).
6. If $s_{11} < S_{11}$, we set $s_{11} = s_{11} + 1$ and go to (2); otherwise we stop.

We describe step (2) of the above algorithm via an example. Suppose that $s_{11} = s_{12} = 4$ and $s_2 = 2$. Using 0.75 as a scaling factor, s_{11} and s_{12} are scaled down to 3 and s_2 is called down to 1.5. We see from Figure 7.2a that both s_{11} and s_{12} correspond to PS or PM with membership grades 0.67. From Figure 7.2b, s_2 is PM with grade 0.625. The inputs s_{11}, s_{12}, and s_2 have 2, 2, and 1 fuzzy sets, respectively, and hence, $2 \times 2 \times 1 = 4$ fuzzy decisions are fired for d. According to the fuzzy rule base of Table 7.1 and Mamdani implication, the four fuzzy decisions d correspond to the linguistic value 1 with membership grade 0.625. In view of Figure 7.2c, the defuzzified output d^* is 0, which corresponds to the decision 1.

By repeating step (2) of the algorithm for various values of the input variables, we obtain a policy of a switching-surface form in the three-dimensional space of s_{11}, s_{12}, and s_2. When $s_2 \geq 1$, the fuzzy controller dictates $d = 1$ irrespective of the other inputs; that is, server 1 always serves class 1 customers, if any exist. The fuzzy control policy for $s_2 = 0$ is a switching curve as shown in Figure 7.3. The switching surface divides the state space into two parts in which either class 1 or class 2 customers should be served by server 1 exclusively. When there are no customers present in the system, the server in station 1 simply idles.

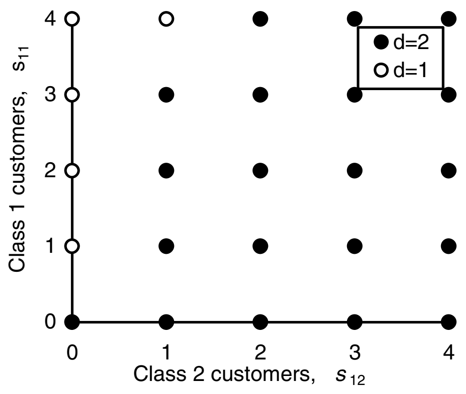


Figure 7.3. Decisions d dictated by the fuzzy control policy when $s_2 = 0$.

7.3 Two Stations in Tandem with Two Arrival Streams and Service Costs

7.3.1 Problem Description

Now we extend the previous model by introducing service costs whenever server 1 is used. The service rate u of server 1 is selected from a finite countable set of service rates u_k, $k = 1, 2, \ldots, m$ where $0 \leq u_1 < \ldots < u_m < \infty$ with corresponding service cost per unit time r_k, where $0 \leq r_1 < \ldots < r_m < \infty$. Again we assume that $\lambda_1 + \lambda_2 < u_m$ and $\lambda_2 < \mu$. The controller's task is to dynamically schedule server 1 between the two classes of customers and determine the server's rate u in order to minimize the average holding and service costs of the system over an infinite horizon.

It may be conjectured that a "bang-bang" control is not optimal because of the existence of service costs, and therefore, the determination of the optimal policy will be more involved.

7.3.2 Fuzzy Controller

The state of the system is described by four variables (s_{11}, s_{12}, s_2, k), where s_{11}, s_{12}, and s_2 have the same meaning as in Section 7.2, and k is the service type. As previously, the decision epochs coincide with the times when a customer arrives at an empty station 1 and/or leaves station i, $i = 1, 2$. A significant difference is that, because of the service costs, the service type k does play an important role in the fuzzy decision inference. Again, we consider two cases $h_1 \geq h_2$ and $h_1 < h_2$.

Case A: $h_1 \geq h_2$

All previous ideas on how to schedule server 1 are valid here. We give priority to class 1 over class 2 customers in station 1 under the condition that server 2 should not be starved as long as there are class 2 customers in the system. Hence, we use the fuzzy rule base shown in Table 7.1 and the membership functions for s_{11}, s_{12}, s_2, and d shown in Figure 7.2.

Next, we develop one more rule base to determine the optimal service rate for server 1. In general, the choice of u depends on the current rates of inventory and service costs. When the inventory costs are relatively high, u should be increased, and when the service cost is relatively high, the rate should be decreased. We choose the current service rate type $k = 1, 2, \ldots, m$ and holding cost of the system per time unit $hs \in (0, \infty)$ as the fuzzy inputs of the rule base and the variation δk of the service rate type as the fuzzy output, where $\delta k = -(m - 1), \ldots, -1, 0, 1, \ldots (m - 1)$. The inventory cost rate depends on the preceding decision d regarding the class of customer to be next served. For example, if $d = 1$ and the current inventory cost rate in station 1 $h_1(s_{11} + s_{12})$ is higher than the current service cost rate r_k, then δk should be set to a large value in order to reduce the inventory costs because, after the completion of service, class 1 customers will leave the system. However, if $d = 2$ and $h_2 s_2$ is greater than $h_1(s_{11} + s_{12})$, a high service rate in server 1 could be less beneficial or even detrimental for the system because class 1 customers will stay in the high-cost place (station 2) longer. In order to incorporate this condition into the decision-making process, we define hs as the nonnegative part of the holding cost rate difference between the two stations. Symbolically,

$$hs = \max\{0, h_1(s_{11} + s_{12}) - h_2 s_2 1_{\{d=2\}}\}, \tag{7.1}$$

where $1_{\{d=2\}}$ is an indicator function that takes the value 1 if $d = 2$ and 0 otherwise. This expression says that when a class 1 customer is to be next served, only the holding cost rate in station 1 is taken into account; otherwise, when a class 2 customer is to be next served, the difference between the holding cost rates in both stations is considered.

The membership functions for k and δk are shown in Figure 7.4, and for hs, they are shown in Figure 7.2b. Server 1 amends its service rate type by simply adding the defuzzified crisp output $\delta k^* = -(m - 1), \ldots, -1, 0, 1, \ldots (m - 1)$ to the current type k. We see from Figure 7.2b that the membership functions for hs are denser for small values of hs. Hence, when hs is small, we change the service rate without delay, whereas as hs increases, the rate is less prone to changes.

Next, we develop the rule base for δk. We examine two extreme cases. When the service rate of server 1 has its highest value but hs is zero, the server should be switched down to the slowest service rate u_1. This yields the rule *if k is PB and hs is ZO, then δk is NB*. Conversely, when $u = u_1$ and hs is very high, the rate should be increased to the largest value, which is equivalent to the rule *if k is ZO and hs is PB, then δk is PB*. All other rules fall within these two extreme cases. The complete rule base is shown in Table 7.3.

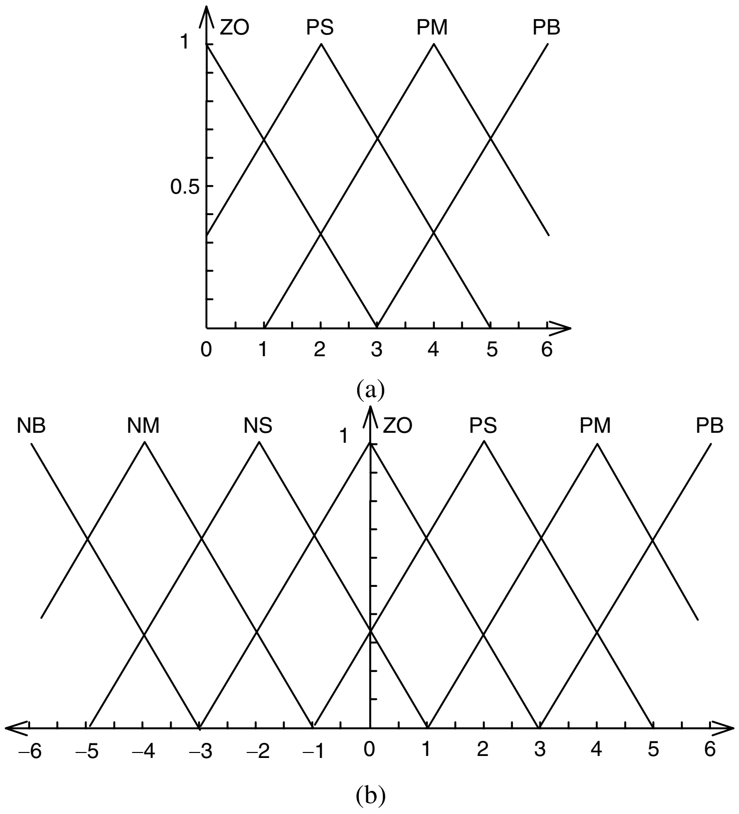


Figure 7.4. Membership functions: (a) k and (b) δk.

Table 7.3. Rule base for the service rate type in Case A.

Rules 1–8			*Rules 9–16*		
hs	*k*	δk	*hs*	*k*	δk
ZO	ZO	ZO	ZO	PM	NM
PS	ZO	PS	PS	PM	NS
PM	ZO	PM	PM	PM	ZO
PB	ZO	PB	PB	PM	PS
ZO	PS	NS	ZO	PB	NM
PS	PS	ZO	PS	PB	NM
PM	PS	PS	PM	PB	NS
PB	PS	PM	PB	PB	ZO

Finally, we determine the scaling factors for the input variables of the rule base. When $k = 1$, the corresponding fuzzy input is ZO with membership grade 1, and when $k = m$ (the highest rate type), the fuzzy input is declared PB with grade 1. Hence, the scaling factor for k is $6/(m - 1)$. The scaling factor for the other input variable *hs* is determined directly from Rule 4 of Table 7.3: *If hs is PB and k is ZO, then* δk *is PB*. In this case, we have an individually optimal criterion whereby it is better to pay more for a higher service rate when the difference of the service costs exceeds the difference of the holding costs *hs*, which is given by Equation (7.1). Therefore, the peak value of the fuzzy set PB for *hs* is fixed at $r_m - r_1$ and the corresponding scaling factor is $6/(r_m - r_1)$.

Case B: $h_1 < h_2$

We also treat this case by decomposing the decision-making process into two stages. At each stage, we combine ideas from previous cases. Specifically, for the fuzzy output *d*, we use the same inputs, rule bases, and membership functions as in Case B of Section 7.2, and for the fuzzy output δk, the fuzzy inputs, rule bases, and membership functions coincide with those of Case A.

7.3.3 A Numerical Example

We examine a two-station tandem queuing network with two classes of customers and service costs. The server in the first station can select among seven service rates $u_k = 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10$ with corresponding cost rates $r_k = 10, 20, 30, 40, 50, 60, 70$, where $k = 1, 2, \ldots, 7$. The rest of the parameters are as in the example of Section 7.2.3. This example corresponds to Case A.

We determine the optimal policy from the architecture of the fuzzy controller using the algorithm of Section 7.2.3. In step (2) of the algorithm, in addition to the decision *d*, we determine δk via fuzzification, fuzzy inference, and height defuzzification using *k* and *hs* as inputs.

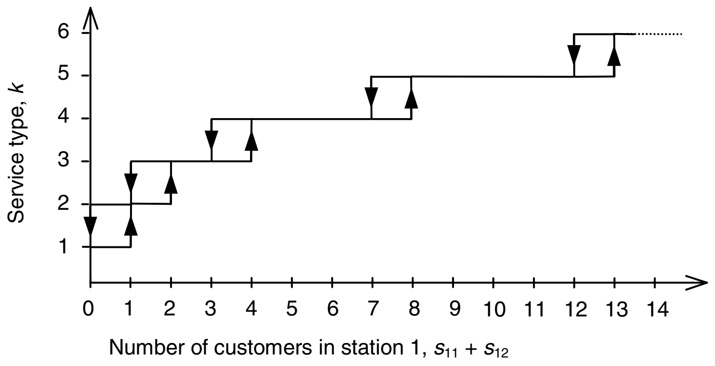


Figure 7.5. The connected increasing policy. (J Intell Fuzzy Syst, Vol. 8, p. 38, by Runtong Zhang and Yannis A. Phillis. © 2000 by IOS Press. Used with permission.)

The fuzzy control policy for the service order *d* has the same switching structure as that of Section 7.2.3. The control policy for the service type *k*, which by Equation (7.1) is a function of $s_{11} + s_{12}$ (the number of customers in station 1), is shown in Figure 7.5. We see from this figure that the fuzzy control policy has the same connected increasing structure as in the case of the single queuing station serving a single arrival stream, which was discussed in Section 4.4.

7.4 Three-Station Network with Two Arrival Streams

7.4.1 Problem Description

In this section, we consider the problem of simultaneous admission control and scheduling of two customer classes for the queuing network illustrated in Figure 7.6. The system consists of three stations, and each of the stations i, $i = 0, 1, 2$, has an exponential server i whose mean processing rate is μ_i. Two classes of customers called class i, $i = 1, 2$, need to be first served nonpreemptively by server 0 and subsequently by their corresponding server i. There are always customers of both classes in front of server 0, and each of the servers 1 and 2 has its own infinite buffer called queue i.

A reward w_i, $i = 1, 2$, is earned whenever server 0 completes a service on a class i customer and a holding cost h_i per customer in queue i per unit time is incurred. We assume that the processing rate of server 0 is greater than the sum of the processing rates of servers 1 and 2, that is, $\mu_0 > \mu_1 + \mu_2$. When server 0 completes a service, a decision must be made as to whether the server will idle or which customer to serve next in order to maximize the average benefit of the system.

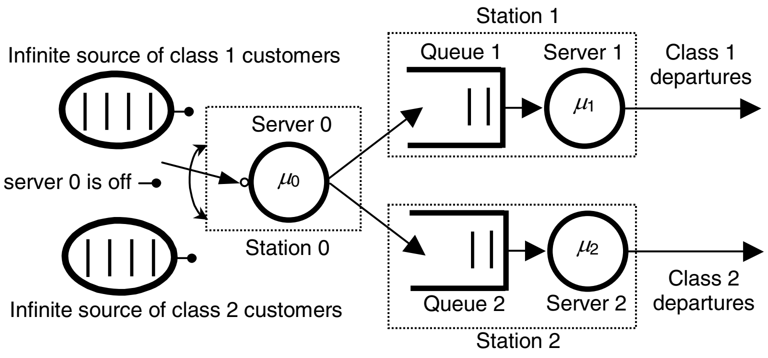


Figure 7.6. A queuing network with server scheduling and admission control.

7.4.2 Fuzzy Controller

The state of the system is described by (s_1, s_2), where $s_i = 0, 1, 2, \ldots$ is the number of class i customers in queue i, $i = 1, 2$. The state of the system changes whenever any server completes a service. However, because the service in server 0 is nonpreemptive, the decision epochs correspond to the time instants when server 0 has just completed a service.

