

# **FUZZIFYING MARKOV DECISION PROCESS**

A Thesis

Submitted to the Graduate Faculty of the  
Louisiana State University and  
Agricultural and Mechanical College  
in partial fulfillment of the  
requirements for the degree of  
Masters of Science in Electrical Engineering  
in

The Department of Electrical and Computer Engineering

by

Syed Irshad Ahmed  
B.E., Osmania University, 1999  
May 2005

# Table of Contents

|  |    |
|--|----|
| ABSTRACT.....  | IV |
| CHAPTER 1 INTRODUCTION.....                            | 1  |
| 1.1 Background.....                                    | 1  |
| 1.2 Research Goal and Scope.....                       | 2  |
| 1.3 Thesis Outline.....                                | 3  |
| CHAPTER 2 MARKOV DECISION PROCESSES.....               | 5  |
| 2.1 An Example.....                                    | 6  |
| 2.2 Finite State Controlled Markov Chain.....          | 9  |
| 2.2.1 Dynamic Programming.....                         | 13 |
| 2.2.2 Algorithm.....                                   | 18 |
| 2.3 Infinite Horizon.....                              | 19 |
| 2.3.1 Stationary Markov Policy.....                    | 20 |
| 2.3.2 Value Iteration.....                             | 23 |
| 2.3.3 Linear Programming.....                          | 23 |
| CHAPTER 3 FUZZY MATHEMATICS.....                       | 27 |
| 3.1 Elements of Fuzzy Mathematics.....                 | 28 |
| 3.2 Operations on Fuzzy Sets.....                      | 29 |
| 3.3 Overview of Design.....                            | 30 |
| 3.3.1 Preprocessing.....                               | 31 |
| 3.3.2 Fuzzification.....                               | 32 |
| 3.3.3 Rule Base.....                                   | 34 |
| 3.3.4 Inference Engine.....                            | 36 |
| 3.3.5 Defuzzification.....                             | 39 |
| 3.3.6 Post Processing.....                             | 41 |
| CHAPTER 4 FUZZIFYING MARKOV DECISION PROCESSES.....    | 42 |
| 4.1 Motivation.....                                    | 44 |
| 4.2 Proposed Fuzzyfying Markov Decision Processes..... | 45 |
| 4.2.1 Fuzzy Membership Function.....                   | 45 |
| 4.3 Example.....                                       | 51 |
| CHAPTER 5 CONCLUSION.....                              | 61 |
| 5.1 Concluding Remarks.....                            | 61 |
| 5.2 Future Research.....                               | 62 |

REFERENCES.....63

VITA.....64

## **Abstract**

Markov decision processes have become an indispensable tool in applications as diverse as equipment maintenance, manufacturing systems, inventory control, queuing networks and investment analysis. Typically we have a controlled Markov chain on a suitable state space in which transitional probabilities depend on the policy (or decision maker) which comes from a set of possible actions. The main problem of interest would be to find an optimal policy that minimizes the associated cost.

Linear Programming has been widely used to find the optimal Markov decision policy. It requires solutions of large systems of simultaneous linear equations. By the fact that the complexity in linear programming increases much faster with the increase in the number of states which is often called curse of dimensionality, the linear programming method can handle only small models.

This thesis presents a new method to lessen the curse of dimensionality. By assuming certain monotonicity property for the transition probability, it is shown that a fuzzy membership function can be used to reduce the number of states. The use of membership functions help to reduce the number of the states. However all the states remain intact through the use of the membership value. That is, those states eliminated can be recovered through interpolation with the aid of membership functions. This new proposed method is shown to be effective in coping with the curse of dimensionality.

# **Chapter 1**

## **Introduction**

### **1.1 Background**

Since the introduction of optimal control, the subject of decision making under uncertainty has grown into a ramified area with applications in several branches of engineering and in those areas of social sciences concerned with policy analysis and prescription. The theoretical approaches to the subject are based on the invention in the 1950s of dynamic programming and the rich body of research in statistical time series analysis. Although theoretically appealing, these approaches found little practical use since they demand a computing capacity that was too expansive.

Markov decision processes, one of the main optimal decision making tools, have become indispensable in applications as diverse as manufacturing systems, equipment maintenance, inventory control, queuing networks and investment analysis. Typically we have a controlled Markov chain on a suitable state space in which transitional probabilities depend on the policy (or decision maker) which come from a set of possible actions. The main problem of interest would be to find an optimal policy that minimizes the associated cost.

Typically we have finite horizon and infinite horizon Markov decision processes. Under finite planning horizon, the method of value iteration [ref 1] is perfect, but rarely the planning horizon is well defined. Most often the process is assumed to operate over an unknown period of time with no predetermined stage of termination. In such cases the abstraction of an infinite planning horizon seems more relevant.

Another method is linear programming [ref 2]. The linear programming method has almost the opposite characteristics of the value iteration method. Because of the more complicated mathematical formulation involving solutions of large systems of simultaneous linear equations, the method can handle rather small models with, for example, a few hundred states. Complexity in linear programming increases much faster with the increase in number of states which is often called curse of dimensionality. On the other hand, the method is exact and very efficient in the sense of fast convergence. The rewards are not allowed to depend on the stage except for a fixed rate of annual increase or decrease. Due to these problems various industries are facing tremendous difficulty in using Markov decision process when a large number of states is involved.

## **1.2 Research Goal and Scope**

The objective of this research is to develop an approximate optimization technique. We introduce the concept of fuzzy membership function into Markov decision processes. Each state has a particular transitional probability. By the use of membership functions, we can mathematically reduce the number of the states, but all the states eliminated can be recovered through the use of membership values. This thesis is devoted to develop a systematic procedure to accomplish the approximate optimal decision making. By an application of the fuzzy principles to Markov decision processes we are able to cope with

the “curse of dimensionality”, assuming that certain monotonicity property holds true for the transition probability.

### **1.3 Thesis Outline**

The remainder of the thesis is organized as follows.

Chapter 2 is based primarily on the existing literature. It reviews the basic results of stochastic systems modeled as finite state controlled Markov chains and infinite state controlled Markov chains. In the first section we present a basic stochastic problem using a queuing system. We introduce the concept of finite state controlled chains and dynamic programming in the second section. Then we extend the results to the case of infinite horizon with stationary policy to find the optimal policy using linear programming.

The fuzzy theory is discussed in Chapter 3, which is again based on the existing literature. The chapter is divided into three sections. In the first section preliminary elements on fuzzy mathematics are discussed. In the second section we discuss some operations on fuzzy sets. In the third section we give an introduction to fuzzy control design. We introduce the concept of fuzzy membership functions which will be used in the fourth chapter to reduce the number of states in a Markov decision processes.

In Chapter 4 we introduce a new optimization technique for Markov decision processes called Fuzzified Markov decision processes. This chapter is also divided into three sections. In the first section we give a brief introduction to the problem in hand and present an example to motivate the importance of the problem to be studied. In the second section we propose a Fuzzified Markov decision processes technique, and develop an approximate optimization procedure. The effectiveness of the proposed technique is illustrated by simulations examples with applications in the third section.

Chapter 5 concludes the thesis together with some ideas presented for the future directions of research along the line of this thesis.

## **Chapter 2**

### **Markov Decision Processes**

This chapter is based primarily on the existing literature, and is divided into three sections. It presents the basic results for stochastic systems modeled as finite state controlled Markov chains and infinite state controlled Markov chains. In the first section we present a basic stochastic problem using a queuing system. We introduce the concept of finite state controlled chains and dynamic programming in the second section. Then we extend the results to the case of infinite horizon with stationary policy to seek the optimal policy using the linear programming.

Consider a system being observed over a finite or infinite time horizon split up into period or stages. At each stage, the state of the system is observed, and a decision (or an action) concerning the system has to be made. The decision influences the state to be observed at the next stage, and depending on the state and the decision made, an immediate reward is gained. The expected total rewards from the present stage until the end of the planning horizon is expressed by a value function. The relation between the value function at the present stage and the one at the following stage is expressed by the functional equation. Optimal decisions depending on stage and state are determined

backwards step by step as those maximizing the right hand side of the functional equation. This way of determining an optimal policy is based on the Bellman principle of optimality which says: “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”.

Consider a discrete time Markov decision process with a finite state space  $X = \{1, 2, \dots, x\}$  and a finite action set  $U$ . a policy  $g$  is a map assigning to each state  $i$  an action  $g(i) \in U$ . Let  $P_{ij}$  be the transition probability from state  $i$  to state  $j$  and  $P_{ij}^a$  be the transition probability from state  $i$  to state  $j$  if action  $a$  is taken.

An optimal policy is defined as a policy that maximizes or minimizes some predefined objective function. The optimization technique depends on the form of the objective function or in other words on the criterion of optimality. The choice of criterion depends on whether the planning horizon is finite or infinite. Let us proceed with an example.

## 2.1 An Example

Consider a machine whose condition at time  $k$  is described by the state  $x_k$  which can take the values 1 or 2 with the interpretation that  $x_k = 1$  or  $x_k = 2$  depending on whether the machine is in operational or failed condition. For the moment there is no control actions allowed so that the machine behavior is autonomous. Suppose the machine is operational at time  $k$ , so  $x_k = 1$ , and there is a probability  $q > 0$  that it will fail in next period, so  $x_{k+1} = 2$ ; with probability  $1 - q$  it will continue to remain operational, so  $x_{k+1} = 1$ . Suppose further that  $q$  does not depend upon previous values  $x_{k-1}, \dots, x_0$ . Finally, suppose

that a failed machine continues to remain failed, so that  $x_{k+1} = 2$  with probability 1, if  $x_k = 2$ . Then  $\{x_k, k \geq 0\}$  is a Markov chain whose transition probabilities are described by the matrix

$$P = \begin{bmatrix} 1-q & q \\ 0 & 1 \end{bmatrix} \quad (2.1.1)$$

The Markov property is expressed by

$$pro\{x_{k+1} = j / x_k = i, x_{k-1}, \dots, x_0\} = P_{ij}, \quad i, j \in \{1, 2\} \quad (2.1.2)$$

We now introduce two control actions. Let  $u_k^1$  denote the intensity of machine use at time  $k$ . it takes on values  $u_k^1 = 0, 1 \text{ or } 2$  accordingly as machine is not used, is in light use, or in heavy use. Suppose that the greater the intensity of use, the larger the likelihood of machine failure. Let  $u_k^2$  denote the intensity of machine maintenance effort. Suppose it takes only two values 0 or 1, the higher values denoting greater maintenance. The idea is that maintenance reduces the likelihood of machine failure and permits a failed machine to become operational.

The effects of these two control actions, intensity of machine use and maintenance, can be modeled as a controlled transition probability matrix as follows

$$P(u^1, u^2) = \begin{bmatrix} 1 - q_1(u^1) + q_2(u^2) & q_1(u^1) - q_2(u^2) \\ q_2(u^2) & 1 - q_2(u^2) \end{bmatrix} \quad (2.1.3)$$

The values of  $q$  are such that  $q_1(0) < q_1(1) < q_1(2)$  and  $q_2(0) < q_2(1)$  because a lightly used or better maintained machine is less likely to fail than a heavily used or less maintained machine.

Equation (2.1.3) is illustrated in the state transition diagram below

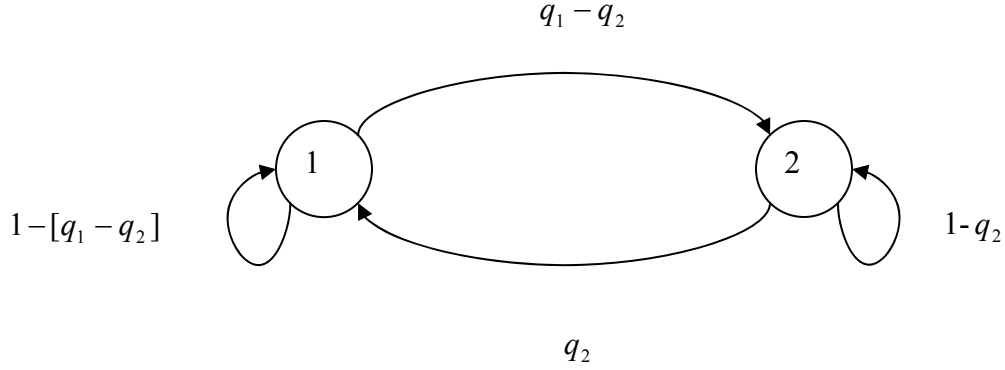


Fig. 2.1.1 State transition diagram

The values of  $q$  are such that  $q_1(0) < q_1(1) < q_1(2)$  and  $q_2(0) < q_2(1)$  because a lightly used or better maintained machine is less likely to fail than a heavily used or less maintained machine. Suppose the state is observed and consider a feedback policy  $\{g_0, g_1, \dots\}$  which is time invariant, that is,  $g_k \equiv g$  and let  $u_k = g(x_k)$ . This results in transition probability matrix  $P^g = \{P_{ij}^g\}$  where

$$P_{ij}^g := P_{ij}(g(i)), \quad i, j \in \{1, 2\}$$

For example, if  $g(1) = (2, 0)$ , and  $g(2) = (0, 1)$ , then

$$P^g = \begin{bmatrix} 1 - q_1(2) + q_2(0) & q_1(2) - q_2(0) \\ q_2(1) & 1 - q_2(2) \end{bmatrix}$$

The resulting process  $\{x_k\}$  is a Markov chain with stationary transition probability  $P^g$ .