The fuzzy controller has the state variables s_i, $i = 1, 2$, as fuzzy inputs. The fuzzy inference is completed in two stages. First, we compute the decision d_{I} of whether server 0 will next be idle (d_{I} = OFF) or working (d_{I} = ON). Second, if d_{I} = ON, then we compute the decision d_{II} about the class of customer to be served; thus, $d_{\mathrm{II}} = 1$ or 2.

We use five linguistic values, ZO, PS, PM, PB, and PVB, for each fuzzy input s_i, two values, OFF and ON, for the fuzzy output d_{I}, and two values, 1 and 2, for the fuzzy output d_{II}.

The rule base is intuitive. When there are many customers in queues 1 and 2, to avoid a high holding cost, server 0 should idle, i.e., $d_{\mathrm{I}} =$ OFF. When there are few or no customers in queue $i = 1$ or 2, in order to earn a reward, server 0 should admit a class i customer, i.e., $d_{\mathrm{II}} = i$. This corresponds to the rules *if* s_i *is ZO, then* $d_{II} = i$. However, when *both* stations 1 and 2 have few or no customers at all, a customer is randomly chosen between the two classes. In this case, we write $d_{\mathrm{II}} = *$ and the corresponding rules read *if* s_1 *is ZO(PS) and* s_2 *is ZO(PS), then* d_{II} *is* *. A rule base that relies on the above arguments is shown in Table 7.4.

Table 7.4. Rule base.

Rules 1–10				*Rules 11–20*				*Rules 21–25*			
s_1	s_2	d_{I}	d_{II}	s_1	s_2	d_{I}	d_{II}	s_1	s_2	d_{I}	d_{II}
ZO	ZO	ON	*	PM	ZO	ON	2	PVB	ZO	ON	2
ZO	PS	ON	1	PM	PS	ON	2	PVB	PS	OFF	
ZO	PM	ON	1	PM	PM	OFF		PVB	PM	OFF	
ZO	PB	ON	1	PM	PB	OFF		PVB	PB	OFF	
ZO	PVB	ON	1	PM	PVB	OFF		PVB	PVB	OFF	
PS	ZO	ON	2	PB	ZO	ON	2				
PS	PS	ON	*	PB	PS	OFF					
PS	PM	ON	1	PB	PM	OFF					
PS	PB	OFF		PB	PB	OFF					
PS	PVB	OFF		PB	PVB	OFF					

The universe of discourse for the fuzzy inputs s_i, $i = 1, 2$, is $[0, \infty)$ and for the fuzzy outputs d_{I} and d_{II} is $[0, 1]$. The membership functions for s_i are shown in Figure 7.7a, for d_{I} in Figure 7.7b, and for d_{II} in Figure 7.7c.

To determine the membership functions of s_i, we observe that the waiting time of customers increases with s_i as the sequence $s_i(s_i + 1)/2 = 1, 3, 6, \ldots$. This happens because one customer in queue i incurs a sojourn time proportional to one service time, which is the mean service time of the customer currently in service in server i. If there are two customers in queue i, the second customer will stay in the queue for two service times until the predecessor is serviced. Thus, on average, the system incurs a holding cost proportional to $1 + 2 = 3$ for both customers. Therefore, we devise the fuzzy membership functions for s_1 and s_2 as in Figure 7.7a.

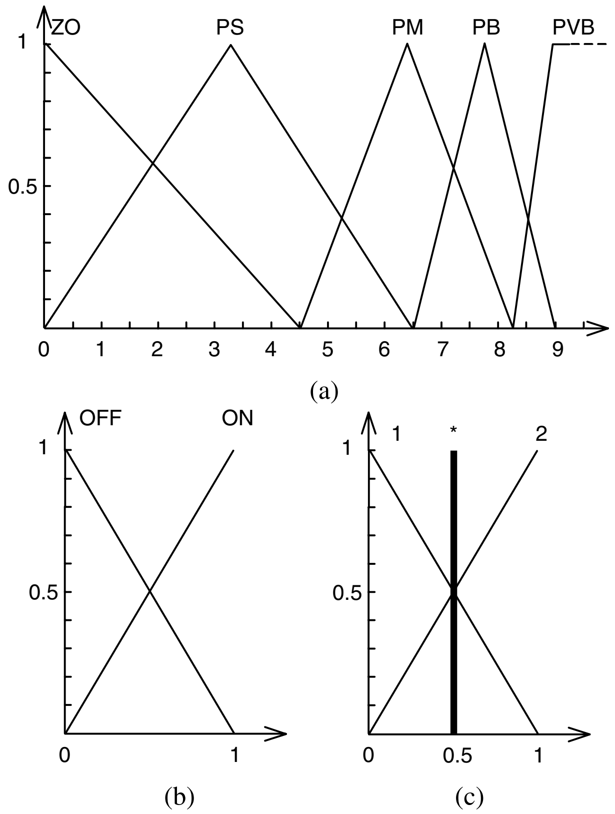


Figure 7.7. Membership functions: (a) s_1 and s_2; (b) d_I; (c) d_{II}.

We now determine the scaling factors for s_1 and s_2. Let us assume momentarily that station 2 and the source of class 2 customers have been removed. We shall determine a condition for s_1 so that a class 1 customer is rejected. Under an individually optimal criterion, this decision is optimal if the reward w_1 does not compensate for the customer's expected holding cost in queue 1. If the customer is admitted, then, on average, its service in server 0 will last $1/\mu_0$ time units, during which server 1 will complete μ_1/μ_0 customers. Upon departure from server 0, the customer will see $s_1 - \mu_1/\mu_0$ customers ahead. Therefore, the decision to reject a class 1 customer is optimal whenever

$$w_1 < \left(s_1 - \frac{\mu_1}{\mu_0}\right)\frac{h_1}{\mu_1}$$

or, equivalently,

$$s_1 > \frac{w_1\mu_1}{h_1} + \frac{\mu_1}{\mu_0}.$$

In a similar fashion, we obtain a condition for rejecting class 2 customers in the absence of class 1 customers. In general, the fuzzy set PVB for s_i, $i = 1, 2$, with membership grade 1.0 is fixed at

$$n_i = \frac{w_i \mu_i}{h_i} + \frac{\mu_i}{\mu_0}, \tag{7.2}$$

and in view of Figure 7.7a, the scaling factor for s_i is $\zeta_{si} = 9/n_i$.

7.4.3 A Numerical Example

The queuing network of Figure 7.6 has parameters $\mu_0 = 4$, $\mu_1 = 1$, $\mu_2 = 2$, $w_1 = w_2 = 3$, and $h_1 = h_2 = 1$.

We determine the optimal policy directly from the architecture of the fuzzy controller as follows:

1. We arbitrarily define two numbers S_1 and S_2 as the upper limits of s_1 and s_2, respectively, and set $s_1 = s_2 = 0$.
2. Using the current s_1 and s_2 as crisp inputs, we determine the decisions d_I and d_II via fuzzification, fuzzy inference, and defuzzification.
3. We plot the decision d in the two-dimensional space of s_1 and s_2.
4. If $s_2 < S_2$, we set $s_2 = s_2 + 1$ and go to (2); otherwise we set $s_2 = 0$ and go to (5).
5. If $s_1 < S_1$, we set $s_1 = s_1 + 1$ and go to (2); otherwise we stop.

From this algorithm, we obtain a policy of a switching form in the two-dimensional space of s_1 and s_2, as shown in Figure 7.8. The switching curve divides the state space into three areas. In the areas 1 and 2, server 0 selects a customer of class 1 and class 2, respectively, whereas in the area 0, this server is turned off (d_I = OFF).

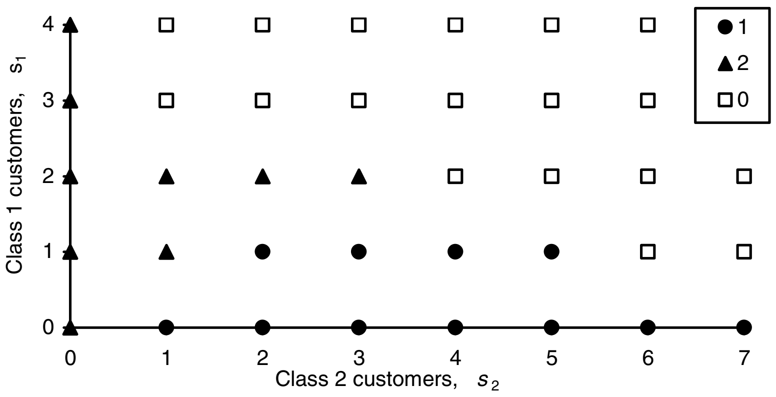


Figure 7.8. Decisions dictated by the fuzzy control policy. (J Intell Fuzzy Syst, Vol. 8, p. 41, by Runtong Zhang and Yannis A. Phillis. © 2000 by IOS Press. Used with permission.)

We elaborate step (2) of the algorithm. From Equation (7.2), we obtain $n_1 = (3 \times 1)/1 + 1/4 = 3.25$ and $n_2 = (3 \times 2)/1 + 2/4 = 6.5$, which correspond to PVB for s_1 and s_2, respectively. Hence, the scaling factors for s_1 and s_2 are $\zeta_{s1} = 2.77$ and $\zeta_{s2} = 1.38$, respectively. Let us assume that the current numbers of queuing customers of class 1 and 2 are $s_1 = 1$ and $s_2 = 3$, which should be scaled up to 2.77 and 4.14. We see from Figure 7.7a that s_1 corresponds to ZO with grade 0.38 and PS with grade 0.92 and s_2 corresponds to ZO with grade 0.09 and PS with grade 0.73. According to the fuzzy rule base (Table 7.4) and Mamdani implication, the fuzzy decisions are formulated as follows:

If s_1 is ZO with grade 0.38 and s_2 is ZO with grade 0.09, then d_{I} is ON and d_{II} is * both with grade 0.09.

If s_1 is ZO with grade 0.38 and s_2 is PS with grade 0.73, then d_{I} is ON and d_{II} is 1 both with grade 0.38.

If s_1 is PS with grade 0.92 and s_2 is ZO with grade 0.09, then d_{I} is ON and d_{II} is 2 both with grade 0.09.

If s_1 is PS with grade 0.92 and s_2 is PS with grade 0.73, then d_{I} is ON and d_{II} is * both with grade 0.73.

All fuzzy outputs d_{I} are ON; therefore, the decision d_{I} is ON, which means that server 0 should receive a customer. From Figure 7.7c, the peak values and heights of the fuzzy decisions d_{II} for the rules above are $e_1 = e_4 = 1.5$, $e_2 = 1$ and $e_3 = 2$, and $f_1 = f_3 = 0.09$, $f_2 = 0.38$, and $f_4 = 0.73$. By the height method of defuzzification, the crisp output $d_{\text{II}}*$ is given by

$$d_{\text{II}}* = \frac{\sum_{i=1}^{4} e_i f_i}{\sum_{i=1}^{4} f_i} = 1.388.$$

As $d_{\text{II}}* < 1.5$, the decision d_{II} is 1, which means that server 0 should receive a class 1 customer.

7.5 Three-Station Network with Controlled and Uncontrolled Arrivals

7.5.1 Problem Description

Now we add uncontrolled arrivals to stations 1 and 2 of the previous model. The resulting queuing system is shown in Figure 7.9. Again, we wish to decide the optimal admission and scheduling policies so that the average profit is maximized. For stability, we assume that the system has enough capacity to serve the unconditionally accepted customers in finite time; that is, $\lambda_i < \mu_i$, $i = 1, 2$.

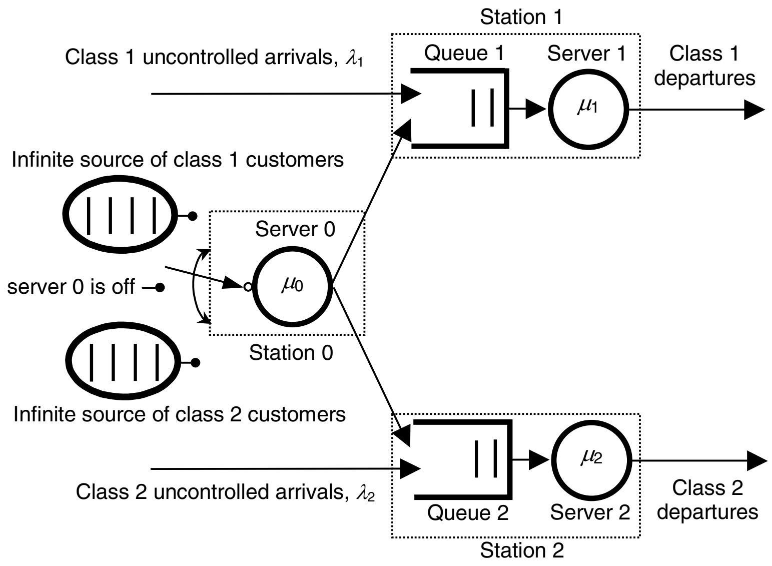

Station 1
Class 1 uncontrolled arrivals, λ_1
Queue 1
Server 1
μ_1
Class 1 departures
Infinite source of class 1 customers
Server 0
server 0 is off
μ_0
Station 0
Infinite source of class 2 customers
μ_2
Class 2 uncontrolled arrivals, λ_2
Queue 2
Server 2
Class 2 departures
Station 2

Figure 7.9. A queuing network with server scheduling and admission control.

7.5.2 Fuzzy Controller

As previously, the state of the system is described by (s_1, s_2), where $s_i = 0, 1, 2, \ldots,$ is the number of customers in queue i, $i = 1, 2$. The fuzzy rule base relies on the one shown in Table 7.4, which is applicable when $\lambda_1 = \lambda_2 = 0$. However, the arrival rates λ_1 and λ_2 also play a role here. Specifically, a higher arrival rate of uncontrolled class i customers weakens the decision to accept class i customers. Indeed, if the manager of the system knows that the length of queue i grows fast because of the uncontrolled arrivals, then, to avoid holding costs, the manager should accept fewer class i customers from the controlled infinite source.

A rule base that relies on the above arguments is shown in Table 7.5. The fuzzy controller uses four inputs, s_1, s_2, λ_1, and λ_2 and produces two outputs d_{I} and d_{II}, whose meaning is the same as in the previous model. For brevity, this table records only combinations of the input variables for which server 0 decides to receive a customer; that is, $d_{\mathrm{I}} = \mathrm{ON}$. All other combinations lead to $d_{\mathrm{I}} = \mathrm{OFF}$.