The probability distribution of  $x_k$  can be written as a row vector

$$p_k := (\text{prob}\{x_k = 1\}, \text{prob}\{x_k = 2\}).$$

By the Markov property

$$p_{k+m} = p_k [P^g]^m, \quad m \geq 0 \tag{2.1.4}$$

And, in particular  $p_k = p_0[P^g]^k$ , where  $p_0$  is the initial distribution of  $x_0$ .

Often as  $k \rightarrow \infty$ ,  $p_k$  converges to a probability distribution that does not depend on the initial distribution  $p_0$ . We then say that it is an ergodic chain. The limiting probability distribution is called the steady state distribution. It is the solution of the linear equations

$$p = pP^g, \quad p(1) + p(2) = 1 \quad (2.1.5)$$

Equation 2.1.5 always has a solution. In the ergodic case the solution is unique and the limiting distribution has the following interpretation:

$$p(i) = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{k=1}^n I(x_k = i) \quad \text{wp1} \quad (2.1.6)$$

where  $I$  is the indicator function and *wp1* stand for probability.

From (2.1.5) it is evident that the steady state probability  $p$  depends on the feedback law  $g$ . So by changing the policy  $g$ , that is, by changing the use and maintenance of the machine, we can alter the number of times it fails. Now the question is which policy leads to the best probability distribution. We will examine this in the next section.

## 2.2 Finite State Controlled Markov Chain

The preceding example generalizes to the case of an arbitrary finite state controlled Markov chain whose state  $x_k$  takes values in  $\{1, 2, \dots, I\}$ . The control  $u_k$  takes values in a prespecified set  $U$ .  $U$  may be finite or infinite. The transition probability are specified by the  $I \times I$  matrix valued function on  $U$ ,

$$u \rightarrow P(u) := \{P_{i,j}(u), i \geq 1, j \geq 1\}, \quad (2.1.7)$$

With the interpretation that

$$\text{prob}\{x_{k+1} = j / x_k = i, x_{k-1}, \dots, x_0, u_k, \dots, u_0\} = P_{ij}(u_k) \quad (2.1.8)$$

Suppose the state is observed. Let  $g = \{g_0, g_1, \dots\}$  be a feedback policy such that  $g_k$  depends only on the current state  $x_k$ . We call such  $g$  a stationary Markov policy.

Let  $g$  be a Markov policy and let  $x_k$  be the resulting state process. Denote the probability distribution of  $x_k$  by  $I$  dimensional row vector.

$$p_k^g := (\text{prob}\{x_k = 1\}, \dots, \text{prob}\{x_k = I\}),$$

Some basic concepts are introduced, understanding of which is essential to determine the optimal Markov policy

**Lemma 1:** *When a Markov policy is employed, the resulting state process  $x_k$  is Markov process. Its one step transition probability at time  $k$  is given by the matrix*

$$P_k^g := \{(P_k^g)_{ij} := P_{ij}(g_k(i)), i \geq 1, j \geq 1\},$$

*Its  $m$ -step transition probability at time  $k$  is given by the matrix*

$$P_k^g \dots P_{k+m-1}^g,$$

*So its  $ij$ th element is the probability that the state will be  $j$  at time  $k+m$  given that it is  $I$  at time  $k$ . Hence*

$$p_{k+m} = p_k P_k^g \dots P_{k+m-1}^g.$$

*In particular,*

$$p_k = p_0 P_0^g \dots P_{k-1}^g,$$

*where  $p_0$  is the probability distribution of the initial state  $x_0$ .*

**Proof:** The proof is immediate from the Markov property (2.1.8)

Since the transition probability matrix  $P_k^g$  depends on the time  $k$ , we say that  $x_k$  is a Markov chain with nonstationary transition probability.  $\square$

A Markov policy  $g$  determines the probability distribution of the state process  $x_k$  and the control process  $u_k = g_k(x_k)$ . Different policies will lead to different probability distribution. In optimal control problems one is interested in finding the best or optimal policies. This is done by specifying a cost function. This is a sequence of real valued functions of the state and control,

$$C_k(i, u), 1 \leq i \leq I, u \in U, k \geq 0$$

The interpretation is that  $C_k(i, u)$  is the cost to be paid if at time  $k$ ,  $x_k = i$  and  $u_k = u$

If the policy is fixed then the cost over a horizon “ $N$ ” is  $\sum_{k=0}^N C_k(x_k, u_k)$ , which a random variable. The expected cost is given by

$$\begin{aligned} J(g) &= E^g \left\{ \sum_{k=0}^N C_k(x_k, u_k) \right\} \\ &= E^g \left\{ \sum_{k=0}^N C_k(x_k, g_k(x_k)) \right\} \tag{2.1.9} \\ &= \sum_{k=0}^N E \{ C_k(x_k, g_k(x_k)) \} \\ &= \sum_{k=0}^N \sum_{i=1}^I p[x_k = i] C_k(i, g_k(i)) \\ &= \sum_{k=0}^N [p\{x_k = 1\} \dots \dots \dots p\{x_k = I\}] \begin{bmatrix} C_k(1, g_k(1)) \\ \vdots \\ C_k(I, g_k(I)) \end{bmatrix} \end{aligned}$$

$$\begin{aligned}
&= \sum_{k=0}^N p_k C_k^g \\
&= \sum_{K=0}^N p_0 (p_0^g p_1^g \dots p_{k-1}^g) C_k^g
\end{aligned}$$

The best policy is the one that minimizes the expected cost. The solution approach is dynamic programming. Dynamic programming will be discussed in detail in the next section. Central to dynamic programming is a recursive technique for calculating the cost of a Markov policy  $g$ .

Since the technique depends only on the fact that the state process corresponding to  $g$  is Markov, we introduce it here. For each time  $1 \leq k \leq N$ , and state  $1 \leq i \leq I$ , let  $V_k^g(i)$  denote the expected cost incurred during  $k, \dots, N$  when  $x_k = i$ . That is,

$$V_k^g(i) := E^g \left\{ \sum_{l=k}^N c_l(x_l, g_l(x_l)) / x_k = i \right\}. \quad (2.1.10)$$

**Lemma 2:** The functions  $V_k^g(i)$  can be calculated by the backward recursion,

$$V_k^g(i) = c_k(i, g_k(i)) + \sum_{j=1}^I P_k^g{}_{ij} V_{k+1}^g(j), \quad 0 \leq k < N, \quad (2.1.11)$$

Starting with the final condition

$$V_N^g(i) = c_N(i, g_N(i)) \quad (2.1.12)$$

**Proof:** From the definition we immediately get (2.1.12). Next

$$\begin{aligned}
V_k^g(i) &= E^g \left\{ \sum_{l=k}^N c_l(x_l, g_l(x_l)) / x_k = i \right\} \\
&= c_k(i, g_k(i)) + E^g \left\{ E^g \left\{ \sum_{l=k+1}^N c_l(x_l, g_l(x_l)) / x_{k+1}, x_k = i \right\} / x_k = i \right\} \\
&= c_k(i, g_k(i)) + E^g \{ V_{k+1}^g(x_{k+1}) / x_k = i \}
\end{aligned}$$

$$= c_k(i, g_k(i)) + \sum_{j=1}^I V_{k+1}^g(j) \text{Pr ob}\{x_{k+1} = j / x_k = i\} \quad (2.1.13) \square$$

### 2.2.1 Dynamic Programming

We consider the problem of selecting a feedback control so as to minimize the expected cost. The optimality conditions are obtained by dynamic programming. Consider the stochastic system described by the state space model

$$\begin{aligned} x_{k+1} &= f_k(x_k, u_k, w_k), \\ y_k &= h_k(x_k, v_k), \quad k = 0, 1, \dots \end{aligned}$$

Suppose that for each  $k$ , the control value  $u_k$  is to be selected from a prespecified control set  $U \subset R^m$ . A feasible control law is any sequence  $g = \{g_0, g_1, \dots\}$  such that

$$u_k = g_k(y^k) \in U \quad \text{for all } y^k.$$

Let  $G$  denote the set of all feasible control laws. Suppose that we are given a cost function

$$\sum_{k=0}^{N-1} c_k(x_k, u_k) + c_N(x_N)$$

where  $c_k(x_k, u_k)$  is called the immediate or one-period cost, and  $c_N(x_N)$  is the terminal cost. Since  $N < \infty$ , a control law is now specified by the finite sequence  $g = \{g_0, g_1, \dots, g_{N-1}\}$ .

Let  $g$  be a feasible law. Let  $x^g, y^g$  and  $u^g$  denote the processes corresponding to it. By definition the cost associated with  $g$  is

$$C^g = \sum_0^{N-1} c_k(x_k^g, u_k^g) + c_N(x_N^g) \quad (2.1.14)$$

**Definition 1:** A Markov policy  $g^*$  in  $G$  is optimal if  $J(g^*) = J^* = \text{Inf}\{J(g) / g \in G\}$ . Thus  $J^*$  is called the minimum expected cost. Our aim is to characterize the optimal control laws using dynamic programming.

**Lemma 3:** Let  $g = \{g_0, g_1, \dots, g_{N-1}\}$  be a Markov policy. Define recursively the functions

$$V_N^g(x) := c_N(x), \quad (2.1.15)$$

$$V_k^g(x) := c_k(x, g_k(x)) + E_{w_k} V_{k+1}^g[f_k(x, g_k(x), w_k)] \quad (2.1.16)$$

Then the random variable  $V_k^g(x_k^g)$  satisfies

$$V_k^g(x_k^g) = E\left\{\sum_{l=k}^{N-1} c_l(x_l^g, u_l^g) + c_N(x_N^g) / x_k^g\right\}, k = 0, \dots, N \quad (2.1.17)$$

The proof is very simple and hence it is not discussed here. Since  $\{x_k\}$  is a Markov, we also have

$$V_k^g(x_k^g) = E\left\{\sum c_l(x_l^g, u_l^g) + c_N(x_N^g) / x_k^g, \dots, x_0^g\right\} \quad (2.1.18) \square$$

**Definition 2:** Let  $g \in G$ . The random variable

$$J_k^g = E\left\{\sum c_l(x_l^g, u_l^g) + c_N(x_N^g) / x_0^g, \dots, x_k^g\right\} \quad (2.1.19)$$

is called the cost-to-go at  $k$  corresponding to  $g$ .