The membership functions for λ_i, $i = 1, 2$, are shown in Figure 7.4a, and those for s_i, d_{I}, and d_{II} are shown in Figure 7.7. We choose four linguistic values for each fuzzy input λ_i, and by the stability condition $\lambda_i < \mu_i$, we set PB for λ_i with membership grade 1.0 at the value μ_i, $i = 1, 2$. The universe of discourse for λ_i, $i = 1, 2$, is $[0, 6)$. Hence, the corresponding scaling factors are $\zeta_{\lambda_i} = 6/\mu_i$, $i = 1, 2$.

Table 7.5. Rule base.

Rules with $\lambda_2 = ZO$					*Rules with $\lambda_2 = PS$*					*Rules with $\lambda_2 = PM$*					*Rules with $\lambda_2 = PB$*				
s_1	s_2	λ_1	λ_2	d_{II}	s_1	s_2	λ_1	λ_2	d_{II}	s_1	s_2	λ_1	λ_2	d_{II}	s_1	s_2	λ_1	λ_2	d_{II}
ZO	ZO	ZO	ZO	*	ZO	ZO	ZO	PS	1	ZO	ZO	ZO	PM	1	ZO	ZO	ZO	PB	1
PS	ZO	ZO	ZO	2	PS	ZO	ZO	PS	*	PS	ZO	ZO	PM	1	PS	ZO	ZO	PB	1
PM	ZO	ZO	ZO	2	PM	ZO	ZO	PS	2	PM	ZO	ZO	PM	*	PM	ZO	ZO	PB	1
PB	ZO	ZO	ZO	2	PB	ZO	ZO	PS	2	PB	ZO	ZO	PM	2	PB	ZO	ZO	PB	*
PVB	ZO	ZO	ZO	2	PVB	ZO	ZO	PS	2	PVB	ZO	ZO	PM	2	PVB	ZO	ZO	PB	2
ZO	PS	ZO	ZO	1	ZO	PS	ZO	PS	1	ZO	PS	ZO	PM	1	ZO	PS	ZO	PB	1
PS	PS	ZO	ZO	*	PS	PS	ZO	PS	1	PS	PS	ZO	PM	1	PS	PS	ZO	PB	1
PM	PS	ZO	ZO	2	PM	PS	ZO	PS	*	PM	PS	ZO	PM	1	PM	PS	ZO	PB	1
ZO	PM	ZO	ZO	1	ZO	PM	ZO	PS	1	ZO	PM	ZO	PM	1	ZO	PM	ZO	PB	1
PS	PM	ZO	ZO	1	PS	PM	ZO	PS	1	PS	PM	ZO	PM	1	PS	PM	ZO	PB	1
ZO	PB	ZO	ZO	1	ZO	PB	ZO	PS	1	ZO	PB	ZO	PM	1	ZO	PB	ZO	PB	1
ZO	PVB	ZO	ZO	1	ZO	ZO	PS	PS	*	ZO	PVB	ZO	PM	1	ZO	ZO	PS	PB	1
ZO	ZO	PS	ZO	2	PS	ZO	PS	PS	2	ZO	ZO	PS	PM	1	PS	ZO	PS	PB	1
PS	ZO	PS	ZO	2	PM	ZO	PS	PS	2	PS	ZO	PS	PM	*	PM	ZO	PS	PB	*
PM	ZO	PS	ZO	2	PB	ZO	PS	PS	2	PM	ZO	PS	PM	2	PB	ZO	PS	PB	2
PB	ZO	PS	ZO	2	PVB	ZO	PS	PS	2	PB	ZO	PS	PM	2	PVB	ZO	PS	PB	2
PVB	ZO	PS	ZO	2	ZO	PS	PS	PS	1	PVB	ZO	PS	PM	2	ZO	PS	PS	PB	1
ZO	PS	PS	ZO	*	PS	PS	PS	PS	*	ZO	PS	PS	PM	1	PS	PS	PS	PB	1
PS	PS	PS	ZO	2	ZO	PM	PS	PS	1	PS	PS	PS	PM	1	ZO	PM	PS	PB	1
PM	PS	PS	ZO	2	ZO	PB	PS	PS	1	ZO	PM	PS	PM	1	ZO	PB	PS	PB	1
ZO	PM	PS	ZO	1	ZO	PVB	PS	PS	1	ZO	PB	PS	PM	1	ZO	PVB	PS	PB	1
PS	PM	PS	ZO	*	ZO	ZO	PM	PS	2	ZO	PVB	PS	PM	1	ZO	ZO	PM	PB	1
ZO	PB	PS	ZO	1	PS	ZO	PM	PS	2	ZO	ZO	PM	PM	*	PS	ZO	PM	PB	*
ZO	PVB	PS	ZO	1	PM	ZO	PM	PS	2	PS	ZO	PM	PM	2	PM	ZO	PM	PB	2
ZO	ZO	PM	ZO	2	PB	ZO	PM	PS	2	PM	ZO	PM	PM	2	PB	ZO	PM	PB	2
PS	ZO	PM	ZO	2	PVB	ZO	PM	PS	2	PB	ZO	PM	PM	2	PVB	ZO	PM	PB	2
PM	ZO	PM	ZO	2	ZO	PS	PM	PS	*	PVB	ZO	PM	PM	2	ZO	PS	PM	PB	1
PB	ZO	PM	ZO	2	PS	PS	PM	PS	2	ZO	PS	PM	PM	1	ZO	PM	PM	PB	1
PVB	ZO	PM	ZO	2	ZO	PM	PM	PS	1	ZO	PM	PM	PM	1	ZO	PB	PM	PB	1
ZO	PS	PM	ZO	2	ZO	PB	PM	PS	1	ZO	PB	PM	PM	1	ZO	PVB	PM	PB	1
PS	PS	PM	ZO	2	ZO	PVB	PM	PS	1	ZO	PVB	PM	PM	1	ZO	ZO	PB	PB	*
PM	PS	PM	ZO	2	ZO	ZO	PB	PS	2	ZO	ZO	PB	PM	2	PS	ZO	PB	PB	2
ZO	PM	PM	ZO	*	PS	ZO	PB	PS	2	PS	ZO	PB	PM	2	PM	ZO	PB	PB	2
PS	PM	PM	ZO	2	PM	ZO	PB	PS	2	PM	ZO	PB	PM	2	PB	ZO	PB	PB	2
ZO	PB	PM	ZO	1	PB	ZO	PB	PS	2	PB	ZO	PB	PM	2	PVB	ZO	PB	PB	2
ZO	PVB	PM	ZO	1	PVB	ZO	PB	PS	2	PVB	ZO	PB	PM	2	ZO	PS	PB	PB	1
ZO	ZO	PB	ZO	2	ZO	PS	PB	PS	2	ZO	PS	PB	PM	*	ZO	PM	PB	PB	1
PS	ZO	PB	ZO	2	PS	PS	PB	PS	2	ZO	PM	PB	PM	1	ZO	PB	PB	PB	1
PM	ZO	PB	ZO	2	ZO	PM	PB	PS	*	ZO	PB	PB	PM	1	ZO	PVB	PB	PB	1
PB	ZO	PB	ZO	2	ZO	PB	PB	PS	1	ZO	PVB	PB	PM	1					
PVB	ZO	PB	ZO	2	ZO	PVB	PB	PS	1										
ZO	PS	PB	ZO	2															
PS	PS	PB	ZO	2															
PM	PS	PB	ZO	2															
ZO	PM	PB	ZO	2															
PS	PM	PB	ZO	2															
ZO	PB	PB	ZO	*															
ZO	PVB	PB	ZO	1															

7.5.3 A Numerical Example

We examine the queuing network of Figure 7.9 with parameters arrival rates $\lambda_1 = \lambda_2 = 0.1$, service rates $\mu_3 = 4$, $\mu_1 = 1$, and $\mu_2 = 2$, rewards $w_1 = w_2 = 3$, and holding costs per customer per time unit $h_1 = h_2 = 1$.

The scaling factors for the four inputs of the fuzzy controller are $\zeta_{s1} = 2.77$ and $\zeta_{s2} = 1.38$, $\zeta_{\lambda 1} = 6/\mu_1 = 6$, and $\zeta_{\lambda 2} = 6/\mu_2 = 3$. By the algorithm of Section 7.4.3, we obtain the fuzzy control policy as shown in Figure 7.8, which turns out to be an exact copy of the monotonic switching policy we found in the previous example.

In the absence of an analytical solution to this problem, we compare the fuzzy policy with other admission and scheduling protocols that are often used in practice. We consider three admission policies:

- *Case 1*: Server 0 is always off.
- *Case 2*: Server 0 admits one customer per time unit.
- *Case 3*: Server 0 admits two customers per time unit.

For each of Cases 2 or 3 we examine three scheduling policies:

- *Least Expected Work*: Server 0 receives a customer from the class with the smallest work in queue i, s_i/μ_i, $i = 1, 2$.
- *Shortest Queue*: Server 0 receives a customer from the class with the smallest s_i, $i = 1, 2$.
- *Round Robin*: Server 0 receives the first customer from class 1, the second from class 2, and so on.

The simulation results for each combination of admission and scheduling policies are summarized in Table 7.6.

Table 7.6. Mean profit rate for each admission and scheduling policy. (IEEE T Fuzzy Syst, Vol. 9, p. 311, by Runtong Zhang and Yannis A. Phillis. © 2001 by IEEE. Used with permission.)

	Case 2	*Case 3*
Least Expected Work	1.66	0.90
Shortest Queue	1.58	0.67
Round Robin	1.07	–
Case 1	–0.16	
Fuzzy Control	1.91	

We see that the fuzzy controller achieves the highest mean profit rate (1.91). Also, the Least Expected Work is better than the Shortest Queue scheduling policy, which in turn is better than the Round Robin controller for all admission policies.

8 Applications of Fuzzy Queuing Control to the Internet

8.1 Introduction

In this chapter, we present a few applications of fuzzy queuing control in communication networks with emphasis on the Internet. We begin with a brief description of the Internet. The book by Kurose and Ross (2003) provides a detailed description of computer networks and the Internet.

The Internet is a worldwide communication network that interconnects millions of computers and other devices that are called *end systems* or *hosts* and can receive, store, or transmit information.

Communication links are physical media through which data are transferred. Coaxial cables, optical fibers, and radio waves are examples of communication links with different transmission rates or *bandwidths*, measured in bits per second.

Routers or *switches* are used to forward data from one link to another. A router has several *ports* where links are attached and several *buffers* where data are stored temporarily before they are transmitted to other links. Each port of the router has one link and one buffer. Thus, routers act as intermediate queuing nodes and allow multiple end systems to share a link or a path of consecutive links and routers at the same time. This is achieved by a technique known as *packet switching*.

When a host sends a message to another host, the message is divided into smaller pieces of data, the *packets*. Each packet is separately numbered, and it includes the Internet address of the destination host and a segment of the original message.

Hosts and routers use a specific format for addressing and forwarding packets, called Internet Protocol (IP). IP is a connectionless protocol. Packets can travel through different routes across the network, and each traveling packet is treated separately from the other packets of the same message. Hence, packets arrive at their destination in a different order from the order they were sent. When the last packet arrives, the original message is retrieved by putting all packets back in the right order. This is done using the Transmission Control Protocol (TCP).

An important indicator of Internet performance is packet *delay*. Delays in communication networks happen in various ways. A router spends some time reading a packet's header to determine the communication link where the packet should be sent. This results in *processing delay*. Then the packet is sent to a buffer where it waits to be transmitted through the corresponding link and is subject to *queuing delay*. When the link is available, it starts transmitting the bits of the packet one by one. The time until the last bit of the packet enters the physical medium of the link is equal to the ratio of the number of bits of the packet to the bandwidth of the link. This time is called *transmission delay*. Finally, the total time the last bit of the packet spends inside the physical medium of the link is called *propagation delay* and is equal to the ratio of the length of the link to the travel speed, typically close to the speed of light.

Another important performance measure is packet *loss*. Packets are lost or dropped when they attempt to enter a full buffer in a router. Lost packets can be retransmitted, but they incur further delay, and because they bind additional resources, they may cause network congestion.

Queuing delays and the fraction of lost packets increase with the traffic intensity when too many hosts and routers send data too frequently.

Quality of Service (QoS) refers to a collection of methods for controlling the way in which users share the available network resources in order to deal with the effects of congestion. One such method is the Differentiated Services mechanism (*DiffServ*). DiffServ is based on the idea that all the packets can be grouped into a small number of classes. Packets with similar performance requirements belong to the same class. For example, e-mail messages are *elastic* applications because they have no real-time requirements (a few minutes of delay in the delivery of a message is usually unimportant). In contrast, commercial applications such as video on demand, medical diagnostics, and web-based teaching are *critical* because their providers must guarantee some minimum performance. To provide QoS support, DiffServ allows for a sort of “controlled unfairness” in the use of resources in favor of critical applications.

Up until now, several DiffServ schemes have been proposed that, among others, handle drop and delay priorities. Drop priority is associated with the risk of discarding a packet. If the cost of packet loss is high, then the packet should receive preference service; otherwise normal service is selected. Delay priority is associated with the total delay of a packet. Real-time applications have a high delay priority over non-real-time ones. These two priorities raise important optimization issues for the Internet, but their relationship has not been investigated by researchers.

An extensive survey on applications of queuing theory and dynamic programming to communication networks is presented by Altman (2000).

In this chapter, we apply fuzzy logic to control congestion in the Internet. We examine three problems:

- Scheduling for packet delay and loss balancing
- Drop-based congestion control
- Routing

We extend previous results to build the fuzzy controllers for the first two problems and present a new approach for solving the third problem.

8.2 Drop and Delay Balancing in the Differentiated Services

8.2.1 Problem Description

We consider a communication network operating under a simple DiffServ scheme that provides four types of service. Depending on their service requirements, the packets that travel through the network are encoded as follows:

11: real-time flow (1) with preference service selection (1)
10: real-time flow (1) with normal service selection (0)
01: non-real-time flow (0) with preference service selection (1)
00: non-real-time flow (0) with normal service selection (0)

Each router in the network has four buffers denoted 11, 10, 01, and 00 where packets wait before they can be transmitted through a link that is attached to the router. An incoming packet is scheduled to a buffer according to its type of service, but this packet is discarded if its buffer is full. All four buffers operate on a First In First Out (FIFO) discipline. To satisfy real-time requirements, all packets in the real-time buffer shall be transmitted before any packets in the non-real-time buffer. For this reason, the four types of service 11, 10, 01 and 00 are ordered on a preference scale from 3 to 0. In addition, to achieve small delays and delay variations, the real-time buffer is kept relatively small, whereas the non-real-time buffer could be larger.

An incoming packet passes first through a comparison unit. If the corresponding buffer is not full, the packet is accepted; otherwise it is discarded. Given the destination buffer of a packet, the state of the lower priority buffers is recorded. If these buffers are empty, the packet goes directly to its own buffer. If, on the other hand, subsequent buffers are not empty, the current packet will be given priority over those in the nonempty buffers. Service type 11 is always given priority, and service type 01 is given priority over 00. However, priorities between 10 and 01 are not obvious. To resolve this question, we isolate buffers 10 and 01 by examining a related queuing system with two arrival streams as shown in Figure 8.1.

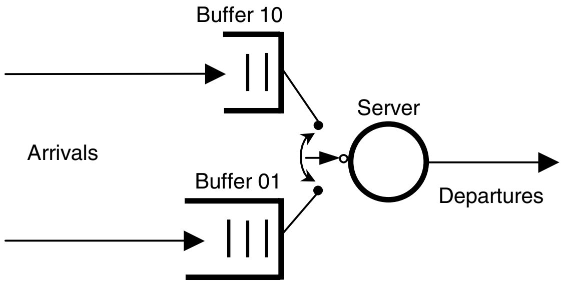


Figure 8.1. Controlled system.

Recall that buffer 10 contains real-time packets with normal service and that 01 contains non-real-time packets with preference service. Both buffers have limited capacities B_{10} and B_{01}, but $B_{10} \leq B_{01}$ as explained previously. All packets are subject to a FIFO discipline. The problem is to determine a queueing policy that dynamically accepts packets from either buffer so as to satisfy a given optimality rule.

8.2.2 Fuzzy Controller

The state of the system can be described by (s_{10}, s_{01}), where $s_{10} = 0, 1, \ldots, B_{10}$, and $s_{01} = 0, 1, \ldots, B_{01}$ are the numbers of packets in buffers 10 and 01, respectively. The number of packets in queue is $x = s_{10} + s_{01} = 0, 1, \ldots, B_{10} + B_{01}$. Decisions are made at each packet arrival and departure.

If a departure channel (link) is not available, a packet at this node cannot be sent off. Hence, the decision epochs coincide with the times when a packet arrives in an empty buffer with a departure channel available or leaves the node with packets in the buffers.

We now directly write the following obvious crisp rules: (1) *If the state* $x = s_{10} = s_{01} = 0$, *that is, if there are no packets in the queues, then no packets are sent off*; (2) *If* $x = 1$, *that is, if there is one and only one packet in the system, then this packet is sent off*. Henceforth we will only focus on scheduling waiting packets when both buffers are not empty and a departure channel is available.

We choose as fuzzy inputs the numbers of packets s_{10} and s_{01} in buffers 10 and 01, with linguistic values ZO, PS, PM, and PB. The fuzzy output is the decision $d = 01$ or 10, which indicates that a packet from buffer 10 or 01 is selected to be sent.

The rule base relies on the following observations. When there are many packets in the real-time buffer 10 while there are relatively few packets in the non-real-time one, to avoid a long delay, a real-time packet is sent off; otherwise a non-real-time packet is sent off. When both buffers have similar occupancies, we should check if the buffers are approaching full usage. If both queues are short, attention should be paid to delay priorities and a real-time packet should be sent off; otherwise a non-real-time packet is sent off, because now the drop priority is more important. In other words, whenever non-real-time packets with preference do not risk being discarded, the delay priority of real-time packets is more important. If the traffic is heavy and both buffers are full, it is possible that real-time packets be delayed longer than an allowed limit. Such a situation is unusual, but it does not affect the implementation of the fuzzy controller.

From these remarks, we develop the rule base shown in Table 8.1, where * for the fuzzy output d means that nothing is selected.

Table 8.1. Rule base.

Rules 1–8			*Rules 9–16*		
s_{10}	s_{01}	d	s_{10}	s_{01}	d
ZO	ZO	*	ZO	PM	01
PS	ZO	10	PS	PM	01
PM	ZO	10	PM	PM	10
PB	ZO	10	PB	PM	10
ZO	PS	01	ZO	PB	01
PS	PS	10	PS	PB	01
PM	PS	10	PM	PB	01
PB	PS	10	PB	PB	01

The universe of discourse for the fuzzy inputs s_{10} and s_{01} is [0, 6] and for the output d is [1, 2]. The membership functions for the fuzzy inputs s_i, $i = 01, 10$, and the decision d are devised as in Chapter 7 and are shown in Figure 8.2.

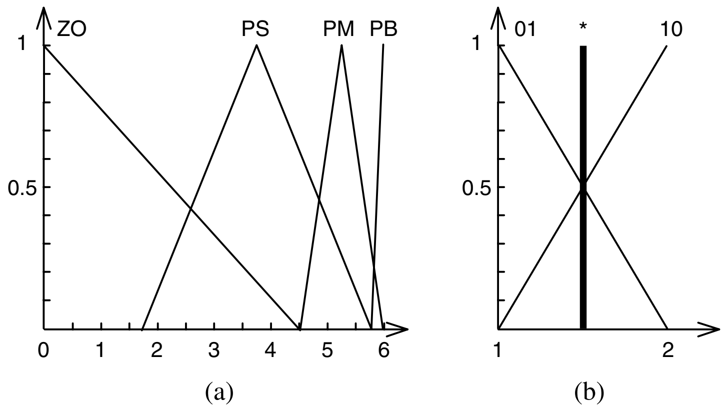


Figure 8.2. Membership functions: (a) s_{10}, s_{01} and (b) d.

As both buffers are finite, we set PB for the fuzzy inputs s_{10} and s_{01} with membership grade 1.0 at B_{10} and B_{01}, respectively. The scaling factors then become $\zeta_{si} = 6/B_i$, $i = 10, 01$.

8.2.3 A Numerical Example

The controlled system is shown in Figure 8.1 with buffer capacities $B_{10} = 5$ and $B_{01} = 50$. We determine the queueing management policy using fuzzy logic. The algorithm is outlined as follows:

1. The scaling factors for both fuzzy inputs s_{01} and s_{10} are first determined.
2. We start from an initial state $s_{10} = s_{01} = 0$.
3. Using the current s_{10} and s_{01} as crisp inputs, we determine the decision d via fuzzification, fuzzy inference based on the rule base in Table 8.1, and defuzzification.
4. We plot the decision d in the two-dimensional space of s_{10} and s_{01}.

5. If $s_{01} < B_{01}$, we set $s_{01} = s_{01} + 1$ and go to (3); otherwise we set $s_{01} = 0$ and go to (6).
6. If $s_{10} < B_{10}$, we set $s_{10} = s_{10} + 1$ and go to (3); otherwise we stop.

For example, let us assume that the current numbers of packets in queue are $s_{10} = 2$ and $s_{01} = 30$. These values are scaled to $2 \times \zeta_{s10} = 2 \times 6/5 = 2.4$ and $30 \times \zeta_{s10} = 30 \times 6/50 = 3.6$. From Figure 8.2a, $s_{10} = 2.4$ corresponds to ZO with grade 0.47 and PS with grade 0.35, and $s_{01} = 3.6$ corresponds to ZO with grade 0.20 and PS with grade 0.93. According to the fuzzy rule base and Mamdani implication, the fuzzy decisions d are formulated as follows:

If s_{10} is ZO with grade 0.47 and s_{01} is ZO with grade 0.20, then d is * with grade 0.20.
If s_{10} is ZO with grade 0.47 and s_{01} is PS with grade 0.93, then d is 01 with grade 0.47.
If s_{10} is PS with grade 0.35 and s_{01} is ZO with grade 0.20, then d is 10 with grade 0.20.
If s_{10} is PS with grade 0.35 and s_{01} is PS with grade 0.93, then d is 10 with grade 0.35.

From Figure 8.2b, the peak values of the decision d are $e_1 = 1.5$, $e_2 = 1$, and $e_3 = e_4 = 2$, and the heights of the decision are $f_1 = 0.20$, $f_2 = 0.47$, $f_3 = 0.20$, and $f_4 = 0.35$. By the height method of defuzzification, the crisp output d^* is

$$d^* = \frac{\sum_{i=1}^{4} e_i f_i}{\sum_{i=1}^{4} f_i} = \frac{1.87}{1.22} = 1.53.$$

As $d^* > 1.5$, the decision d is 2, which means that a packet from buffer 10 is sent off.

Following this algorithm, we obtain the policy of Figure 8.3. This policy is defined by a switching curve, where “10” packets are transmitted if the state falls in the upper area; otherwise “01” packets are transmitted.

We see from this figure that packets from buffer 10 are sent off more frequently than packets from 01. This policy gives priority to real-time packets as expected when traffic is below capacity. When traffic becomes heavy, however, priorities change. Near the origin, the switching curve appears to be a straight line but changes sharply later. In the absence of such change, the fuzzy policy would be similar to a strict queue policy.

High-performance communication networks require control mechanisms as simple and efficient as possible. Hence, the queue management policy for a given node can be calculated off-line and stored in a table format.

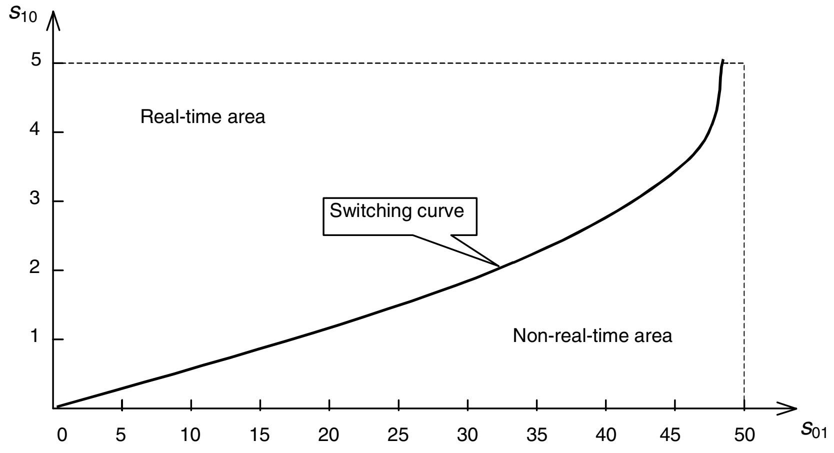


Figure 8.3. The queue management policy.

A few important observations ought to be made here to justify the fuzzy approach.

1. There are no analytical control algorithms for this system in the literature to compare our solution with, and there is no universally acceptable performance measure to be optimized.

2. A strict policy would be linear or piecewise linear, but no mathematical means exist to set the slope.

3. The fuzzy solution resolves the fairness issue for all packets, and when packets wait too long in a buffer, it reverses its policy.

4. This is the first problem in a multitude of other Internet problems that, because of their complexity, could be handled with fuzzy logic. High-performance communication networks require simple and fast algorithms that will guarantee proper network functioning. The proposed algorithm satisfies these requirements and has been tested in actual practice.

8.2.4 Performance Evaluation

Consider the network of Figure 8.4 in which S_i, $i = 1, \dots, n$, are data transmitters (typically PCs), E_1 and E_2 are edge nodes or routers, C_1 and C_2 are core nodes or intermediate routers, and D_i, $i = 1, \dots, n$, are data destinations such as PCs. All the links of the network have a speed of 10 Mbps and a delay of 1 ms, with the exception of the link between C_2 and E_2, which has a speed of 2 Mbps. This is the bottleneck of the network. The parameters of the network are those of the previous example 8.2.3.

Each node uses the standard Internet transmission control protocol with exponential on/off times whose mean is 500 ms. The simulation time corresponds to 60 s of operation. In each router, buffer 10 has a capacity of 5 packets and buffer 01 has a capacity of 50 packets.

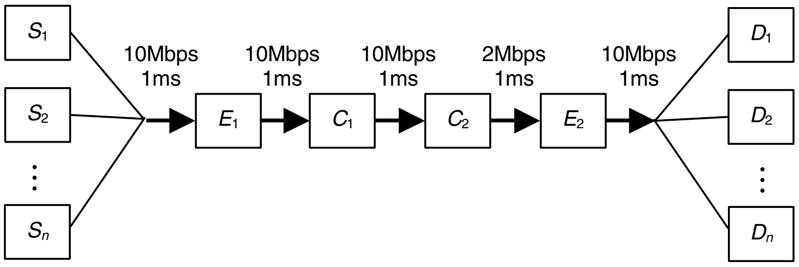


Figure 8.4. Network topology.

We examine two cases with light and heavy load.

Light Load

Let $n = 15$. S_1, …, S_7 send "10" packets at a rate of 100 kbps each or a total rate of 700 kbps, whereas S_8, …, S_{15} send "01" packets at a total rate of 800 kbps. Two policies are tested, fuzzy and crisp. When the crisp policy is employed, the node serves non-real-time packets only after all real-time packets have been served.

For the data of this case, no link is congested and both policies produce the same results summarized in Table 8.2. This is so because the fuzzy policy takes over only when the load is heavy. The throughput for both policies is 1454 kbps.

Table 8.2. Drop statistics under light load.

Buffer	*Packets sent*	*Packets forwarded*	*Drops*
10	23,892	23,892	0
01	28,386	28,350	36
Total	52,278	52,242	36

Heavy Load

Now let $n = 20$. All S_i have a sending rate of 100 kbps each. "10" packets are transmitted by S_1, …, S_{10} at a total rate of 1000 kbps, and "01" packets are transmitted by S_{11}, …, S_{20} at the same total rate. The link between C_2 and E_2 is congested. The results are now different as shown in Table 8.3.

Table 8.3. Drop statistics under heavy load.

Policy	*Buffer*	*Packets sent*	*Packets forwarded*	*Drop s*	*Throughput* [kbps]
	10	33,200	32,940	260	
	01	35,680	32,455	3225	
Fuzzy	*Total*	68,880	65,395	3485	1828
	10	33,691	33,691	0	
	01	35,680	32,165	3515	
Crisp	*Total*	69,371	65,856	3515	1834

The fuzzy policy provides a fairer treatment for packets with preference service because it does not allow packets to stay too long in a buffer. The throughput, however, is about the same for both policies.

8.3 Congestion Control in the Differentiated Services

8.3.1 Problem Description

When the network becomes congested, the TCP flow control mechanism slows down the sending rates to avoid packet losses. In this section, we describe a drop-based congestion control method for DiffServ networks. First, the controller denies admission to packets having low service preferences, and then, if necessary, it starts dropping packets with a higher service preference so as to maximize a certain performance measure.

We consider a single-server queuing system with two classes of arriving packets as shown in Figure 8.5.