From definition (2.1.19) we see that

$$J_o^g = E\{C^g / x_0\}, \quad J(g) = EJ_o^g \quad (2.1.20)$$

While lemma (3) is valid only for Markov policies, the comparison principle below holds for arbitrary feedback policies

**Lemma 4(Comparison Principle):** Let  $V_k(x)$ ,  $0 \leq k \leq N$ , be any functions such that

$$V_N(x) \leq c_N(x), \quad (2.1.21)$$

$$V_k(x) \leq c_k(x, u) + E_{w_k} V_{k+1}[f_k(x, u, w_k)], \quad (2.1.22)$$

for all  $x$  and for all  $u$ . Let  $g \in G$  be arbitrary. Then, w.p.1

$$V_k(x_k^g) \leq J_k^g, \quad k = 0, \dots, N \quad (2.1.23)$$

**Proof:** We proceed by induction. From (2.1.19) to (2.1.21)

$$J_N^g = E\{c_N(x_N^g / x_o^g, \dots, x_N^g) / x_o^g, \dots, x_N^g\} = c_N(x_N^g) \geq V_N(x_N^g),$$

so that (2.1.23) is true for  $k=N$ . Suppose it is true for  $k+1$ . Then by (2.1.22)

$$\begin{aligned} V_k(x_k^g) &\leq E\{c_k(x_k^g, u_k^g) + V_{k+1}^g[f_k(x_k^g, u_k^g, w_k)] / x_o^g, \dots, x_k^g\} \\ &\leq E\{c_k(x_k^g, u_k^g) + E\{\sum_{l=k+1}^{N-1} c_l(x_l^g, u_l^g) + c_N(x_N^g) / x_o^g, \dots, x_{k+1}^g\} / x_o^g, \dots, x_k^g\} \\ &= E\{\sum_k^{N-1} c_l(x_l^g, u_l^g) + c_N(x_N^g) / x_o^g, \dots, x_k^g\} = J_k^g, \end{aligned}$$

and so (2.1.23) holds for  $k$ . □

We get an immediate corollary.

### **Corollary1**

Let  $V_k(x)$  be a function satisfying (2.1.21), (2.1.22). Then  $J^* \geq EV_o(x_o)$ . Hence if  $g \in G$  is such that  $J_o^g = V_o(x_o)$ , then  $g$  is optimal.

**Proof:** For any  $g$  in  $G$  we have  $J_o^g \geq V_o(x_o)$  by (2.1.23). Taking expectation  $J(g) \geq EV_o(x_o)$  and  $J^* \geq EV_o(x_o)$ . Finally, if  $J_o^g = V_o(x_o)$ , then  $J(g) = EV_o(x_o) \leq J^*$ , so that  $g$  must be optimal and  $J(g) = J^*$  □

The two preceding lemmas can be combined to obtain the fundamental result of dynamic programming.

**Theorem:** Define recursively the functions

$$V_N(x) = c_N(x) \quad (2.1.24)$$

$$V_k(x) = \text{Inf}_{u \in U} [c_k(x, u) + E_{w_k} V_{k+1}(f_k(x, u, w_k))] \quad (2.1.25)$$

1. Let  $g$  in  $G$  is arbitrary. Then  $V_k(x_k^g) \leq J_k^g$  w.p.1; in particular,  $J(g) \geq EV_o(x_o)$ .
2. A Markov policy  $g = \{g_0, g_1, \dots, g_{N-1}\}$  in  $G_M$  is optimal if the infimum in (2.1.24) is achieved at  $g_k(x)$ , and then  $V_k(x_k^g) = J_k^g$  w.p.1 and  $J^* = J(g) = EV_o(x_o)$ .
3. A Markov policy  $g = \{g_0, g_1, \dots, g_{N-1}\}$  in  $G_M$  is optimal if for each  $k$ , the infimum at  $x_k^g$  in (2.1.24) is achieved by  $g_k(x_k^g)$ , i.e

$$V_k(x_k^g) = c_k(x_k^g, g_k(x_k^g)) + E_{w_k} V_{k+1}[f_k(x_k^g, g_k(x_k^g), w_k)],$$

w. p. 1.

**Proof:** The functions  $V_k(x)$  defined by (2.1.24) and (2.1.25) clearly satisfy (2.1.21) and (2.1.22) and so part (1) follows from lemma. To prove the sufficiency in part (2), let  $g = \{g_k\}$  be a Markov policy that achieves the infimum in (2.1.25), so

$$V_k(x) = c_k(x, g_k(x)) + E_{w_k} V_{k+1}[f_k(x, g_k(x), w_k)]$$

By lemma it follows that  $V_k(x_k^g) = J_k^g$  for all  $k$  and in particular  $J_o^g = V_o(x_o)$ . By corollary,  $g$  is optimal and  $J^* = J(g) = EV_o(x_o)$ .

To prove the necessity in part (3) suppose the Markovian policy  $g$  is optimal. We prove by induction that  $g_k(x_k^g)$  achieves the infimum in (2.1.25) at  $x_k^g$  with probability 1.

Consider  $k=N-1$ . Suppose the assertion is false. Then there exist another function

$g'_{N-1} : R^n \rightarrow U$  such that

$$\begin{aligned} c_{N-1}(x_{N-1}, g_{N-1}(x_{N-1}g)) + E_{w_{n-1}} V_N[f_{N-1}(x_{N-1}^g, g_{N-1}(x_{N-1}^g), w_{N-1})] \\ \geq c_{N-1}(x_{N-1}, g'_{N-1}(x_{N-1}g)) + E_{w_{n-1}} V_N[f_{N-1}(x_{N-1}^g, g'_{N-1}(x_{N-1}^g), w_{N-1})] \end{aligned}$$

w.p.1; moreover, the inequality is strict with positive probability. Hence taking expectations on both sides and using (2.1.24) gives

$$\begin{aligned} E[c_{N-1}(x_{N-1}, g_{N-1}(x_{N-1}g)) + c_N(f_{N-1}(x_{N-1}^g, g_{N-1}(x_{N-1}^g), w_{N-1}))] \\ > E[c_{N-1}(x_{N-1}, g'_{N-1}(x_{N-1}g)) + c_N(f_{N-1}(x_{N-1}^g, g'_{N-1}(x_{N-1}^g), w_{N-1}))] \end{aligned} \quad (2.1.26)$$

Consider the Markov policy  $g' = \{g'_0, \dots, g'_{N-2}, g'_{N-1}\}$ . Evidently  $x_k^g = x_k^{g'}$ ,  $0 \leq k \leq N-1$

and so  $u_k^g = u_k^{g'}$ ,  $0 \leq k \leq N-1$ ,  $u_{N-1}^{g'} = g'_{N-1}(x_{N-1}^g)$ . It follows that

$$Ec_k(x_k^g, u_k^g) = Ec_k(x_k^{g'}, u_k^{g'}), \quad 0 \leq k \leq N-2 \quad (2.1.27)$$

Adding (2.1.26) and (2.1.27) gives  $J(g) > J(g')$  and so  $g$  cannot be optimal contrary to the hypothesis. Thus  $g_{N-1}(x_{N-1}^g)$  does achieve the infimum in (2.1.25) for  $N-1$ , and so  $J_{N-1}^g = V_{N-1}(x_{N-1}^g)$ .

Now suppose by induction that  $g_{k+1}(x_{k+1}^g)$  achieves the infimum and that  $J_{k+1}g = V_{k+1}(x_{k+1}^g)$ . We prove this for  $k$ . Indeed, otherwise there is a function

$g'_k : R^n \rightarrow U$  such that

$$\begin{aligned} c_k(x_k^g, g_k(x_k^g)) + E_{w_k} V_{k+1}[f_k(x_k^g, g_k(x_k^g), w_k)] \\ \geq c_k(x_k^g, g'_k(x_k^g)) + E_{w_k} V_{k+1}[f_k(x_k^g, g'_k(x_k^g), w_k)] \end{aligned}$$

w.p.1. This inequality is strict with positive probability so that taking expectations gives

$$\begin{aligned}
& Ec_k(x_k^g, g_k(x_k^g)) + E_{w_k} V_{k+1}(x_{k+1}^g) \\
& > Ec_k(x_k^g, g'_k(x_k^g)) + EV_{k+1}(f_k(x_k^g, g'_k(x_k^g), w_k))
\end{aligned} \tag{2.1.28}$$

Consider the policy  $g' = \{g_0, \dots, g_{k-1}, g'_k, g_{k+1}, \dots, g_{N-1}\}$ . Then certainly

$$Ec_l(x_l^g, u_l^g) = Ec_l(x_l^{g'}, u_l^{g'}), \quad l = 0, \dots, k-1, \tag{2.1.29}$$

Also, by the induction hypothesis,  $g_{k+1}, \dots, g_{N-1}$  achieve the infimum in (2.1.28), and so

by lemma 4

$$\begin{aligned}
& EJ_{k+1}^g(x_{k+1}^g) = EV_{k+1}(x_{k+1}^g), \text{ and} \\
& EJ_{k+1}^{g'}(x_{k+1}^{g'}) = EV_{k+1}(x_{k+1}^{g'})
\end{aligned} \tag{2.1.30}$$

From (2.1.28), (2.1.29), and (2.1.30) it follows that

$$\begin{aligned}
J(g) &= E \sum c_l(x_l^g, u_l^g) + Ec_k(x_k^g, u_k^g) + V_{k+1}(x_{k+1}^g) \\
&> E \sum c_l(x_l^g, u_l^g) + Ec_k(x_k^g, u_k^{g'}) + V_{k+1}(x_{k+1}^{g'}) \\
&= J(g')
\end{aligned}$$

and so  $g$  cannot be optimal contrary to hypothesis. Thus  $g_k(x_k^g)$  must achieve the infimum in (2.1.28) and the result follows by induction.  $\square$

### 2.2.2 Algorithm

First, define  $V_N(x) = C_N(x)$ ,

Then find the function  $g_{N-1} : R^n \rightarrow U$  by

$$g_{N-1}(x) = \arg \inf [C_{N-1}(x, u) + E_{w_{N-1}} V_N(f_{N-1}(x, u, w_{N-1})) / u \in U],$$

and denote the resulting value  $V_{N-1}(x)$

Second, find the function  $g_{N-2} : R^n \rightarrow U$  by

$$g_{N-2}(x) = \arg \inf [C_{N-2}(x, u) + E_{w_{N-2}} V_N(f_{N-2}(x, u, w_{N-2})) / u \in U],$$

and call this resulting value  $V_{N-2}(x)$ . Proceeding in this way we obtain  $g_{N-1}, V_{N-1}, \dots, g_0, V_0$ .

## 2.3 Infinite Horizon

Let us consider when time horizon  $N$  is infinite. This is not an immediate extension, since if one simply sets  $N = \infty$  in (2.1.9), in most cases one gets  $J(g) = \infty$  for every  $g$ . The notion of best  $g$  becomes meaningless. There are two ways to treat the infinite horizon problem. The first approach is to introduce a discount factor  $\beta$ ,  $0 < \beta < 1$  and define the expected discounted cost.

$$J(g) = E^g \sum_{k=0}^{\infty} \beta^k c_k(x_k, u_k)$$

Observe that if  $c_k$  is bounded, and then  $J(g)$  will be finite. Since the cost incurred at time  $k$  is weighted by  $\beta^k$ , present cost is more important than future cost. In an economic context,  $\beta = (1+r)^{-1}$  where  $r > 0$  is the interest rate. With this interpretation,  $J(g)$  is present value of cost. From (2.1.9) it follows that

$$J(g) = \sum_{k=0}^{\infty} \beta^k p_0(p_0^g p_1^g \dots p_{k-1}^g) C_k^g$$

Define

$$V_k^g(i) := E^g \left\{ \sum_{l=k}^{\infty} \beta^l c_l(x_l, g_l(x_l)) / x_k = i \right\}$$

The second approach is followed when discounting is inappropriate. A policy is then evaluated according to its average cost per unit time,

$$J(g) = \lim_{N \rightarrow \infty} \frac{1}{N} E^g \sum_{k=0}^{N-1} c_k(x_k, g_k(x_k)) \quad (2.1.31)$$

Using (2.1.9) this equals

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=0}^{N-1} p_0(P_o^g \dots P_{k-1}^g) c_k^g \quad (2.1.32)$$

From this expression we see that if  $P_k^g$  varies with  $k$ , then the limit above need not exist.

### 2.3.1 Stationary Markov Policy

A Markov policy  $g = \{g_0, g_1, \dots\}$  is stationary or time invariant if  $g_0 \equiv g_1 \equiv \dots \equiv g$ , with a slight abuse of notation. Let  $g$  be stationary; then the transition probability matrix is stationary,  $P_k^g \equiv P^g$ . Suppose the cost functions are also time invariant,  $c_k \equiv c$ . Fix a discount  $0 < \beta < 1$ . Then

$$\begin{aligned} V_k^g(i) &= E^g \left\{ \sum_{l=k}^{\infty} \beta^l c(x_l, g(x_l)) / x_k = i \right\} \\ &= \beta^k E^g \left\{ \sum_{l=0}^{\infty} \beta^l c(x_l, g(x_l)) / x_0 = i \right\} \\ &= \beta^k V_o^g(i) \end{aligned}$$

In vector notation  $\beta^k V_o^g = \beta^k c^g + \beta^{k+1} P^g V_o^g$ , or in matrix notation,

$$[I - \beta P^g] V_o^g = c^g$$

Next we study the average cost when policy and cost are stationary. The next three lemmas are stated without proof.

**Lemma 5:** *If  $P$  is the transition probability matrix, then the Cesaro limit*

$$\lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=0}^{N-1} P^k =: \Pi,$$

always exists. The matrix  $\Pi$  is a stochastic matrix and it satisfies the equation  $\Pi = \Pi P$ . Thus for stationary  $g$  and time invariant cost, the average cost per unit time (2.1.31) and (2.1.32), is

$$J(g) = \lim_{N \rightarrow \infty} \frac{1}{N} E^g \sum_{k=0}^{N-1} c(x_k, g(x_k)) = p_0 \Pi c^g \quad (2.1.33)$$

Let  $\pi$  be one of the rows of  $\Pi$ . Then  $\pi = \pi P$

Moreover since  $\Pi$  is a stochastic matrix,  $\pi$  can be regarded as a probability distribution. This has an interpretation that if Markov chain has initial probability distribution given by  $\pi$ , then the probability distribution of the state remains at  $\pi$  for all time. Thus  $\pi$  is said to be an invariant probability distribution.

**Lemma 5:** If  $P$  is an irreducible transition probability matrix, then there is a unique row vector  $\pi$  such that

$$\pi P = \pi, \quad \sum_{i=1}^I \pi_i = 1. \quad (2.1.34)$$

Moreover  $\pi_i > 0$ , for all  $i$ . finally, the matrix  $\Pi$  in (1.33) has all rows equal to  $\pi$   $\square$

**Lemma 6:** If  $P$  is irreducible and aperiodic, then

$$\lim_{k \rightarrow \infty} P^k = \Pi \quad (2.1.35)$$

Where  $\Pi$  is a matrix with all rows equal to  $\pi$   $\square$

Consider first the problem of minimizing the finite horizon cost

$$E^g \sum_{k=0}^{N-1} \beta^k x(x_k, u_k) \quad (2.1.36)$$

with  $0 < \beta < 1$ . We have the optimal value function  $V_k(i)$  satisfying the dynamic programming equations,

$$V_n(i) = 0, \quad (2.1.37)$$

$$V_k(i) = \inf_{u \in U} \{ \beta^k c(i, u) + \sum_{j=1}^I V_{k+1}(j) P_{ij}(u) \} \quad (2.1.38)$$

$V_k(i)$  is the minimum value of  $E^g \sum_{n=k}^{N-1} \beta^n x(x_n, u_n)$  starting in state  $i$  at time  $k$ . it is more

convenient to work not with  $V_k(i)$ , but the related quantity,

$$W_N(i) := \beta^{n-N} V_{N-n}(i), \quad 0 \leq n \leq N, 1 \leq i \leq I \quad (2.1.39)$$

The index  $n$  is the time to go. It can be shown that

$$W_o(i) := 0, \quad (2.1.40)$$

$$W_n(i) := \inf_{u \in U} \{ c(i, u) + \beta \sum_{j=1}^I W_{n-1}(j) P_{ij}(u) \} \quad (2.1.41)$$

**Lemma 7:** Let  $W_n(i) := \min_g E^g \sum_{k=0}^{n-1} \beta^k x(x_k, u_k) / x_0 = i$ . Then  $W_n(i)$  satisfies (2.1.40) and

(2.1.41)

For the discounted cost  $E^g \sum_{k=0}^{N-1} \beta^k x(x_k, u_k)$  it is natural to expect that its minimum value starting in state  $x_0 = i$ , denoted by  $W_\infty(i)$ , will satisfy (2.1.41) with  $\infty$  replacing  $n$ .

so it is plausible that  $W_\infty$  will satisfy

$$W_\infty(i) = \inf_{u \in U} \{ c(i, u) + \beta \sum_{j=1}^I W_\infty(j) P_{ij}(u) \}, \text{ for all } i \quad (2.1.42) \quad \square$$

This is the dynamic programming equation for the discounted cost criterion. Equation (2.1.42) is a system of  $I$  equations with  $I$  unknowns  $W_\infty(1), \dots, W_\infty(I)$ . Next we examine

the existence and uniqueness of solutions to these equations, and show that the solution indeed gives the value function.

### 2.3.2 Value Iteration

In value iteration method optimal policies are determined sequentially using the functional equations

$$f_i(n) = \max_a \left\{ r_i^a + \beta \sum_{j=1}^n p_{ij}^a f_j(n-1) \right\}$$

where the action  $a$  maximizing the right hand side is optimal for state  $i$  at the stage in question. The function  $f_i(n)$  is the total expected discounted rewards from the process when it starts from state  $i$  and will operate for  $n$  stages before termination. Thus  $f_i(0)$  is the salvage value of the system when it is in state  $i$ . At each stage an optimal value is chosen using the above equation. The value iteration method is identical to what is usually referred to as successive iteration or successive approximation. We are not going to discuss this method in detail since the value iteration method is not exact and the convergence is slow.

### 2.3.3 Linear Programming

In infinite horizon case we have to use linear programming instead of dynamic programming. Therefore to find the optimal policy, we can convert the problem into a linear program and solve it using simplex method.

It can be shown that if  $u$  is a bounded function on the state space satisfying:

$$u(i) \leq \min_a \left[ c(i, a) + \sum_j P_{ij}(a) u(j) \right] \quad (2.1.43)$$

then  $u \leq V$  with  $V$  the optimal value function.

Because the optimal value function  $V$  satisfies the inequality, as well as the equality, it follows that it is the largest function that satisfies (2.1.43). Hence, letting  $\beta_i$  be such that  $0 < \beta_i < 1$ , it follows that  $V$  will be the unique solution of the linear program.

$$\max_u \left[ \sum_{i=0}^m \beta_i u(i) \right]$$

subject to

$$u(i) \leq \min \left[ c(i, a) + \alpha \sum_j P_{ij}(a) u(j) \right], \text{ for all } i.$$

We can take  $\beta_i$  as 1 for all  $i$  as  $\beta_i$  is chosen to prevent the objective function from becoming infinite. After this assumption, our linear problem takes this form

$$\max_u \left[ \sum_{i=0}^m u(i) \right] \tag{2.1.44}$$

subject to

$$u(i) \leq \min \left[ c(i, a) + \alpha \sum_j P_{ij}(a) r(j) \right], \text{ for all } i \text{ and all } u \in U \tag{2.1.45}$$

Objective function (2.1.44) and constraint (2.1.45) allows the use of first linear program to calculate optimal policy. This model is called short term or action selection model. Solving this problem would give the optimal values of  $u(i) = V(i)$  for all  $i$ , with the requirement that  $u$  is the optimal action associated with the state if and only if the corresponding constraint in (2.1.45) holds as equality.

After the solution to the linear program (2.1.44)-(2.1.45) has been obtained, we calculate the shadow costs of the actions. These costs are calculated by substituting the optimal values of  $u(i)$  into the right hand side of the constraints (2.1.45):

$$V(i, a) = c(i, a) + \alpha \sum_j P_{ij}(a) u(j) \quad (2.1.46)$$

The optimal action  $a^*$  in each state is chosen from the condition:

$$V(i, a^*) = \min_a [V(i, a)] \quad (2.1.47)$$

The difference that the shadow cost of the optimal action makes with the shadow cost of the do-nothing action is interpreted as the unit shadow benefit of taking the optimal action:

$$B(i, a^*) = V(i, 0) - V(i, a^*) \quad (2.1.48)$$

It follows from (2.1.48) that when the optimal action is do-nothing, its shadow benefit becomes zero.

The above model specifies a stationary policy that is optimal and can be used immediately, whether the process is stationary or not, and hence it is the model, with which we can determine the recommended actions for each condition state of every element. However, this model would not directly give the long term conditions of the elements. To calculate the interest another model is used;

$$\min \sum_i \sum_a w_{ia} g(i, a) \quad (2.1.49)$$

$$\text{subject to } \sum_i \sum_a w_{ia} = 1 \quad (2.1.50)$$

$$\sum_a w_{ja} = \sum_i \sum_a w_{ia} P_{ij}(a), \text{ for all } j \quad (2.1.51)$$

where  $w_{ia}$  denote the limiting probability that the element will be in state  $i$  and action  $a$  is chosen when policy  $g$  is followed

$$w_{ia} \geq 0 \quad (2.1.46)$$

Thus for any policy  $g$ , there is a vector  $w = w_{ia}$  which satisfies (2.1.49), (2.1.50) and (2.1.51) with the interpretation that  $w_{ia}$  equals the steady state probability of being in state  $i$  and choosing action  $a$  if  $g$  is employed.

The values of  $w_{ia}$  can be interpreted as the proportion of time that the unit element is in condition state  $i$  and action  $a$  is taken, or alternatively, the proportion of units for which action  $a$  is taken in condition state  $i$ . The linear problem defined above is called long-term or steady state model.

## **Chapter 3**

### **Fuzzy Mathematics**

In this chapter we will discuss basic results in Fuzzy theory, from the existing literature. This chapter is divided into three sections. In the first section we discuss preliminary elements of fuzzy mathematics. In the second section we discuss some operations on fuzzy sets. In the third section we give an introduction to fuzzy control design. Here we introduce concept of fuzzy membership functions which will be used in the next chapter to reduce the number of states in a Markov decision process.

The field of fuzzy systems and control has been well developed about one decade ago. Motivated by the practical success of fuzzy control in consumer products and industrial process control, there has been an increasing amount of work on the rigorous theoretical studies of fuzzy systems and fuzzy control. Researchers are trying to explain why the practical results are good, to systematize the existing approaches, and to develop more powerful design tools.

Fuzzy systems are knowledge-based or rule based systems. The heart of the fuzzy system is a knowledge base consisting of the so called fuzzy IF-THEN rule. A fuzzy IF-THEN rule is an IF-THEN statement in which some words are characterized by continuous membership functions. For example the following is a fuzzy IF –THEN rule:

“IF the speed of a car is high, THEN apply less force to the accelerator”

A fuzzy system is constructed from collection of fuzzy IF-THEN rules. The basic configuration of a pure fuzzy system is shown in the Fig.2.1. The “Fuzzy Rule Based” represents the collection of fuzzy IF-THEN rules. The fuzzy inference engine combines these fuzzy IF-THEN rules into a mapping from fuzzy sets in the input space  $U \subset R^n$  to fuzzy sets in the output space  $V \subset R$  based on fuzzy logic principles. If the dashed feedback line in Fig.3.1.1 exists, the system becomes the so called fuzzy dynamic systems.

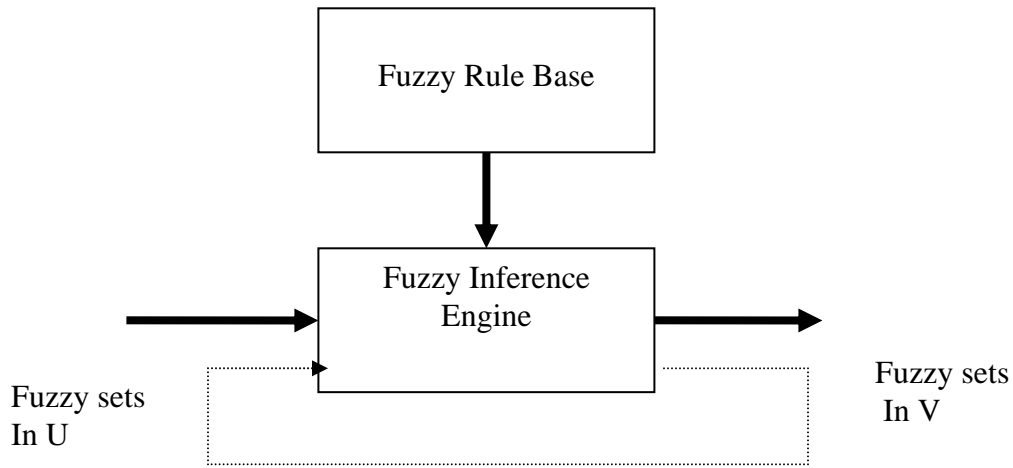


Figure 3.1.1. Basic configuration of pure fuzzy system.