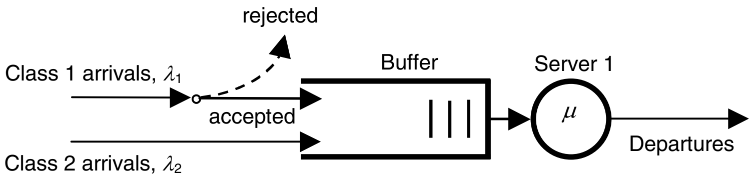


Figure 8.5. A single-server queue with arrival control.

Two independent Poisson streams of packets, called class 1 and class 2 packets, arrive at the buffer with constant rates λ_1 and λ_2. The buffer has unlimited capacity, and the order of service is irrelevant. The buffer is served by one exponential server with service rate μ. Class 1 packets upon arrival are either permitted or prohibited to enter the buffer, whereas class 2 packets enter the buffer without restriction. The system receives a fixed reward w for each accepted packet and pays a holding cost h per packet per unit time in the system. We wish to decide the optimal admission policy so that the average profit (reward minus cost) is maximized.

This is a special case of the problem in Section 6.4 and will be treated similarly.

8.3.2 Fuzzy Controller

The decision epochs at which the arriving packets are controlled coincide with the arrival times of class 1 packets. The state of the system at the decision epochs is described by the total number x of class 1 and class 2 packets in the system including the one in service (if any), $x = 0, 1, \ldots$.

To avoid the trivial situation where an arriving class 1 packet is immediately denied entrance even when the system is empty, we assume that

$$w > \frac{h}{\mu}.$$

Under this condition, it is always beneficial to admit a class 1 packet when the system is empty. Hence, we write a crisp rule for $x = 0$: *If there are no packets in the system, then an arriving class 1 packet is admitted immediately.*

Next, we consider cases where $x \geq 1$. As only class 2 packet arrivals are uncontrolled, the system is stable if

$$\lambda_2 < \mu.$$

When the number of packets in queue is large or an arrival rate is high, then we should reject class 1 customers to avoid queuing delays and costs. We therefore choose the number of packets in the buffer $s = x - 1$, $s = 0, 1, \ldots$, and the arrival rates $\lambda_1 \in [0, \infty)$ and $\lambda_2 \in [0, \mu)$ as fuzzy inputs with four linguistic values ZO, PS, PM, and PB. The decision $d = 1, 0$ to admit an arriving class 1 packet is the fuzzy output with linguistic values YES and NO.

The establishment of the fuzzy rule base relies on the following arguments:

1. As the number of packets in queue increases, the holding cost becomes greater than the reward and, therefore, admission should be denied.

2. A higher arrival rate of class 1 packets weakens the decision *d is YES*. Indeed, if the manager of the system knew that packets were arriving at a very high rate, he would be unwilling to accept new packets when the server is busy because they merely incur a holding cost.

3. Similarly, λ_2 affects negatively the decision to admit an arriving class 1 packet, in anticipation of future uncontrolled class 2 arrivals.

The rule base and the membership functions for the inputs and output of the fuzzy controller are identical to those of Section 6.4, but we present them here for convenience.

For brevity, Table 8.4 records only the rules whose output is YES. All other combinations of s, λ_1, and λ_2 lead to the decision NO. The membership functions for λ_1, λ_2, s, and d are shown in Figure 8.6. The threshold values above which the fuzzy inputs s, λ_1, and λ_2 are declared PB with membership grade 1.0 are computed using the ideas of Chapter 6.

Table 8.4. Rules whose output is YES.

s	λ_1	λ_2	d	s	λ_1	λ_2	d
ZO	ZO	ZO	YES	ZO	PM	ZO	YES
ZO	ZO	PS	YES	PS	ZO	ZO	YES
ZO	ZO	PM	YES	PS	ZO	PS	YES
ZO	PS	ZO	YES	PS	PS	ZO	YES
ZO	PS	PS	YES	PM	ZO	ZO	YES

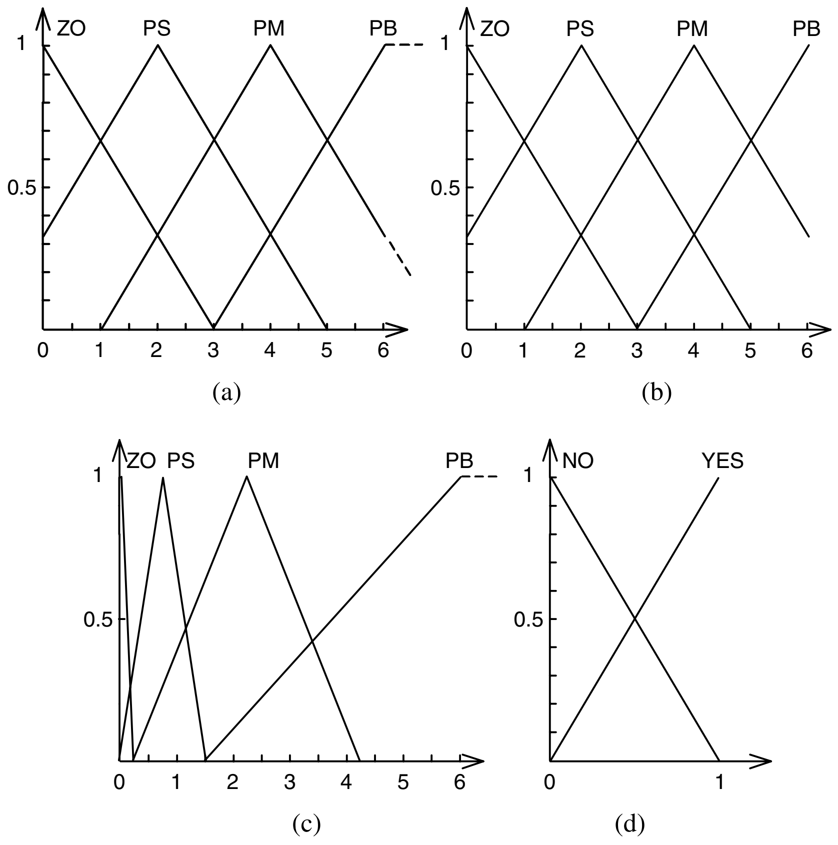


Figure 8.6. Membership functions: (a) λ_1; (b) λ_2; (c) s; (d) d.

It follows from the rule *if s is PB and λ_1 is ZO and λ_2 is ZO, then d is NO* and the discussion of Section 6.2 that PB for s with membership grade 1.0 is fixed at

$$x_{\lambda_1=\lambda_2=0}=\frac{w\mu}{h}.$$

Regarding λ_1, we consider the rule *if s is ZO and λ_1 is PB and λ_2 is ZO, then d is NO*. As in Section 6.2, we find the threshold value

$$\lambda_{1,\,s=\lambda_2=0}=\frac{\mu(w\mu-2h)}{h},$$

which is declared PB with grade 1. Finally, because λ_2 is bounded from above by μ, we set

$$\lambda_{2,\,s=\lambda_1=0}=\mu.$$

8.3.3 Performance Evaluation

We simulated the network shown in Figure 8.5 with parameters $\lambda_1 = 0.25$, $\lambda_2 = 0.05$, $\mu = 0.30$, $w = 50$, and $h = 2$. For this system, the fuzzy control policy produces the same decisions as those for the M/M/2 system with service rate $\mu = 0.15$, which was studied in Section 6.4.3. This policy imposes a threshold $x^* = 3$ on the total number of customers in the system.

By modeling the system as a Markov chain, we compute the mean profit rate $J(x^*)$ under different threshold values x^*. The results are $J(2) = 12.3$, $J(3) = 7.1$, $J(4) = 3.9$, and $J(5) = 1.5$. We see that the fuzzy control policy is very close to the optimal one.

8.4 Quality of Service Routing for Next-Generation Networks

8.4.1 Problem Description

As we have seen in Section 8.1, the current Internet is connectionless, that is, data packets of a message may follow different paths to the destination. However, this architecture does not guarantee bounded delays and small loss probabilities. The next generation of high-speed networks is likely to be *circuit-switched* for real-time traffic rather than packet-switched (Chen and Nahrstedt 1998). This means that before a host can send the first packet to another host, a connection (circuit) must be established between the two hosts and this connection will not be changed afterward. A circuit is a path from the sender to the destination that passes through a number of intermediate routers and links.

Routing is the problem of selecting a path with sufficient resources to satisfy the QoS requirements of a particular connection, using information about the state of the network. The state of a path is typically determined by the states of its routers and links, including the available (free) bandwidth, buffer space, and queuing and propagation delays. Routing problems are distinguished as unicast or multicast, depending on the number of destination nodes. A review of routing algorithms can be found in Chen and Nahrstedt (1998).

In this section, we shall develop a fuzzy routing algorithm.

Consider the network shown in Figure 8.7. For simplicity, it is assumed that the links have the same bandwidth and the same lengths. These assumptions are logical because, typically, the propagation delays through communication links are much shorter than the queuing delays at the switching nodes. Each node has an incoming packet buffer with a maximum capacity of B. The nodes 1, 5, 6, 7, 8, 9, and 10 at the perimeter of the network act as traffic generating, destination, or switching nodes. Nodes 2, 3, and 4 are pure switching nodes.

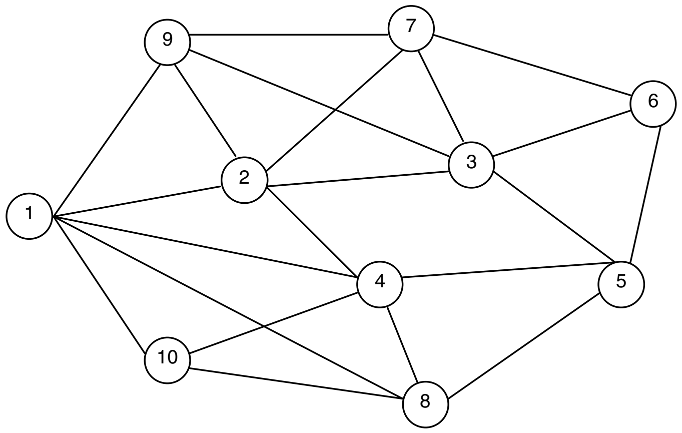


Figure 8.7. A communication network.

According to the QoS requirements, a path should be determined before a message is sent off at its generating node and the chosen route will not be changed afterward. The problem is to determine the optimal routing policy for each traffic flow at its generating node based on the state of the system so as to minimize packet delays, packet losses, and connection rejections at the generating nodes.

8.4.2 Fuzzy Routing

For any pair of source and destination nodes, the state of each eligible path is described by the numbers of queuing packets in every buffer on the path. The state changes whenever an arrival or departure at any nodes along the given path occurs. The decision epochs coincide with the times when a new session or traffic flow is generated and sent to the network.

To solve this problem, we apply fuzzy logic to determine path ratings, based on the previous criteria, for all eligible paths between the source and destination nodes. The path with the highest rating is then chosen to route the traffic flow. A connection is only rejected if all the buffers on the chosen path are currently full; otherwise all the packets of the session are routed over the chosen path. Whenever a packet arrives at a full buffer, the packet is dropped.

Consider a path with s links and corresponding buffers $i = 1, 2, \ldots, s$. The utilization ρ_i of each buffer i on the path is defined as the fraction of buffer space occupied. If n_i is the current queue size in buffer i and B its capacity, then

$$\rho_i = \frac{n_i}{B}. \tag{8.1}$$

Next, we take the sum of these utilization measures and generate a weighting measure λ_i for each buffer i

$$\lambda_i = \frac{\rho_i}{\sum_{j=1}^{s} \rho_j} . \tag{8.2}$$

We define path utilization ρ as the weighted sum of buffer occupancies along the path

$$\rho = \sum_{i=1}^{s} \lambda_i n_i . \tag{8.3}$$

For each path, we choose the number of routers s on the path and the path utilization ρ as fuzzy inputs and the path rating r as the fuzzy output of the algorithm. All variables are represented by four linguistic values ZO, PS, PM, and PB. A path's rating is a decreasing function of s and ρ. The fuzzy rule base is shown in Table 8.5.

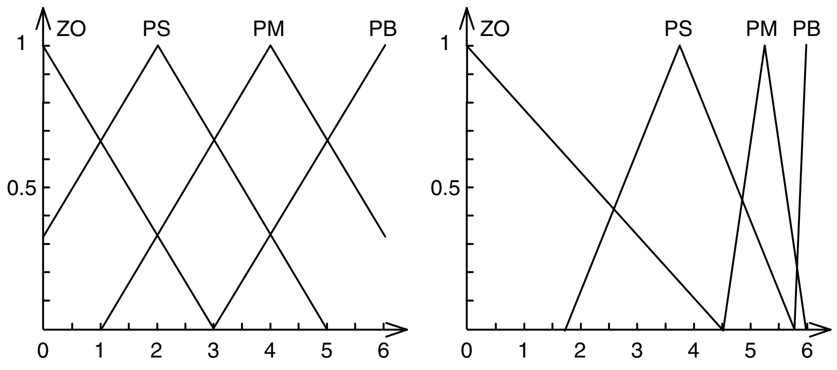
Table 8.5. Rule base.

Rules 1–8			*Rules 9–16*		
s	ρ	r	s	ρ	r
ZO	ZO	PB	ZO	PM	PS
PS	ZO	PM	PS	PM	ZO
PM	ZO	PS	PM	PM	ZO
PB	ZO	ZO	PB	PM	ZO
ZO	PS	PM	ZO	PB	ZO
PS	PS	PS	PS	PB	ZO
PM	PS	ZO	PM	PB	ZO
PB	PS	ZO	PB	PB	ZO

The universe of discourse for the fuzzy variables is [0, 6]. The membership functions for s and r are shown in Figure 8.8a. We observe that ρ is a global measure of buffer occupancies and that the total delay is a function of the queue size n_i; thus, it increases as the sequence $1 + 2 \ldots + n_i = n_i(n_i + 1)/2 = 1, 3, 6, \ldots$. From this reasoning, we choose the fuzzy membership functions for ρ as shown in Figure 8.8b.

The fuzzy routing algorithm is outlined as follows:

1. All eligible paths between source and destination nodes are recorded at each decision epoch. The corresponding state information is also recorded.
2. For each path, we calculate crisp values for s and ρ using Equations (8.1)–(8.3). We compute the rating r via fuzzification, fuzzy inference, and defuzzification.
3. The path with the highest rating is chosen to route the traffic flow.



(a) (b)

Figure 8.8. Membership functions: (a) s and r; (b) ρ.

8.4.3 Performance Evaluation

Consider the communication network shown in Figure 8.7. The links between the nodes are all 2 km in length, and the bandwidth of the links is 100 Mbps. Each node has a queue capacity of 50 packets. Arrivals or connection attempts at each source node occur in a Poisson manner. A connection is rejected only if all buffers on the chosen path to route the call are full.