### 3.1 Elements of Fuzzy Mathematics

**Fuzzy Set:** A fuzzy set in a universe of discourse  $U$  is characterized by a membership function  $\mu_A(x)$  that takes values in the interval  $[0, 1]$ . Therefore, a fuzzy set is a generalization of a classical set by allowing the membership function to take any value in the interval  $[0, 1]$ . A fuzzy set  $A$  in  $U$  may be represented as a set of ordered pairs of a generic element  $x$  and its membership value, that is,

$$A = \{(x, \mu_A(x)) / x \in U\}$$

**Support:** The support of a fuzzy set A in the universe of discourse U is a crisp set that contains all the elements of U that have nonzero membership values in A, that is,

$$Supp(A) = \{x \in U / \mu_A(x) > 0\}$$

**Center:** If the mean value of all points at which the membership function of the fuzzy set achieves its maximum value is finite, then define this mean value as the center of the fuzzy set.

**Height:** The height of a fuzzy set is the largest membership value attained by any point. If the height of a fuzzy set equals one, it is called a normal fuzzy set.

**$\alpha$ -cut:** An  $\alpha$ -cut of a fuzzy set A is a crisp set  $A_\alpha$  that contains all the elements in U that have membership values in A greater than or equal to  $\alpha$ , that is,

$$A_\alpha = \{x \in U / \mu_A(x) \geq \alpha\}$$

### 3.2 Operations on Fuzzy Sets

**Complement:** Let  $c: [0, 1] \rightarrow [0, 1]$  be a mapping that transform the membership function of fuzzy set A into the membership function of the complement of A, that is,

$$c[\mu_A(x)] = \mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

**Union:** Let  $s: [0, 1] \times [0, 1] \rightarrow [0, 1]$  be a mapping that transforms the membership functions of fuzzy sets A and B into the membership function of the union of A and B, that is,

$$s[\mu_A(x), \mu_B(x)] = \mu_{A \cup B}(x)$$

In order for the function to be qualified as a union, it must satisfy at least the following four requirements:

Axiom 1:  $s(1, 1) = 1$ ,  $s(0, a) = s(a, 0) = a$ .

Axiom 2:  $s(a, b) = s(b, a)$

Axiom 3: if  $a \leq a'$  and  $b \leq b'$ , then  $s(a, b) \leq s(a', b')$

Axiom 4:  $s(s(a, b), c) = s(a, s(b, c))$

**Intersection:** Let  $t : [0,1] \times [0,1] \rightarrow [0,1]$  be a function that transforms the membership functions of fuzzy sets A and B into the membership function of the intersection of A and B, that is ,

$$t[\mu_A(x), \mu_B(x)] = \mu_{A \cap B}(x)$$

In order for the function  $t$  to be qualified as an intersection, it must satisfy at least the following four requirements:

Axiom 1:  $t(0,0) = 0; t(a,1) = t(1,a) = a$

Axiom 2:  $t(a, b) = t(b, a)$

Axiom 3: If  $a \leq a'$  and  $b \leq b'$ , then  $t(a, b) \leq t(a', b')$

Axiom 4:  $t[t(a, b), c] = t[a, t(b, c)]$

### 3.3 Overview of Design

There are specific components characteristics of a fuzzy controller to support a design procedure. In the block diagram in Fig.3.1.2 the controller is between a preprocessing block and post processing block. The fuzzy controller consists of a fuzzification block, rule base, inference engine and defuzzification block. Fuzzy membership function is used in fuzzification block. The fuzzy membership function is the heart of the fuzzy controller. We have a set of rules in rule base, which are used by inference engine. The defuzzification block converts the output to membership form. Scaling is done in preprocessor and post processor. The following explains the diagram block by block.

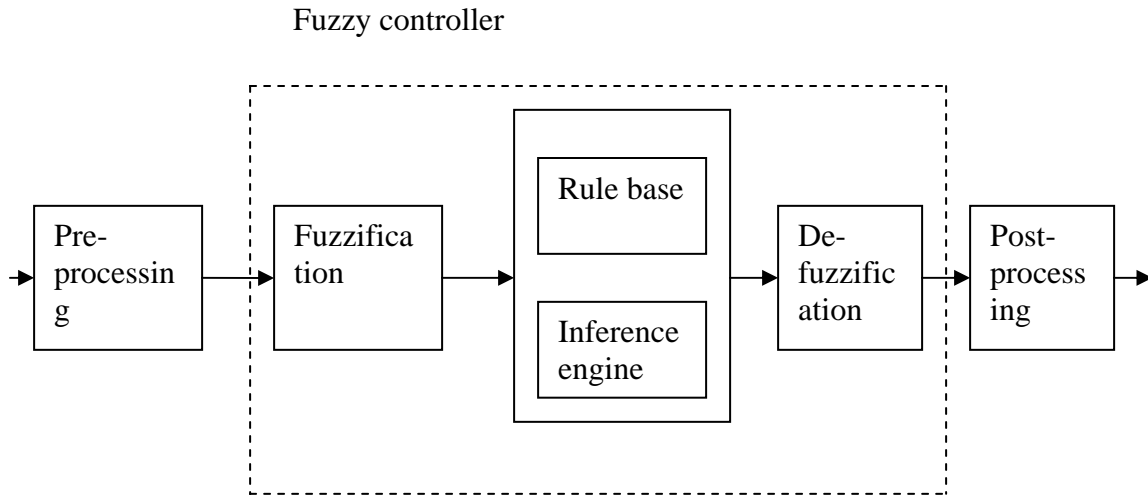


Fig.3.1.2 Blocks of a fuzzy controller

### 3.3.1 Preprocessing

The inputs are most often hard or crisp measurements from some crisp measuring equipment, rather than linguistic. A Preprocessor, conditions the measurements before they enter the controller. Examples of preprocessors are

- Quantization in connection with sampling or rounding to integers;
- Normalization or scaling onto a particular, standard range;
- Filtering in order to remove noise;
- Averaging to obtain short term or long term tendencies;
- A combination of several measurements to obtain key indicators; and
- Differentiation or integration or they discrete equivalent

A quantizer is necessary to convert the incoming values in order to find the best level in discrete universe. Quantization is a means to reduce data, but if the quantization is too coarse the controller may oscillate around the reference or even become unstable.

When the input to the controller is error, the control strategy is static mapping between input and control signal. A dynamic controller would have additional inputs, for example

derivatives, integrals, or previous values of measurement backward in time. These are created in preprocessor thus making the controller multidimensional, which requires many rules and makes it more difficult to design. The preprocessor then passes the data on to the controller.

### 3.3.2 Fuzzification

The first block inside the controller is fuzzification, which converts each piece of input data to degrees of membership function by a lookup in one or several membership functions. The fuzzification block thus matches the input data with the conditions of the rule to determine how well the condition of each rule matches that particular instance. There is a degree of membership for each linguistic term that applies to that input variable.

**1. Singleton Fuzzifier:** The singleton fuzzifier maps a real valued point  $x^* \in U$  into a fuzzy singleton  $A'$  in  $U$ , which has membership value 1 at  $x^*$  and 0 at all other points in  $U$ ; that is,

$$\mu_{A'}(x) = \begin{cases} 1 & \text{If } x=x^* \\ 0 & \text{otherwise} \end{cases}$$

Example: Suppose the thermostat output temperature is  $45^\circ$ , singleton fuzzifier maps it as

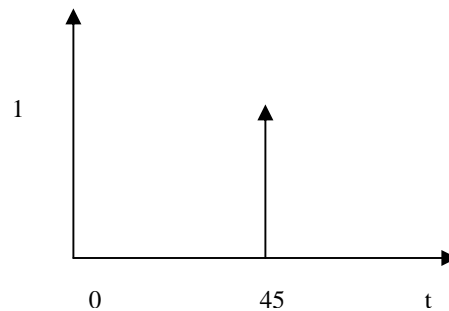


Fig 3.1.3 singleton fuzzifier

**2. Gaussian fuzzifier:** The Gaussian fuzzifier maps  $x^* \in U$  into a fuzzy set  $A'$  in  $U$ , which has the following Gaussian membership function:

$$\mu_{A'}(x) = e^{-\left(\frac{x_1 - x_1^*}{a_1}\right)^2} \otimes \dots \otimes e^{-\left(\frac{x_n - x_n^*}{a_n}\right)^2}$$

Where  $a_i$ , are positive parameters and the t-norm  $\otimes$  is usually chosen as algebraic product or min.

**3. Triangular Fuzzifier:** the triangular fuzzifier maps  $x^* \in U$  into a fuzzy set  $A'$  in  $U$ , which has the following triangular membership function

$$\mu_{A'}(x) = \left\{ \begin{array}{l} \left(1 - \frac{x_1 - x_1^*}{b_1}\right) \otimes \dots \otimes \left(1 - \frac{x_n - x_n^*}{b_n}\right) \text{ if } |x_i - x_i^*| \leq b_i, i = 1, 2, \dots, n \\ 0 \text{ otherwise} \end{array} \right\}$$

Where  $b_i$ , are positive parameters and the t-norm  $\otimes$  is usually chosen as algebraic product or min. Some other fuzzifier

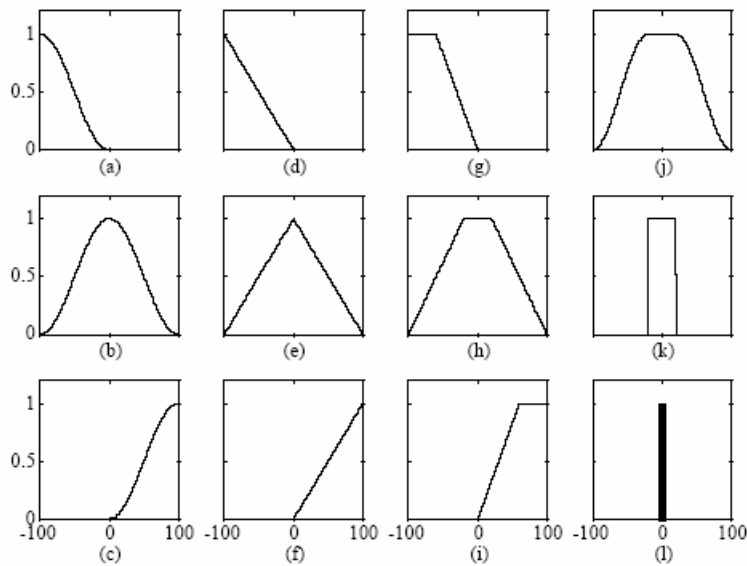


Fig 3.1.4 (a) s-function, (b) pi-function, (c) z-functions, (d-f)triangular versions, (g- i)trapezoidal versions, (j) flat pi-function, (k)rectangle,(l) singleton

### 3.3.3 Rule Base

The rules may use several variables both in condition and conclusion of the rules. The controllers can therefore be applied to both MIMO and SISO problems. The typical SISO problem is to regulate control signal based on an error signal. The controller may actually need the error, change in error, and the accumulated error as the inputs, but we will call it single loop control, because in principle all three are formed from the error measurement. Basically a linguistic controller contains rules in the IF THEN format, but they can be represented in different formats. In many systems, the rules are presented to the end user in the format similar to the one below,

Table 3.1.1 Table based control

| <b>Error</b> | <b>Change in error</b> | <b>Output</b>   |
|--------------|------------------------|-----------------|
| Negative     | Positive               | Zero            |
| Negative     | Zero                   | Negative medium |
| Negative     | Negative               | Negative big    |
| Positive     | Positive               | Positive medium |
| Positive     | Zero                   | Zero            |
| Positive     | Negative               | Negative medium |
| Zero         | Positive               | Positive big    |
| Zero         | Zero                   | Positive medium |
| Zero         | Negative               | Zero            |

Because the fuzzy propositions are interpreted as fuzzy relations, the key question remaining is how to interpret the IF-THEN operation. In classical propositional calculus, the expression IF  $p$  THEN  $q$  is written as  $p \rightarrow q$  with the implication  $\rightarrow$  regarded as a connective defined by the following table, where  $p$  and  $q$  are propositional variables whose values are either truth (T) or false (F).  $p \rightarrow q$  is equivalent to

$$\bar{p} \vee q$$

and

$$(p \wedge q) \vee \bar{p}$$

Table 3.1.2. Truth table for  $p \rightarrow q$

| $p$ | $q$ | $p \rightarrow q$ |
|-----|-----|-------------------|
| T   | T   | T                 |
| T   | F   | F                 |
| F   | T   | T                 |
| F   | F   | T                 |