The fuzzy routing scheme is tested against three other routing algorithms: a *fixed directory routing* algorithm, a *shortest path routing* algorithm, and a crisp version of the fuzzy algorithm, which is called *crisp routing*. The fixed directory routing algorithm gives priority to paths with the smallest number of routers.

The shortest path routing algorithm selects the path with the shortest delay. A delay estimate is the sum of the times-to-service all packets currently at each router on the path. In case of a tie, the path with the smallest number of routers is chosen.

Crisp routing selects the path with the smallest utilization ρ computed from Equations (8.1)–(8.3). In the case of a tie, the path with the smallest number of routers is chosen.

The performance of each routing algorithm was evaluated using simulation. The mean interarrival times ranged between 0.5 and 1.0 second. For each interarrival time, the average performance was computed from ten simulation runs, each spanning 300 seconds. During each run, common sequences of random numbers were used so that all routing algorithms could be compared in a fair fashion.

Figure 8.9 illustrates that the fuzzy routing algorithm rejects a smaller percentage of connections than the other three routing algorithms. Recall that the only reason for which connections are rejected is that all buffers on the route chosen are full.

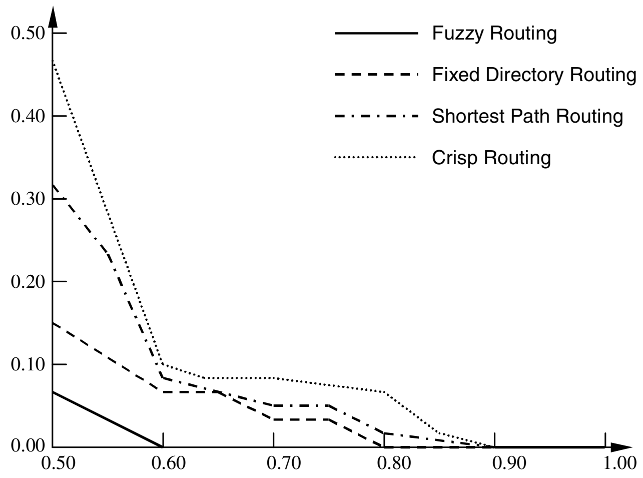


Figure 8.9. Percentage of connections rejected versus interarrival time.

From Figure 8.10, we see that the fuzzy routing algorithm also loses a smaller percentage of packets because of buffer overflow, than the other routing algorithms.

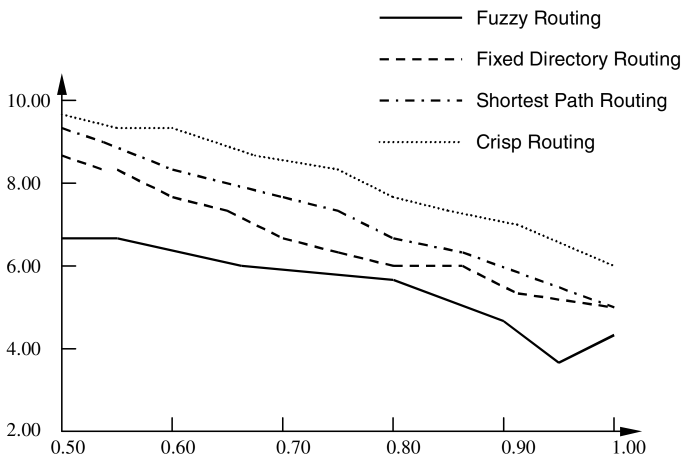


Figure 8.10. Percentage of packets lost versus interarrival time.

Finally, from Figure 8.11, we see that the fuzzy routing algorithm also achieves the shortest packet delays.

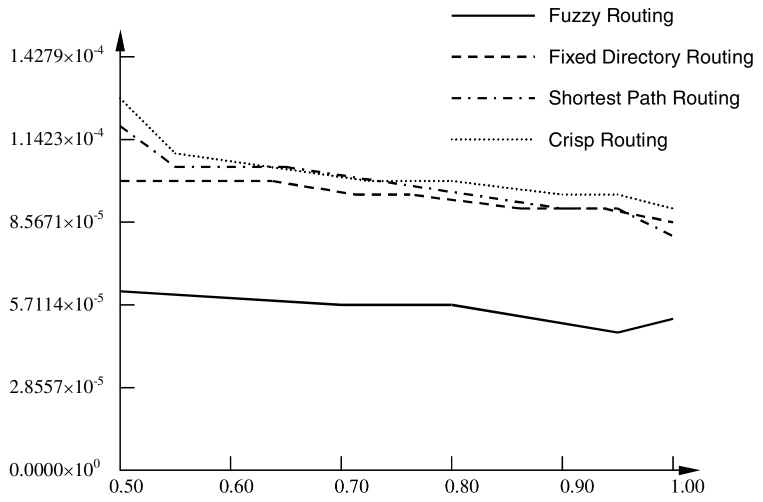


Figure 8.11. Mean packet delay versus interarrival time.

Overall, the fuzzy routing algorithm outperforms the other three routing algorithms in all criteria. The fuzzy algorithm disperses traffic in a more uniform manner among the paths in the network. It also handles an increased traffic load more efficiently.

9 Closure

Fuzzy logic, despite its lack of rigor, has scored remarkable successes in the field of nonlinear control. Its application to queuing systems control seems to be natural. This book presents the first systematic approach toward fuzzy queuing control.

The problems we solved showed us two things. When a complete analytical solution is known, the same results may be obtained via fuzzy control. If a solution is not known, the fuzzy approach still works, but unfortunately no proof of optimality can be derived. Simulation, however, shows that the fuzzy solutions give the lowest cost, although such investigations are by no means exhaustive.

The systems we presented in this book were governed by Markov or semi-Markov decision processes. We took advantage of the memoryless properties to develop control algorithms. Lack of memory, convenient as it may be, is not always realistic. One would be tempted to try fuzzy logic when the processes involved are conditioned probabilistically on more than one state. Such is the case in most complex manufacturing or communication networks. Practitioners in these fields are looking forward to applying new efficient control techniques. This is a natural extension of the present results, although not obvious or easy.

Another interesting avenue of investigation could be the systematization of all the algorithms of the book. Indeed, the control algorithms we have presented share a few general characteristics in the way the steps are made, but the details are ad hoc for each case. A framework would be desirable, whereby the algorithms are adapted to each particular system via learning mechanisms such as neuro-fuzzy techniques. This task is also nontrivial.

The aim of this book is to present in a coherent unit a new field. Our results may be used by students and practitioners in the field of queuing control. Some of the algorithms have already been applied to practical situations.

Fuzzy queuing control is by no means a complete discipline. A lot of work awaits us ahead in several directions. We have already given some hints for new avenues. Our hope is that this work will become a stepping stone to open up such new avenues that will provide practical results. The demand for good queuing control algorithms is ever increasing.

Appendix: Markov Queuing Models and Simulation

A.1 Introduction

Several results were presented in Chapters 4–8 of this book to illustrate the control algorithms. To validate the performance of these algorithms, we used simulation. In this appendix, we provide a brief introduction to the simulation of queuing systems.

A queuing system is a stochastic dynamic system where customers arrive, wait in queue until they are served by one or more servers, and finally depart. The evolution of a queuing system can be described by observing the states of its servers (e.g., type of service, busy, idle, on, or off) and the number of customers in each queue as they evolve over time. Evolution can be conceptualized as a sequence of states that the system visits during a period of observation. We shall refer to such sequence as the *trajectory* or *sample path* of the system. *Simulation* is a modeling technique that uses the computer to generate a sample path and test attributes of the stochastic system. A simulation model is a symbolic expression of the underlying laws that relate the current state to past states. Thus, given the state at time zero, the simulation model generates the next states and the corresponding transition epochs in sequence.

In the next sections, we present methods for simulating random variables and review some results from the theory of Markov processes that we use to develop simulation models for selected queuing systems.

A.2 Simulating Random Variables

A deterministic dynamic system has a unique trajectory. Queuing systems, however, are stochastic because customers arrive and complete service at random times. When we simulate a queuing system, part of the simulation effort goes to generating sequences of random variables, which represent a possible realization of the customer arrival and processing times. Such sequences are generated by the *random variate generators.*

Uniform random numbers or simply *random numbers* are sample values of a random variable U that is uniformly distributed on the interval [0, 1]. The most common random number generators are the *linear congruential generators*. Such generators start with an initial integer value Z_0, called the *seed*, and yield successive random numbers u_0, u_1, … by computing

$$Z_{n+1} = (aZ_n + b) \bmod c,$$

$$u_{n+1} = \frac{Z_{n+1}}{c},$$

where "x mod y" denotes the remainder of the division of x by y. The parameters a, b, and Z_0 are positive integers, all smaller than c. As Z_n uniquely determines the next random number and c is integer, the above scheme generates at most c different random numbers ranging from 0 to $(c - 1)$ before returning to the value Z_0 from which it has started. A good random number generator should have a long period and a uniform coverage of the interval [0, 1]. Setting c equal to a large integer is necessary but not sufficient to guarantee a long period for the Z_n's (for example, for $a = 1$ and $b = 0$, we get $Z_n = Z_0$ for all n). Hence, a and b have to be specified carefully.

Setting $b = 0$ yields the so-called *multiplicative congruential generator*, which is more efficient than the linear one because it saves one addition. The period of a multiplicative congruential generator is $c - 1$ if c is prime and the smallest integer k for which $a^k - 1$ is divisible by c is $k = c - 1$ (Knuth 1981). The generators

$$Z_{n+1} = (7^5 \times Z_n) \bmod (2^{31} - 1)$$

$$Z_{n+1} = (630{,}360{,}016 \times Z_n) \bmod (2^{31} - 1)$$

satisfy the above conditions, their period is $2^{31} - 2$, and therefore, they are implementable on any 32-bit personal computer that reserves one bit for the sign.

Next, we describe a common method for generating samples of random variables drawn from general distributions. The corresponding sampled values will be referred to as *random variates* to distinguish them from the random numbers u that have a uniform distribution.

Let X be a real random variable with known distribution function $F_X(x) = P(X \le x)$. Define a new random variable $U \triangleq F_X(X)$. Its distribution function is

$$F_U(u) = P(U \le u) = P[F_X(X) \le u] = P[X \le F_X^{-1}(u)] = F_X[F_X^{-1}(u)] = u,$$

assuming that the inverse function F_X^{-1} exists. By definition, F_X and, therefore, U range over [0, 1]. Thus, U is a random variable over [0, 1], and its distribution function is $F_U(u) = u$, which is the uniform distribution. From this, we obtain the following generator for the random variable X:

1. Generate a random number u.
2. Solve $F_X(x) = u$. Then, $x = F_X^{-1}(u)$ is a sample value of the random variable X.

This method is known as the *inverse transform method.*

The existence of F_X^{-1} is guaranteed if F_X is strictly increasing. If the distribution function is not increasing in the strict sense but it contains flat segments and discontinuities, as in the case of discrete random variables, then the inverse distribution function is defined as

$$F_X^{-1}(u) = \inf\{t: F(t) \ge u\}$$

for every $u \in [0, 1]$. In this case, x is the smallest t such that $F_X(t) \geq u$.

Example A.1

If X is exponentially distributed with parameter λ, then $F_X(x) = 1 - e^{-\lambda x}$ and the solution to $F_X(x) = u$ is

$$x = -\frac{\ln(1-u)}{\lambda}.$$

As U and $1 - U$ have the same uniform distribution, using u instead of $1 - u$ will produce a legitimate sample value for X; thus,

$$x = -\frac{\ln u}{\lambda}. \tag{A.1}$$

This saves a subtraction.

Example A.2

Suppose that X has a geometric distribution on $\{0, 1, \ldots\}$ with parameter q, probability mass function $P(X = k) = q(1 - q)^k$, $k = 0, 1, \ldots$, and distribution function

$$F_X(k) = q(1-q)^0 + \ldots + q(1-q)^k$$

$$= 1 - (1-q)^{k+1}.$$

The inverse transform returns the smallest integer x such that $F_X(x) \geq u$. As F_X is increasing, this is only possible if $F_X(x) \geq u > F_X(x - 1)$. From these inequalities, we obtain

$$x = \left\lfloor \frac{\ln(1-u)}{\ln(1-q)} \right\rfloor,$$

where $\lfloor t \rfloor$ is the largest integer such that $\lfloor t \rfloor < t$. As in the previous example, stochastic equivalence of U and $1 - U$ gives rise to a more economical generator,

$$x = \left\lfloor \frac{\ln u}{\ln(1-q)} \right\rfloor.$$

Example A.3

A discrete random variable X has probability mass function

$$P(X = x_j) = p_j, j = 1, 2, \ldots,$$

where $-\infty < x_1 < x_2 < \ldots$, $p_j > 0$, and $p_1 + p_2 + \ldots = 1$. Its distribution function is

$$F_X(x) = P(X \leq x) = \sum_{j: x_j \leq x} p_j$$

and

$$F_X(x_j) = F_X(x_{j-1}) + p_j.$$

The inverse transform returns the smallest value x_j such that $F_X(x_j) \geq u$. As in the previous example, this is only possible if $F_X(x_j) \geq u > F_X(x_{j-1})$. If X has a finite range of values $\{x_1, x_2, \ldots, x_n\}$, its sample values can be computed using the following algorithm:

Algorithm A.1. *Inverse transform method for arbitrary discrete distributions*

```
1. Precompute the values $F_X(x_j)$, $j = 1, 2, \ldots, n$.
2. To generate a sample value for $X$:
   a. Generate a random number $u$ and set $j = 0$.
   b. Set $j := j + 1$.
   c. If $F_X(x_j) \geq u$, then return the sample value $X = x_j$; otherwise go to step
      (b).
```

A.3 The Memoryless Assumption

The complexity of a dynamic system depends on the number of past states needed to determine the next state. This number is related to the memory inherent in the system. In general, systems with long memory require complex models, whereas systems lacking memory are easier to model.

Lack of memory means that the next state of the system depends only on the present state and not on past ones.

For a deterministic system, the memoryless property implies that if the system is in some state during $[0, t]$ and it happens to enter the same state at a later time T, then it will stay there until time $T + t$.