Because fuzzy IF-THEN rules can be viewed as replacing the  $p$  and  $q$  with fuzzy propositions, we can interpret the fuzzy IF-THEN rules by replacing the  $\neg, \vee$  and  $\wedge$  operators with fuzzy complement, fuzzy union, and fuzzy intersection respectively. Since there are a wide variety of fuzzy complement, fuzzy union, and fuzzy intersection operators, a number of different interpretations of fuzzy IF-THEN rules were proposed in the literature. We list some of them below

**Zadeh Implication:** Here the fuzzy IF-THEN rule *IF*  $\langle FP_1 \rangle$  *THEN*  $\langle FP_2 \rangle$  is interpreted as a fuzzy relation  $Q_z$  in  $U \times V$  with the membership function

$$\mu_{Q_z}(x, y) = \max[\min(\mu_{FP_1}(x), \mu_{FP_2}(y)), 1 - \mu_{FP_1}(x)]$$

**Godel Implication:** The Gödel implication is a well-known implication formula in classical logic. By generating it to fuzzy propositions, we obtain the following: the fuzzy IF-THEN rule  $IF \langle FP_1 \rangle THEN \langle FP_2 \rangle$  is interpreted as a fuzzy relation  $Q_G$  in  $U \times V$  with the membership function

$$\mu_{Q_G}(x, y) = \begin{cases} 1 & \text{if } \mu_{FP_1}(x) \leq \mu_{FP_2}(y) \\ \mu_{FP_2}(y) & \text{otherwise} \end{cases}$$

**Mamdani Implication:** The fuzzy IF-THEN rule is interpreted as a fuzzy relation  $Q_{MM}$  or  $Q_{MP}$  in  $U \times V$  with the membership function

$$\mu_{Q_{MM}}(x, y) = \min[\mu_{FP_1}(x), \mu_{FP_2}(y)]$$

$$\mu_{Q_{MP}}(x, y) = \mu_{FP_1}(x) \mu_{FP_2}(y)$$

Mamdani implications are the most widely used implications in fuzzy systems and fuzzy control. They are supported by the argument that fuzzy IF-THEN rule are local

### 3.3.4 Inference Engine

In a fuzzy inference engine, fuzzy logic principles are used to combine the fuzzy IF-THEN rules in the fuzzy rule base into a mapping from a fuzzy set  $A'$  in  $U$  to fuzzy set  $B'$  in  $V$ . There are two ways to infer with a set of rules: composition based inference and individual –rule based inference.

**1. Composition Based Inference:** In composition based inference, all rules in the fuzzy rule base are combined into a single fuzzy relation in  $U \times V$ , which is then viewed as a single fuzzy IF-THEN rule. So the key question is how to perform this combination. We

should first understand what as set of rules mean intuitively, and then we can use appropriate logic operators to combine them.

There are two opposite arguments for what a set of rules should mean. The first one views the rules as independent conditional statements. If we accept this point of view, then the reasonable operator for combining the rules is union. The second one views the rules as strongly coupled conditional statements such that the conditions of all the rules must be satisfied in order for the whole set of rules to have an impact. If we adapt this view, then we should use the operator intersection to combine the rules

Let  $Ru^{(l)}$  be a fuzzy relation in  $U \times V$ , which represents the fuzzy IF-THEN rule; that is,  $Ru^{(l)} = A_1^l \times \dots \times A_n^l \rightarrow B^l$ .

$$\mu A_1^l \times \dots \times A_n^l(x_1, \dots, x_n) = \mu A_1^l(x_1) \otimes \dots \otimes \mu A_n^l(x_n)$$

Where  $\otimes$  represents any t-norm operator. The implication  $\rightarrow$  is defined according to various implications defined in the previous section. If we accept the first view of a set of rules, then the M rules are interpreted as a single fuzzy relation  $Q_M$  in  $U \times V$  defined by

$$Q_M = \bigcup_{l=1}^M Ru^{(l)}$$

This combination is called Mamdani implication. For the second view of set of rules, the M fuzzy rules are interpreted as a fuzzy relation  $Q_G$  in  $U \times V$ , which is defined as

$$Q_G = \bigcap_{l=1}^M Ru^{(l)}$$

Let  $A'$  be any arbitrary fuzzy set in  $U$  and be the input to the fuzzy inference engine.

Then by viewing  $Q_M$  or  $Q_G$  as a single fuzzy IF-THEN rule and using generalized modus ponens, we obtain the output of the fuzzy inference engine as

$$\mu_{B'}(y) = \sup_{x \in U} t[\mu_{A'}(x), \mu_{Q_M}(x, y)]$$

If we use the Mamdani combination

$$\mu_{B'}(y) = \sup_{x \in U} t[\mu_{A'}(x), \mu_{Q_G}(x, y)]$$

If we use the Gödel combination.

### Algorithm

- For the M fuzzy IF-THEN rules, determine the membership functions  $\mu_{A_1^l} \times \dots \times \mu_{A_n^l}(x_1, \dots, x_n) = \mu_{A_1^l}(x_1) \otimes \dots \otimes \mu_{A_n^l}(x_n)$ .
- View  $A_1^l \times \dots \times A_n^l$  as the  $FP_1$  and  $B^l$  as the  $FP_2$  in the implication and determine  $Ru^{(l)} = A_1^l \times \dots \times A_n^l \rightarrow B^l$  for  $l = 1, 2, \dots, M$  according any one of these implications.
- Determine  $\mu_{Q_M}(x, y)$  or  $\mu_{Q_G}(x, y)$
- For any input A', the fuzzy inference engine gives output B'

**2. Individual-Rule Based Inference:** In individual-rule based learning, each rule in the fuzzy rule base determines an output fuzzy set and the output of the whole fuzzy inference engine is the combination of the M individual fuzzy sets. The combination can be taken either by union or intersection.

The computational procedure of the individual based inference is summarized as follows.

- For the M fuzzy IF-THEN rules, determine the membership functions  $\mu_{A_1^l} \times \dots \times \mu_{A_n^l}(x_1, \dots, x_n) = \mu_{A_1^l}(x_1) \otimes \dots \otimes \mu_{A_n^l}(x_n)$ .

- View  $A_1^l \times \dots \times A_n^l$  as the  $FP_1$  and  $B^l$  as the  $FP_2$  in the implication and determine  $R\mu^{(l)} = A_1^l \times \dots \times A_n^l \rightarrow B^l$  for  $l = 1, 2, \dots, M$  according any one of these implications.
- For given input fuzzy set  $A'$  in  $U$ , compute the output fuzzy set  $B_l^l$  in  $V$  for each individual rule  $R\mu^{(l)}$  according to the generalized modus ponens ,i.e.

$$\mu B^l(y) = \sup_{x \in U} t[\mu_{A'}(x), \mu R\mu^l(x, y)] \text{ For } l = 1, 2, \dots, M$$

- The output of the fuzzy inference engine is the combination of the  $M$  fuzzy sets  $\{B_1^l, \dots, B_M^l\}$  either by union, i.e.

$$\mu B^l(y) = \mu B_1^l(y) + \dots + \mu B_M^l(y)$$

or by intersection, that is

$$\mu B^l(y) = \mu B_1^l(y) \otimes \dots \otimes \mu B_M^l(y)$$

### 3.3.5 Defuzzification

To convert the output in membership form the Inference engine to a crisp control we use a defuzzifier.

There are several Defuzzification methods

1. Centre of Gravity
2. Centre of Gravity method for singleton
3. Bisector of Area
4. Mean of Maxima
5. Left most maximum, and right most Maximum.
6. Centre average

**1. Centre of Gravity:** The centre of gravity defuzzifier specifies the  $y^*$  as the centre of the area covered by the membership function of  $B'$ .

$$y^* = \frac{\int_v y \mu_{B'}(y) dy}{\int_v \mu_{B'}(y) dy} = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)}$$

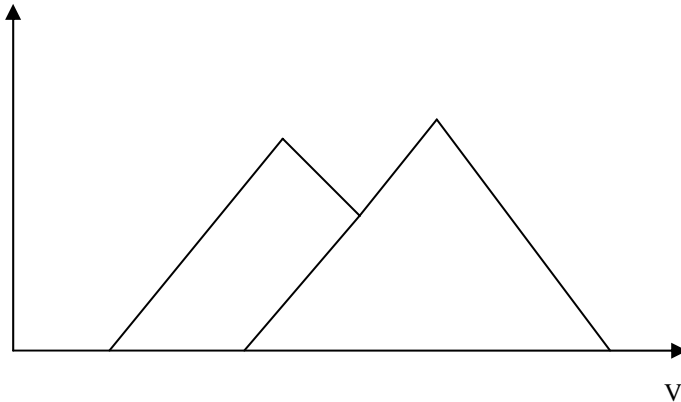


Fig 3.1.5

**2. Centre Average Defuzzifier:** Because the fuzzy set  $B'$  is the union or intersection of  $M$  fuzzy sets, a good approximation is the weighted average of the centers of the  $M$  fuzzy sets

$$y^* = \frac{y^1 w_1 + y^2 w_2}{w_1 + w_2}$$

**3. Bisector of Area:** This method picks the abscissa of the vertical line that divides the area under the curve in two equal halves. In the continuous case,

$$u = \left\{ x / \int_{\min}^x \mu(x) dx = \int_x^{\max} \mu(x) dx \right\}$$

Here  $x$  is the running point in the universe,  $\mu(x)$  is its membership function,  $\min$  is the left most value of the universe, and  $\max$  is the right most value. Its computational complexity is relatively high and it can be ambiguous.

### **3.3.6 Post Processing**

Output scaling is also relevant. In case the output is defined in standard universe this must be scaled to engineering units, for instance volts, meters, or tons per hour. An example is the scaling from standard universe  $[-1, 1]$  to the physical units  $[-10, 10]$  volts. The post processing block often contains an output gain that can be tuned, and sometimes also an integrator.

## **Chapter 4**

### **Fuzzyfying Markov Decision Processes**

This chapter is divided into three sections. In the first section we give a brief introduction to the problem in hand and an example to illustrate the problem of optimization. In the second section we proposed a Fuzzyfying Markov decision processes technique. It is illustrated using many examples and applications. In the third section we give simulation results.

Markov decision processes have become an indispensable tool in applications as diverse as equipment maintenance, manufacturing systems, inventory control, queuing networks and investment analysis. Typically we have a controlled Markov chain on a suitable state space in which transitional probabilities depend on the policy (or decision maker) which comes from a set of possible actions. The main problem of interest would be to find an optimal policy that minimizes the associated cost.

Typically we have finite horizon Markov decision processes and infinite horizon Markov decision process. Under finite planning horizon, the value iteration is perfect, but rarely the planning horizon is well defined. Most often the process is assumed to operate over an unknown period of time with no predetermined stage of termination. In such cases the abstraction of an infinite planning horizon seems more relevant.

The value iteration method is not exact, and the convergence is rather slow. On the other hand, the mathematical formulation is very simple, and the method makes it possible to handle very large models with thousands of states. Further it is possible to let the reward and the physical output depend on the stage number in some predefined way. This method has been used in many applications as an approximation to infinite stage optimum.

Another method is linear programming method. The linear programming method has almost the opposite characteristics of the value iteration method. Because of the more complicated mathematical formulation involving solutions of large systems of simultaneous linear equations, the method can handle rather small models with, say, a few hundred states. Complexity in linear programming increases much faster with the increase in number of states which is often called curse of dimensionality. On the other hand, the method is exact and very efficient in the sense of fast convergence. The rewards are not allowed to depend on the stage except for a fixed rate of annual increase or decrease.

An advantage of the linear programming method is that the equations are general. Under any policy  $g$  we are able to calculate directly the economic consequences by following the policy by solution of the equations. This makes it possible to compare the economic consequences of various non-optimal policies to those of the optimal. Linear programming method has been successfully used, but the models were very small.

In order to remove the problem of large number of states in linear programming method, we introduce a concept of fuzziness. By application of the fuzzy principles to Markov decision processes we are able to reduce the number of states thereby eliminate

the “curse of dimensionality”. Although the combination of fuzzy mathematics and Markov decision process introduces approximation in computing the optimal decision policy, the gain in reduction of the size of linear programming outweighs the approximation error, which will be demonstrated in this chapter.