The memoryless property has a more general interpretation for random variables. Let X be a random variable that represents the time when the first customer arrives at a queuing system. Suppose that at time zero the system is empty. Then $P(X > t \mid X > 0)$ is the probability that the customer will arrive later than time t given that he has not arrived by time 0 or $X > 0$. Now suppose that at time T the customer has not yet arrived. The memoryless property implies that the conditional probability that the arrival will occur later than $T + t$ is independent of T, or

$$P(X > T + t \mid X > T) = P(X > t \mid X > 0). \qquad \text{(A.2)}$$

As $t \geq 0$, we have $P(X > t, X > 0) = P(X > t)$ and $P(X > T + t, X > T) = P(X > T + t)$ and Equation (A.2) becomes

$$\frac{P(X > T + t)}{P(X > T)} = \frac{P(X > t)}{P(X > 0)}. \qquad \text{(A.3)}$$

If X is an absolutely continuous nonnegative random variable with distribution function $F(x)$, then $F(0) = 0$ and Equation (A.3) yields

$$\frac{1 - F(T + t)}{1 - F(T)} = \frac{1 - F(t)}{1 - F(0)}$$

$$= 1 - F(t).$$

or, by defining $G(t) \triangleq 1 - F(t)$,

$$G(T + t) = G(T)\, G(t).$$

The only class of real valued functions $G(t)$, $t > 0$, satisfying the above functional equation is the exponential family with parameter λ (see Parzen 1962 for a proof); that is,

$$G(t) = e^{-\lambda t}.$$

Therefore, $F(t) = 1 - e^{-\lambda t}$, which implies that X has an exponential distribution with mean $1/\lambda$ and density function $f(t) = \lambda\, e^{-\lambda t}$. Its discrete counterpart is the geometric distribution.

Example A.4

Consider a server that is switched off after serving a random number of customers X, geometrically distributed on $\{0, 1, 2, \ldots\}$ with $P(X = 0) = q$. Suppose that the server has served c customers already. What is the probability that the server will be switched off after completing k more customers? Using the formulas for the geometric probability mass and distribution functions given in Example A.2, this probability is expressed as follows:

$$P(X = c + k \mid X \geq c) = \frac{P(X = c + k)}{P(X \geq c)}$$

$$= \frac{q\,(1-q)^{c+k}}{(1-q)^{c}} = q\,(1 - q)^{k},$$

which is independent of c. Hence, the geometric distribution is memoryless.

The exponential distribution is often used to describe the times between successive occurrences of independent events. Independence here means that the time of occurrence of the next event is independent of past events. The number $N(t)$ of occurrences during a time interval $[0, t]$ has the Poisson distribution,

$$P[N(t) = k] = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad . \qquad \text{(A.4)}$$

The equivalence of Poisson and exponential distributions follows by noting that the time X of occurrence of the first event exceeds t if and only if $N(t) = 0$. Therefore,

$$P(X > t) = P[N(t) = 0] = \frac{(\lambda t)^0}{0!} e^{-\lambda t} = e^{-\lambda t},$$

which shows that X is exponential. By the independence of events, it follows similarly that all the times between successive occurrences are exponential.

The Poisson distribution gives rise to one of the simplest stochastic processes exhibiting the memoryless property, the stationary Poisson process.

Definition A.1. The *stationary* or *homogeneous Poisson process* $N(t)$ is a process that counts the number of events that occur between time 0 and time t, $t \geq 0$, and it has the following properties:

1. $N(0) = 0$ with probability 1.
2. $N(t)$ has independent increments; that is, for all choices of $0 = t_0 < t_1 < \ldots < t_n$, the random variables $N(t_i) - N(t_{i-1})$, $i = 1, \ldots, n$, are independent.
3. $P[N(t + \delta) - N(t) = 1] = \lambda\delta + o(\delta)$, where $o(\delta)$, the "little oh" function, denotes a function with the property $\lim_{\delta\to 0}[|o(\delta)|/|\delta|] = 0$.
4. There are no simultaneous events; that is, $P[N(t + \delta) - N(t) > 1] = o(\delta)$.

From the above definition, the following properties can be derived (Parzen 1962):

(a) The probability mass function of $N(t)$ is given by Equation (A.4).
(b) $N(t)$ has stationary increments; that is, the random variables $N(t + \delta) - N(t)$ have the same probability distribution for all $t \geq 0$.
(c) The time between two successive events has an exponential distribution with mean $1/\lambda$.
(d) Given that n events have occurred in $[0, t + \delta]$, the number of events in $[0, t]$ has a *binomial distribution* with parameter $q = t/(t + \delta)$ and probability mass function

$$P[N(t) = n \mid N(t + \delta) = n] = \binom{n}{k} q^k (1-q)^{n-k} .$$

In summary, exponential, geometric, Poisson, and binomial processes are memoryless and these are the only processes with this property. This, however, is not the case in many queuing applications. Practically all manufacturing, communication, and public service systems exhibit strong memory. For example, machines in a production network are subject to deterioration and have relatively short processing times during the first few processing cycles when their tools are new but become slower progressively as their tools wear out. Such systems can be represented more realistically by Markov processes, which are described next.

A.4 Continuous-Time Markov Chains

The queuing networks in this book are modeled by Markov chains.

Definition A.2. A continuous-time *Markov chain* X_t, $t \in (-\infty, \infty)$, is a stochastic process that moves in a countable set of states $\{S_1, S_2, \ldots\}$ and satisfies the *Markov property*,

$$P(X_{tn} \mid X_{tn-1}, X_{tn-2}, \ldots X_{t0}) = P(X_{tn} \mid X_{tn-1}) \tag{A.5}$$

for all choices of $-\infty < t_0 < t_1 < \ldots < t_n < \infty$.

The Markov property says that the future behavior of the process is conditioned only on the current state and not on the past. It extends the memoryless property of the Poisson process to systems with several states and to processes that do not satisfy the independent-increment property.

In analogy to properties 3 and 4 of the definition of the stationary Poisson process, we make the following two additional assumptions:

$$P(X_{t+\delta} = S_j \mid X_t = S_i) = \Lambda_{i,j}\delta + o(\delta), \ \text{for } j \neq i, \tag{A.6}$$

and there are no simultaneous transitions; i.e., for any t_1 and t_2 in $[t, t+\delta]$,

$$P(X_{t_1} = S_j, X_{t_2} = S_k \mid X_t = S_i) = o(\delta), \text{for } j \neq i, k \neq i, \text{and } j \neq k. \tag{A.7}$$

From assumption (A.6), we compute the probability of escaping from state S_i in δ time units

$$P(X_{t+\delta} \neq S_i \mid X_t = S_i) = \Lambda_i\delta + o(\delta), \tag{A.8}$$

where $\Lambda_i \triangleq \sum_{j\neq i}\Lambda_{i,j}$.

The parameters $\Lambda_{i,j}$ and Λ_i are the *transition rates* associated with state S_i. The dynamics of a Markov chain are completely specified by its transition rates. Indeed,

(P1) The sojourn time in state S_i has an exponential distribution with parameter Λ_i

(P2) The probability that the system will move from state S_i to S_j is $\Lambda_{i,j}/\Lambda_i$, independent of the time spent in S_i

Property (P1) follows by observing that Λ_i in (A.8) plays the same role as the parameter λ of the stationary Poisson process in Definition A.1 for which the times between successive events are exponentially distributed with mean value $1/\lambda$.

To prove property (P2), suppose that the chain enters state S_i at time τ and leaves this state at time t; that is, $X_t = S_j \neq S_i$ and $X_u = S_i$ for every u in $[\tau, t)$. By the Markov property, the event $\{X_u = S_i, \text{for } u \in [\tau, t)\}$ carries the same information as the event $\{X_{t^-} = S_i\}$. Therefore,

$$\begin{aligned} P[X_t = S_j \mid X_t \neq S_i, X_u = S_i, u \in [\tau, t)] &= \lim_{\delta\to 0} P(X_t = S_j \mid X_t \neq S_i, X_{t-\delta} = S_i) \\ &= \lim_{\delta\to 0} \frac{P(X_t = S_j, X_t \neq S_i \mid X_{t-\delta} = S_i)}{P(X_t \neq S_i \mid X_{t-\delta} = S_i)} \\ &= \lim_{\delta\to\infty} \frac{\Lambda_{i,j}\delta + o(\delta)}{\Lambda_i\delta + o(\delta)} = \frac{\Lambda_{i,j}}{\Lambda_i}. \end{aligned}$$

Therefore, the probability of a conditional transition to state S_j, given that a transition occurs, is independent of the intertransition time.

Example A.5

We examine the M/M/1 queuing system with server vacations, as described in Section 4.2. The interarrival and service times of customers are independent, exponential random variables with mean values $1/\lambda$ and $1/\mu$, respectively. For this system, the optimal control is of the threshold type; that is, all arrivals finding K customers ahead (in queue and in service) are rejected, for some specified threshold value K. Thus, we have an equivalent M/M/1/K queuing system whose state is described by the number of customers in the system, $n = 0, 1, \ldots, K$. The state transitions are shown in Figure A.12.

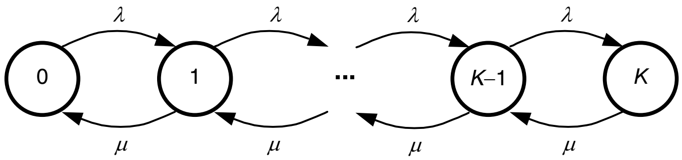


Figure A.12. State transitions for an M/M/1/K queuing system.

The transition rates for each state are given by

$$\Lambda_{n,\,n+1} = \begin{cases} \lambda & \text{if } n < K, \\ 0 & \text{if } n = K, \end{cases}$$

$$\Lambda_{n,\,n-1} = \begin{cases} \mu & \text{if } n > 0, \\ 0 & \text{if } n = 0, \end{cases}$$

and

$$\Lambda_n = \begin{cases} \lambda & \text{if } n = 0, \\ \lambda + \mu & \text{if } 0 < n < K, \\ \mu & \text{if } n = K. \end{cases}$$

All other transitions are zero, e.g., $\Lambda_{n,\,n+10} = 0$.

Example A.6

Consider the system with two workstations in tandem, as described in Section 4.6. Customers arrive in the first station according to a Poisson process with rate λ, and upon completion of service, they join station 2, which is served by a server with mean rate μ. The system controls the server of station 1 by altering its mean service rate to any value u in $[0, a]$. Such decisions are made by the fuzzy controller using information about the state (s_1, s_2), where s_i is the number of customers in station i. Therefore, we have $u = u(s_1, s_2)$. The system can be modeled as a Markov chain whose transitions from state (s_1, s_2) are shown in Figure A.13.

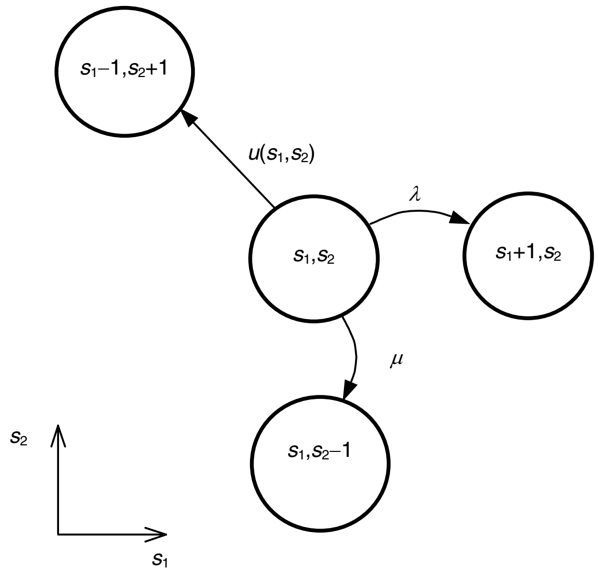


Figure A.13. Transitions from state (s_1, s_2).

The transition rates for each state (s_1, s_2), $s_i = 0, 1, \ldots$, are given by

$$\Lambda_{(s1,s2),\,(s1-1,s2+1)} = \begin{cases} u(s_1, s_2) & \text{if } s_1 > 0, \\ 0 & \text{if } s_1 = 0, \end{cases}$$

$$\Lambda_{(s1,s2),\,(s1+1,s2)} = \lambda,$$

$$\Lambda_{(s1,s2),\,(s1,s2-1)} = \begin{cases} \mu & \text{if } s_2 > 0, \\ 0 & \text{if } s_2 = 0, \end{cases}$$

and

$$\Lambda_{(s1,s2)} = \begin{cases} \lambda + \mu + u(s_1, s_2) & \text{if } s_1 > 0 \text{ and } s_2 > 0, \\ \lambda + \mu & \text{if } s_1 = 0 \text{ and } s_2 > 0, \\ \lambda + u(s_1, s_2) & \text{if } s_1 > 0 \text{ and } s_2 = 0, \\ \lambda & \text{if } s_1 = s_2 = 0. \end{cases}$$

All other transitions are zero, e.g., $\Lambda_{(s1,s2),\,(s1+10,s2)} = 0$.

A.5 Simulation of a Markov Queuing System

We now develop simulation models of Markovian queuing networks using the properties (P1) and (P2) of the previous section.

We begin with queuing systems having a single state variable that ranges over the set $\{S_1, S_2, \ldots\}$, such as the M/M/1/K system of Example A.5. Suppose that at time zero, the system is in state S_i. By property (P1), the system will stay in this state for some random interval of time that has an exponential distribution with mean $1/\Lambda_i$. Upon leaving the initial state, by property (P2), the system moves to another state S_j with probability $\Lambda_{i,j}/\Lambda$, where it stays for some random time, and so forth. The following is a simulation algorithm that generates a sample path of the system.

Algorithm A.2. *Simulation model of a Markov chain*

1. *Initialize:* Specify the system states, the transition rates, the total simulation time T, and the initial state S_i. Set the simulation clock at $t = 0$.
2. *Compute sojourn time in state S_i:* From Equation (A.1), compute a sample value x using the exponential random variate generator with parameter $\lambda = \Lambda_i$:

$$x = -\frac{\ln u}{\Lambda_i}.$$

3. *Advance simulation clock:* Set $t := t + x$. If $t \geq T$, then terminate the simulation; otherwise go to step (4).
4. *Choose next state S_j:* Invoke Algorithm A.1 (Example A.3, Section A.1) with $p_j = \Lambda_{i,j}/\Lambda_i$ to determine the next state S_j of the system.
5. *Move to next state:* Set $i = j$ and go to step (2).

The simulation of Markovian queuing systems follows Algorithm A.2 and differs only in the definition of the state variables and calculation of the corresponding transition rates, as shown in the following examples.

Example A.7

Consider an M/M/1/K queuing system (see Example A.5) that earns a reward w per accepted customer and incurs a holding cost h per customer per time unit. The following algorithm computes the average profit rate of the system during Q state transitions.