## **4.1 Motivation**

In order to illustrate how the curse of dimensionality arises, we shall examine a simple dairy model. For any dairy cow it is relevant to consider at regular intervals whether it should be kept for an additional period or it should be replaced by a heifer. If the line of figure 1 represents time, the markers indicate where we consider replacing. The time interval between two markers is called a stage and in this example we assume the stage length to be one year, which for convenience is assumed always to be equal to a lactation period. At the beginning of each stage, we observe the state of the animal in production. The state space must be defined in such a way that all relevant information is given by the state. In this very simple example we assume, that the only relevant information is whether the cow is low, average or high yielding. Thus we have one state variable and three states. Thus if the cow remains high yielding it will never be replaced according to the optimal policies. This is certainly not realistic, and furthermore, the milk yield also depends on the lactation number. In order to account for age we shall introduce an additional state representing the lactation number of the cow. For convenience, we shall assume that the new variable may take the values 1, 2, 3 or 4 indicating that the maximum age of cow in this model is assumed to be 4 lactations. Now suppose that in addition to lactation and milk yield we also want to take the genetic merit into account. We shall assume that the genetic merit of the cow is either “good”, “average” or “bad”.

The total size of the state space then becomes  $3 \times 4 \times 4 = 36$ . The transition matrices of this 36- state model are now very large, compared to the beginning.

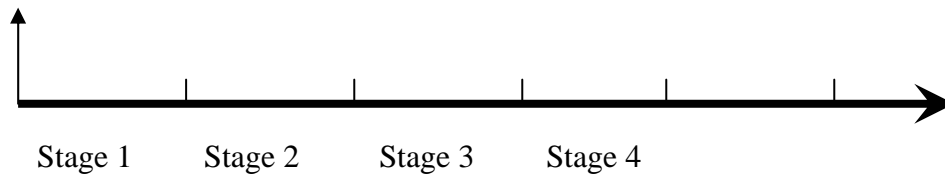


Fig 4.1.1: Number of stages

The stepwise extension of the model clearly illustrates that each time a new state variable at  $n$  levels is added to the model, the size of the state space is increased with a factor of  $n$ . When in real model, several traits are represented by state variable at a realistic number of levels; the size of the state space very soon reaches prohibitive dimensions (millions of states). As an example consider the dairy cow replacement model. The traits considered, when a decision was made were

- The age of the cow (240 levels).
- Milk yield in present lactation (15 levels).
- Milk yield in previous lactation (15 levels).
- Time interval between two successive calving (8 levels).
- Clinical mastitis- an infectious disease in the udder (2 levels).
- Accumulated number of mastitis cases in present lactation (4 levels).
- Accumulated number of mastitis cases in previous lactation (4 levels).

In principle the size of the state space is formed as the product of the number of levels of all traits i.e.  $240 \times 15 \times 15 \times 8 \times 2 \times 4 \times 4 = 11,750,400$  states. In practice it is smaller because some combinations are impossible and because traits related to previous lactation

are not considered during first lactation. Exclusion of such non-feasible states resulted in the model with 6,821,724 states.

## **4.2 Proposed Fuzzyfying Markov Decision Processes**

The main problem associated with linear programming optimization technique is the number of levels. As the number of levels increases, the size of the matrix increases to the same order. This will increase the computational complexity of the Markov Decision Processes. The main idea behind our FMDP is the use of fuzzy membership function to reduce the number of levels.

### **4.2.1 Fuzzy Membership Function**

A membership function is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is referred to as the universe of discourse. A fuzzy set is a generalization of a classical set by allowing the membership function to take any values in the interval  $[0, 1]$ . One of the most commonly used examples of a fuzzy set is a set of “tall” people. In this case the universe of discourse is all potential heights, say from 3 feet to 9 feet, and the word tall would correspond to a curve that defines the degree to which any person is tall. If the set of tall people is given the well defined boundary of a classical set, we might say all people taller than 6 feet are officially considered tall. But such a distinction is clearly absurd it may make sense to consider the set of all real numbers greater than 6 because numbers belong to an abstract plane, but when we want to talk about real people, it is unreasonable to call one person short and another one tall when they differ in height by the width of a hair. If we use an appropriate membership function then we can distinguish between people in an appropriate way

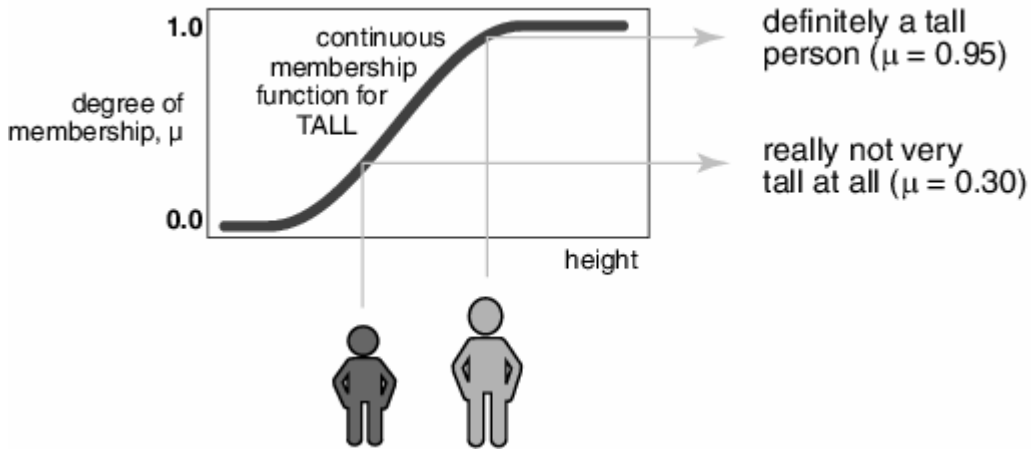


Fig 4.1.2:Fuzzy membership function

The output axis is the known as the membership value between 0 and 1. The curve is known as a membership function and is often given the designation of  $\mu$ . This curve defines the transition from not tall to tall. Both people are tall to some degree, but one is significantly less than the other.

Now the main question arises for how to use the fuzzy membership function to reduce the number of levels. Let us consider an example which has  $2n$  states.

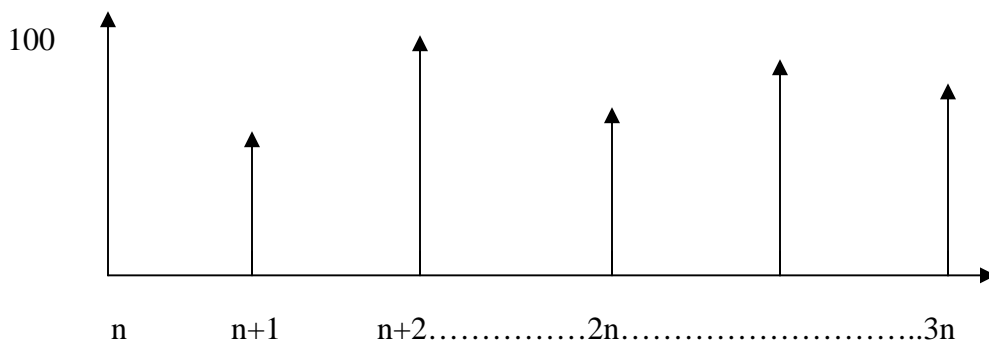


Fig 4.1.3: Number of states

Each state has a particular transitional probability. By the use of membership function we can mathematically reduce number of the states, but all the states remain through the use

of membership value. That is we reduce the number of states but those states reduced still exist. This is shown in one example as in Fig 4, where the states are all eliminated except the state  $kn$  for integer  $k$ .

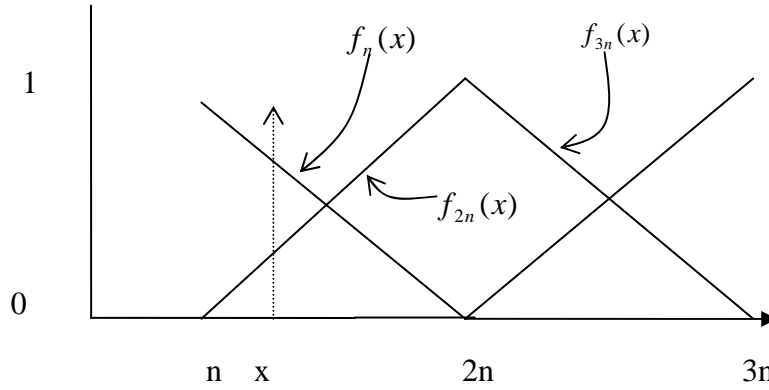


Fig 4.1.4: Reduced states

The membership function  $f_{2n}(x)$  indicates the fuzziness of  $x$  in relation to state  $2n$ . If  $f_{2n}(x) = 1$ , then  $x$  is precisely the state  $2n$ . If  $0 < f_{2n}(x) < 1$ , then  $x$  is also close to state  $n$ , or state  $3n$  dependent on  $f_n(x) = 0$  or  $f_{3n}(x) = 0$ . For instance in our case,  $f_{3n}(x) = 0$  and  $0 < f_{2n}(x) < 1$ . Because  $0 < f_n(x) < 1$ . The state  $x$  must lie in between state  $n$ , and state  $2n$ , which can be interpolated as follows:

$$x = f_n(x)n + f_{2n}(x)2n$$

In our case, as shown in the Fig 4,  $x = \frac{5n}{4}$ .

So  $f_n(x) = \frac{3}{4}$ ,  $f_{2n}(x) = \frac{1}{4}$  and thus using the above equation gives

$$x = \frac{3}{4}n + \frac{1}{4}2n = \frac{5n}{4}$$

This recovers exactly the true state value. In conclusion, the key condition to recover the states that are eliminated is  $f_{kn}(x) + f_{(k+1)n}(x) = 1$  if  $x$  is between state  $kn$  and state  $(k+1)n$ .

Although the use of straight line as fuzzy membership function in the above example works very well in recovering those states eliminated, straight line may not be good choice for making optimal Markov decision. The main reason is that the dimension of the stochastic matrix has also been reduced significantly. This degrades greatly the fidelity of the model so the fuzzy membership function needs to play a role in recovering the optimality of the Markov policy inspite of the reduction in the number of levels.

We note first that although a state, say  $x$  in Fig.4 is eliminated, it still exist through the representation of fuzzy membership function. Hence, its transition probability to other states such as  $kn$  for  $k \neq 1, k \neq 2$  can be estimated through

$$p(kn/x) = f_n(x)p(kn/n) + f_{2n}(x)p(kn/2n) \quad (4.1.1)$$

Since  $f_n(x) + f_{2n}(x) = 1$ ,  $p(kn/x)$  is indeed a transition probability. In addition, if  $x = n$ , then  $f_n(x) = 1, f_{2n}(x) = 0$  and  $p(kn/x) = p(kn/n)$ . Similarly, if  $x = 2n$ , then  $f_n(x) = 0$ , and  $f_{2n}(x) = 1$ , we have  $p(kn/x) = p(kn/2n)$ . Hence (4.1.1) is consistent to the case when  $x$  is a remaining state. However, straight line fuzzy membership function may not approximate the true  $p(kn/x)$  well.

We would like to caution the use of (4.1.1): an assumption has to be made in order for (4.1.1) to make sense. In this case, we assume certain monotonicity property holds true for transition probability. That is, the true value of  $p(kn/x)$  lies between  $p(kn/n)$  and  $p(kn/2n)$  as long as  $x$  is between that state  $n$  and  $2n$ .

In summary we propose to reduce the state levels by introducing the fuzzy membership functions.

Now the main question is whether we can map transitional probabilities on a fuzzy membership function. The minimum requirement of probabilities is the complementary property; that is they should add together to one, or the integral of their density curves should be one. Not all fuzzy membership functions sum up to one, so we have to be very careful in selecting membership function exponential, triangular, sine/cosine etc.

Let us continue with the example of dairy. Initially when we were using the model there were  $240 \times 15 \times 15 \times 8 \times 2 \times 4 \times 4 = 11,750,400$  states. When we use a fuzzy membership function to reduce the number of states the total number of states can be reduce to,

- For the age of the cow we were using 240 levels, this can be reduced to 24 states if we map 10 states using a membership function.
- Milk yield in present lactation can be reduced to 3 states.
- Milk yield in previous lactation can also be reduced to three states.
- Time interval between two successive calving can be reduced to a single state.
- Accumulated number of mastitis cases in previous lactation can be reduced to a single state.
- Accumulated number of mastitis cases in current lactation can also be reduced to a single state.