1. *Initialize:* Specify the parameters λ, μ, h, and w and the total number of transitions Q. Initialize the simulation clock $t = 0$, the state of the system $n = 0$, the current transition $q = 0$, and the profit $J = 0$.
2. *Compute sojourn time in state n:* If $n < K$, then set $\Lambda = \lambda$; otherwise set $\Lambda = 0$. In addition, if $n > 0$, then set $\Lambda := \Lambda + \mu$. Generate a random number u and compute

$$x = -\frac{\ln u}{\Lambda}.$$

3. *Advance time and update profit:* Set $t := t + x$, $q := q + 1$, and $J: = J - hnx$. If $q = Q$, then compute the average profit rate $J_{\text{av}} = J/t$ and terminate the simulation; otherwise go to step (4).

4. *Determine the type of next event:* Generate another random number u. If $u \le \lambda/\Lambda$, then the next event is an arrival; otherwise it is a departure.
5. *Execute next event (move to next state):* If a new customer arrives, then update the state and the profit; that is, $n := n + 1$ and $J := J + w$; otherwise update only the state, $n := n - 1$. Go to step (2).

In the following example, we model a two-dimensional system.

Example A.8

Consider a tandem system as described in Example A.6, which incurs a holding cost rate h_i per customer in station i, $i = 1, 2$. A fuzzy controller controls the rate u of the server in station 1 using information about the state (s_1, s_2), where $u \in [0, a]$. In Section 4.6, we have seen that for $h_1 < h_2$, the optimal control is bang-bang, taking values 0 or a; that is, there is an increasing function $S(s_2)$, such that $u = 0$ if $s_1 < S(s_2)$ and $u = a$ if $s_1 \ge S(s_2)$. The function S is determined by the fuzzy controller. The following algorithm computes the average cost rate of the system during Q state transitions.

1. *Initialize:* Specify the parameters λ, a, μ, h_1, and h_2 $(h_1 < h_2)$, the total number of transitions Q, and the initial state $s_1 = s_2 = 0$. Initialize the simulation clock $t = 0$, the current transition $q = 0$, and the cost $C = 0$.
2. *Compute sojourn time in state* (s_1, s_2)*:* Set $\Lambda = \lambda$. If $s_1 > 0$ and $s_1 \ge S(s_2)$, then set $\Lambda := \Lambda + a$ (server 1 is working). If $s_2 > 0$, then set $\Lambda := \Lambda + \mu$ (server 2 is working). Generate a random number u and compute

$$x = -\frac{\ln u}{\Lambda}$$

3. *Advance time and update cost:* Set $t := t + x$, $q := q + 1$, and $C: = C + h_1 s_1 + h_2 s_2$. If $q = Q$, then compute the average cost rate $C_{\text{av}} = C/t$ and terminate the simulation; otherwise go to step (4).
4. *Determine the type of next event:* Generate another random number u. If $u \le \lambda/\Lambda$, then the next event is an arrival; otherwise if $s_2 > 0$ and $u \le (\lambda + \mu)/\Lambda$, then it is a departure from station 2, but if $s_2 = 0$ or $u > (\lambda + \mu)/\Lambda$, then it is a departure from station 1.
5. *Execute next event (move to next state):* If a new customer arrives, then $s_1 := s_1 + 1$; if a customer leaves station 2, then $s_2 := s_2 - 1$; and if a customer moves from station 1 to station 2, then $s_1 := s_1 - 1$ and $s_2 := s_2 + 1$. Go to step (2).

References

Altman E (2000) Applications of Markov decision processes in communication networks: a survey. (Research report No. 3984, INRIA, Sophia Antipolis)

Altman E, Nain P (1993) Optimal control of the M/G/1 queue with repeated vacations of the server. IEEE T Automat Contr 38:1766–1775

Baras J, Dorsey J (1981) Stochastic control of two partially observed competing queues. IEEE T Automat Contr AC-26:1106–1117

Baras J, Dorsey J, Makowski A (1985) Two competing queues with linear cost: the cμ rule is often optimal. Adv Appl Probab 17:237–238

Bäuerle N, Brüstl G, Rieder U (1998) Optimal scheduling in heterogeneous two-station queueing networks. Math Meth Oper Res 48:337–347

Beja A, Teller A (1975) Relevant policies for Markovian queueing systems with many types of service. Manage Sci 21:1049–1054

Bell C (1971) Characterization and computation of optimal policies for operating an M/G/1 queueing system with removable servers. Oper Res 19:208–218

Bell C (1973) Optimal operation of an M/G/1 priority queue with removable servers. Oper Res 23:1281–1289

Bell C (1975) Turning off a server with customers present: is this any way to run an M/M/c queue with removable servers? Oper Res 23:571–574

Bell C (1980) Optimal operation of an M/M/2 queue with removable servers. Oper Res 28:1189–1204

Bell C, Stidham S (1983) Individual versus social optimization in the allocation of customers to alternative servers. Manage Sci 29:831–839

Blanc JPC, Waal PR, Nain P, Towsley D (1992) Optimal control of admission to a multiserver queue with two arrival streams. IEEE T Automat Contr 37:785–797

Boxma O (1976) On the D-policy for the M/G/1 queue. Manage Sci 22:916–917

Buyukkoc C, Varaiya PV, Walrand J (1985) The cμ rule revisited. Adv Appl Probab 17:186–209

Chen H, Yang P, Yao DD (1994) Control and scheduling in a two station queueing network: optimal policies and heuristics. Queueing Syst 18:301–332

Chen S, Nahrstedt K (1998) An overview of quality of service routing for next-generation high-speed networks: problems and solutions. IEEE Network 12:64–79

Chow YC, Kohler WH (1979) Models for dynamic load balancing in a heterogeneous multiple processor system. IEEE T Comput C-28:354–361

Cohen J (1986) On the optimal switching level for an M/G/1 queueing system. Stoch Proc Appl 4:297–376

Courcoubetis CA, Varaiya PV (1984) Serving the process with least thinking time maximizes resource utilization. IEEE T Automat Contr AC-29:1005–1008

Cox DR, Smith WL (1961) Queues. Methuen Monographs, London

Crabill TB (1972) Optimal control of a service facility with variable exponential service time and constant arrival rate. Manage Sci 18:560–566

Crabill TB (1974) Optimal control of a maintenance system with variable service rates. Oper Res 22:736–745

Crabill TB, Gross G, Magazine MJ (1977) A classified bibliography of research on optimal design and control of queues. Oper Res 25:219–232

Deshmukh S, Jain S (1977) Capacity design and service quality control in a queueing system. Oper Res 25:651–661

Doshi BT (1978) Optimal control of the service rate in an M/G/1 queuing system. Adv Appl Probab 10:682–701

Driankov D, Hellendoorn H, Reinfrank M (1996) An introduction to fuzzy control. Springer-Verlag, Berlin Heidelberg New York

Ephremides A, Varaiya PV, Walrand J (1980) A simple dynamic routing problem. IEEE T Automat Contr AC-25:690–693

Federgruen A, Tijms H (1980) Computation of the stationary distribution of the queue size in an M/G/1 queueing system with variable service rate. J Appl Probab 7:515–522

Gallish E (1979) On monotone optimal policies in a queueing model of M/G/1 type with controllable service time distribution. Adv Appl Probab 11:870–887

Gebhard RF (1967) A queueing process with bilevel hysteretic service control. Nav Res Logist Q 14:55–68

Ghoneim H, Stidham S (1985) Control of arrivals to two queues in series. Eur J Oper Res 21:399–409

Hajek B (1984) Optimal control of two interacting service stations. IEEE T Automat Contr AC-29:491–499

Harrison JM (1985) On the optimality of first come last served queues. Econometrica 53:201–202

Harrison JM, Wein LM (1990) Scheduling networks of queues: heavy traffic analysis of a two-station closed network. Oper Res 38:1052–1064

Hassin R (1975) Dynamic scheduling of a single class queue: discount optimality. Oper Res 23:270–282

Heyman DP (1968) Optimal operating policies for M/G/1 queueing systems. Oper Res 16:362–382

Heyman DP (1977) The T-policy for the M/G/1 queue. Manage Sci 23:775–778

Hlynka M, Stanford DA, Poon WH, Wang T (1994) Observing queues before joining. Oper Res 42:365–371

Jo K (1983) Optimal service-rate control of exponential queueing systems. J Oper Res Soc Jpn 26:147–165

Jo KY, Stidham S (1983) Optimal service-rate control of M/G/1 queueing systems using phase methods. Adv Appl Probab 15:616–637

Johansen SG, Stidham S (1980) Control of arrivals to a stochastic input-output system. Adv Appl Probab 12:972–999

Kimura T (1981) Optimal control of an M/G/1 queueing system with removable server via diffusion approximation. Eur J Oper Res 8:390–398

Kitaev MY, Rykov VV (1995) Controlled queueing systems. CRC Press, Boca Raton

Kleinrock L (1975) Queueing Systems, Volume I: Theory. Wiley-Interscience, New York Chichester Brisbane Toronto

Klimov GP (1974) Time sharing service systems I. Theor Probab Appl 19:532–551

Knudsen NC (1972) Individual and social optimization in a multi-server queue with a general cost-benefit structure. Econometrica 40:515–528

Knuth DE (1981) The art of computer programming, Volume 2. Addison-Wesley, Reading

Koyanagi J, Kawai H (1995) An assignment problem for a parallel queueing system with two heterogeneous servers. Math Comput Model 22:173–181

Kurose JF, Ross KW (2003) Computer networking: a top-down approach featuring the Internet. Addison-Wesley, Boston San Francisco New York

Langen HJ (1982) Applying the method of phases in the optimization of queueing systems. Adv Appl Probab 14:122–142

Lin W, Kumar PR (1984) Optimal control of a queueing system with two heterogeneous servers. IEEE T Automat Contr AC-29:696–703

Lippman SA (1973) Semi-Markov decision process with unbounded rewards. Manage Sci 19:717–731

Lippman SA (1975) Applying a new device in the optimization of exponential queuing systems. Oper Res 23:687–710

Lippman SA, Stidham S (1977) Individual versus social optimization in exponential congestion systems. Oper Res 25:233–247

Lu FV, Serfozo RF (1984) M/M/1 queueing decision processes with monotone hysteretic optimal policies. Oper Res 32:1116–1132

Makis V (1984) A note on the optimal control limit for a batch service queueing system: average cost rate. Opsearch 21:113–116

Miller BL (1969) A queueing reward system with several customer classes. Manage Sci 16:235–245

Naor P (1969) On the regulation of queue size by levying tolls. Econometrica 37:15–24

Parzen E (1962) Stochastic processes. Holden-Day, San Francisco

Phillis YA, Kouikoglou VS (1995) An entropy approach to queueing control. Proceedings of the 34th IEEE CDC, New Orleans, pp 3644–3645

Rath JH (1975) Controlled queues in heavy traffic. Adv Appl Probab 7:656–671

Righter R, Shanthikumar JG (1989) Scheduling multiclass single server queueing systems to stochastically maximize the number of successful departures. Probab Eng Inform Sc 3:323–333

Rosberg Z, Kermani P (1989) Customer routing to different servers with complete information. Adv Appl Probab 21:861–882

Rosberg Z, Varaiya PP, Walrand JC (1982) Optimal control of service in tandem queues. IEEE T Automat Contr AC-27:600–610

Sabeti H (1973) Optimal selection of service rates in queueing with different cost. J Oper Res Soc Jpn 16:15–35

Schassberger R (1975) A note on optimal service selection in a single server queue. Manage Sci 21:1326–1331

Seidmann A, Schweitzer PJ (1984) Part selection policy for a flexible manufacturing cell feeding several production lines. IIE Trans 16:355–362

Shanthikumar JG, Yao DD (1992) Multiclass queueing systems: polymatroidal structure and optimal scheduling control. Oper Res 40:S293–S299

Sobel MJ (1982) The optimality of full service policies. Oper Res 17:639–649

Stidham S (1978) Socially and individually optimal control of arrivals to a GI/M/1 queue. Manage Sci 24:1598–1610

Stidham S (1985) Optimal control of admission to a queueing system. IEEE T Automat Contr AC-30:705–713

Stidham S, Weber R (1989) Monotonic and insenstitive optimal policies for control of queues with undiscounted costs. Oper Res 87:611–625

Stidham S, Weber R (1993) A survey of Markov decision models for control of networks of queues. Queueing Syst 13:291–314

Teghem J (1987) Optimal control of a removable server in an M/G/1 queue with finite capacity. Eur J Oper Res 31:358–367

Tijms H, van der Schouten Duyn F (1978) Inventory control with two switch over levels for a class M/G/1 queueing systems with variable arrival and service rate. Stoch Proc Appl 6:213–222 van Nunen JAEE, Puterman ML (1983) Computing optimal control limits for GI/M/s queuing systems with controlled arrivals. Manage Sci 29:725–734

Viniotis I, Ephremides A (1988) Extension of the optimality of the threshold policy in heterogeneous multiserver queueing systems. IEEE T Autom Contr 33:104–109

Walrand J (1984) A note on 'optimal control of a queueing system with two heterogeneous servers'. Syst Control Lett 4:131–134

Walrand J (1988) An introduction to queueing networks. Prentice-Hall, Englewood Cliffs

Wang K, Huang H (1995) Optimal control of a removable server in an $M/E_k/1$ queueing system with finite capacity. Microelectron Reliab 35, 1023–1030

Weber RR (1978) On the optimal assignment of customers to parallel servers. J Appl Probab 15:406–413

Weber RR, Stidham S (1987) Optimal control of service rates in networks of queues. Adv Appl Probab 19:202–218

Wein LM (1990) Scheduling networks of queues: heavy traffic analysis of a two-station network with controllable inputs. Oper Res 38:1065–1078

Whitt W (1986) Deciding which queue to join: some counterexamples. Oper Res 34:55–62

Winston W (1977) Optimality of the shortest line discipline. J Appl Probab 14:181–189

Winston W (1978) Optimality of monotonic policies for multiple server exponential queueing systems with state-dependant arrival rates. Oper Res 26:1089–1094

Xu S, Righter R, Shanthikumar JG (1992) Optimal dynamic assignment of customers to heterogeneous servers in parallel. Oper Res 40:1126–1138

Xu S, Shanthikumar JG (1993) Optimal expulsion control: a dual approach to admission control of an ordered-entry system. Oper Res 41:1137–1152

Yadin M, Naor P (1963) Queueing systems with a removable service station. Oper Res Quart 14:393–405

Yadin M, Naor P (1967) Queueing systems with variable service capacities. Nav Res Logist Q 14:43–53

Yechiali U (1972) Customers' optimal joining rules for the GI/M/s queue. Manage Sci 18:434–443

Zacks S, Yadin M (1970) Analytic characterization of the optimal control of a queueing system. J Appl Probab 7:617–633

Zadeh LA (1965) Fuzzy sets. Informat Control 8:338–353

Index