Therefore after using our technique for the model we have reduced the number of states to  $24 \times 3 \times 3 \times 2 \times 1 \times 1 = 432$  from 11,750,400 states.

### 4.3 Example

Let us consider the deterioration model of prestressed concrete deck. The model is taken from LA-DOTD. They are 16 states and 8 possible action sets, that is  $n=16$  and  $a=8$ ;

Table 4.1.1 Preservation model details

| S.No | Action                  | Cost |
|------|-------------------------|------|
| 1    | Do-nothing              | 0    |
| 2    | Minor-Minor Maintenance | 5    |
| 3    | Minor Maintenance       | 10   |
| 4    | Minor-Major Maintenance | 15   |
| 5    | Major-Minor Maintenance | 30   |
| 6    | Major Maintenance       | 60   |
| 7    | Major-Major Maintenance | 90   |
| 8    | Replacement             | 301  |

The transition probabilities for the eight possible actions is given by

$$p^0 = \begin{bmatrix} 50 & 30 & 16.21 & 1.8 & 1.1 & .57 & .15 & .06 & .04 & .03 & .02 & .02 & 0 & 0 & 0 & 0 \\ 0 & 51.29 & 28.2 & 14 & 2.5 & 1 & .71 & .6 & .5 & .4 & .4 & .3 & .1 & 0 & 0 & 0 \\ 0 & 0 & 49.11 & 25 & 18 & 2.6 & 1.49 & 1.2 & 1.17 & 1 & .73 & .64 & .56 & 0 & 0 & 0 \\ 0 & 0 & 0 & 53.29 & 28.4 & 15.8 & .85 & .5 & .4 & .36 & .15 & .11 & .7 & .4 & .3 & 0 \\ 0 & 0 & 0 & 0 & 49.19 & 28 & 17.1 & 1.3 & .9 & .75 & .65 & .6 & .57 & .53 & .41 & 0 \\ 0 & 0 & 0 & 0 & 0 & 51.21 & 31.01 & 11.05 & 2.53 & 1.2 & 1 & .8 & .7 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 52.07 & 26.36 & 15.58 & 2.36 & 1.2 & .88 & .7 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 53.19 & 27.58 & 12.52 & 2.45 & 1.65 & 1.1 & .9 & .7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 54.21 & 25.25 & 12.8 & 4.38 & 1.3 & 1.13 & .93 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 51.02 & 31.56 & 16.47 & .45 & .35 & .15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 55.26 & 28.23 & 14.05 & 1.32 & .55 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 52.18 & 31.26 & 15.23 & 1.33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 56.21 & 30.25 & 13.54 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60.29 & 39.71 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 99.05 & .95 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

$$p^1 = \begin{bmatrix} 50 & 30 & 16.21 & 1.8 & 1.1 & .57 & .15 & .06 & .04 & .03 & .02 & .02 & 0 & 0 & 0 & 0 \\ 0 & 51.29 & 28.2 & 14 & 2.5 & 1 & .71 & .6 & .5 & .4 & .4 & .3 & .1 & 0 & 0 & 0 \\ 0 & 0 & 49.11 & 25 & 18 & 2.6 & 1.49 & 1.2 & 1.17 & 1 & .73 & .64 & .56 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 52.07 & 26.36 & 15.58 & 2.36 & 1.2 & .88 & .7 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 53.19 & 27.58 & 12.52 & 2.45 & 1.65 & 1.1 & .9 & .7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 54.21 & 25.25 & 12.8 & 4.38 & 1.3 & 1.13 & .93 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 51.02 & 31.56 & 16.47 & .45 & .35 & .15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 55.26 & 28.23 & 14.05 & 1.32 & .55 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 52.18 & 31.26 & 15.23 & 1.33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 56.21 & 30.25 & 13.54 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60.29 & 39.71 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 99.05 & .95 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$





$$p6 = \begin{bmatrix} 50 & 30 & 16.21 & 1.8 & 1.1 & .57 & .15 & .06 & .04 & .03 & .02 & .02 & 0 & 0 & 0 & 0 \\ 0 & 51.29 & 28.2 & 14 & 2.5 & 1 & .71 & .6 & .5 & .4 & .4 & .3 & .1 & 0 & 0 & 0 \\ 0 & 0 & 49.11 & 25 & 18 & 2.6 & 1.49 & 1.2 & 1.17 & 1 & .73 & .64 & .56 & 0 & 0 & 0 \\ 0 & 0 & 0 & 53.29 & 28.4 & 15.8 & .85 & .5 & .4 & .36 & .15 & .11 & .7 & .4 & .3 & 0 \\ 0 & 0 & 0 & 0 & 49.19 & 28 & 17.1 & 1.3 & .9 & .75 & .65 & .6 & .57 & .53 & .41 & 0 \\ 0 & 0 & 0 & 0 & 0 & 51.21 & 31.01 & 11.05 & 2.53 & 1.2 & 1 & .8 & .7 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 52.07 & 26.36 & 15.58 & 2.36 & 1.2 & .88 & .7 & .5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 53.19 & 27.58 & 12.52 & 2.45 & 1.65 & 1.1 & .9 & .7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 52.18 & 31.26 & 15.23 & 1.33 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 56.21 & 30.25 & 13.54 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 60.29 & 39.71 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 99.05 & .95 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

$$p7 = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

Short Term

Optimization Results:

Table 4.1.2: Action selection policy

| State | Action | Cost   |
|-------|--------|--------|
| 1     | 0      | 211.90 |
| 2     | 0      | 217.34 |
| 3     | 0      | 225.32 |
| 4     | 0      | 223.72 |
| 5     | 0      | 222.98 |
| 6     | 1      | 225.82 |
| 7     | 2      | 229.25 |
| 8     | 3      | 233.52 |
| 9     | 4      | 251.30 |
| 10    | 0      | 267.80 |
| 11    | 0      | 274.16 |
| 12    | 0      | 286.78 |
| 13    | 0      | 295.49 |
| 14    | 0      | 304.43 |
| 15    | 0      | 320.07 |

Long Term

Optimal limiting probability

$$w_{ia} = \begin{bmatrix} .3801 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ .2342 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .1857 \\ 0 & .04 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.01 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.11 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.01 \end{bmatrix}$$

The probability that the element will be in state  $i$  and action  $a$  is chosen.

The states are reduced by using a straight line membership function. The number of states is reduced to 6 states and the numbers of actions are reduced to 4 using singleton fuzzy membership function. The reduced set of actions is given by

Table 4.1.3: Reduced set of actions

| S.No | Action            | Cost |
|------|-------------------|------|
| 1    | Do-nothing        | 0    |
| 2    | Minor Maintenance | 30   |
| 3    | Major Maintenance | 180  |
| 4    | Replacement       | 301  |

The transitional probabilities are given by

$$p0 = \begin{bmatrix} .9282 & .0667 & .0051 & 0 & 0 & 0 \\ 0 & .9978 & .0010 & .006 & .006 & 0 \\ 0 & 0 & .9284 & .037 & .0346 & 0 \\ 0 & 0 & 0 & .9853 & .0147 & 0 \\ 0 & 0 & 0 & 0 & .9571 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$p1 = \begin{bmatrix} .9282 & .0667 & .0051 & 0 & 0 & 0 \\ 0 & .9978 & .0010 & .006 & .006 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .9853 & .0147 & 0 \\ 0 & 0 & 0 & 0 & .9571 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$p2 = \begin{bmatrix} .9282 & .0667 & .0051 & 0 & 0 & 0 \\ 0 & .9978 & .0010 & .006 & .006 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$p3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

## Short Term

Table 4.1.4: Fuzzified action selection policy

| State | Action | Cost   |
|-------|--------|--------|
| 1     | 0      | 96.1   |
| 2     | 0      | 121.53 |
| 3     | 1      | 149.1  |
| 4     | 0      | 127.81 |
| 5     | 2      | 305.26 |

## Long Term

The limiting probability that the element will be in state  $i$  and action  $a$  is chosen is given by

$$w_{ia} = \begin{bmatrix} .70 & 0 & 0 & 0 \\ 0 & 0 & 0 & .05 \\ .05 & 0 & 0 & 0 \\ .20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Comparing the results with the original 16 states gives us a very good approximation of our results. When we reduced the number of states total cost is reduced, and cost obtained is identical for six of the 15 states. That is we have been successful in reducing the number of states to 9 from 16. If we do further study on fuzzy membership function we might be able to reduce the number of states to 5. The long term limiting probability that the element will be in state  $i$  and action  $a$  is chosen is similar to the original sixteen states showing that the element will be in state 1 for most of the time. Thus we have obtained a

result which is very good approximation of the original problem in hand, considering the amount of data reduced. Thus this approximation technique can be used in decision making when there are many states.

## **Chapter 5**

### **Conclusion**

#### **5.1 Concluding Remarks**

Looking back at the initial problem in hand, where the number of states increases multifoldly with the increase in levels, that is the “curse of dimensionality”. The optimal decision making using linear programming method requires solutions of large systems of simultaneous equations. Therefore it can handle rather small models with, say, a few hundred states.

In this thesis we have tried to eliminate the curse of dimensionality by using fuzzy membership function. By the use of fuzzy membership function we mathematically reduce the number of states, but all the states remain intact through interpolation of the membership values. That is we reduce the number of states but those states eliminated can still be recovered. The assumption is the monotonicity property for the transition probability. One of the questions not answered here is what the best possible fuzzy membership function is. Although the use of straight line as fuzzy membership function works well in recovering those states eliminated, straight line may not be a good choice for making optimal Markov decisions. The main reason is that the dimension of the

stochastic matrix has also been reduced significantly. This degrades greatly the fidelity of the model.

Although the combination of fuzzy mathematics and Markov decision processes introduces approximation in computing the optimal decision policy, the gain in reduction of the size of linear programming outweighs the approximation error. Therefore we conclude that this method can be successfully implemented in practical problems where there is a large number of states.

## **5.2 Future Research**

Although some research has been carried out to cope with the curse of dimensionality, there lack good results in this problem area. This thesis is the first effort to apply fuzzy mathematics to lessen the curse of dimensionality problem. Despite some success, more research is needed, which include

- Optimal choice of fuzzy membership function.
- How to eliminate the monotonicity assumption on the transitional probability, because not all transitional probability matrices satisfy such an assumption .

The problem area is still in its early stage, and many problems need to be solved before our proposed technique can be used successfully for manufacturing systems, equipment maintenance, inventory control, queuing networks and investment analysis.

## References

1. S. Balaji, *Markov Decision Processes*.
2. P. R. Kumar, Pravin Varaiya, *Stochastic Systems Estimation, Identification & Adaptive Control*, 1986.
3. G. R. Walsh, *An Introduction to Linear Programming*, 1985.
4. Sheldon M. Ross, *Introduction to Stochastic Dynamic Programming*, 1983.
5. Xioaduan Sun, Robert Wang, Zhongjie Zhang “Analysis of Past NBI Ratings to Determine Future Bridge Preservation Needs,” *Louisiana Transportation Research*, 2004.
6. American Association of State Highway and Transportation Officials (AASHTO) Pontis Technical Manual.
7. Anders R. Kristensen, *Dynamic Programming and Markov Decision Processes*.
8. D.P. Bertsekas, *Dynamic Programming and Optimal Control*, 2000.
9. Li- Xin Wang, *A Course in Fuzzy Systems & Control*, 1997.
10. K. Golabi, P. D. Thomson and C.H. Jun, “Network Optimization System for Bridge Improvements of Maintenance,” Interim report submitted to California Department of Transportation, 1990.
11. K. M. Passino, *Stephen Yurkovich, Fuzzy Control*, 1997.
12. Greg Goebel, *An Introduction to Fuzzy Control Systems*, 2003.
13. Bart Kosko, *Fuzzy Thinking*.
14. Daniel McNeil & Paul Freiberger, *Fuzzy Logic*.
15. M. Jhamshidi, N. Vadiee, T. J. Ross, *Fuzzy Logic and Control*, 1993.
16. D. Dubois, L. Foulloy, G. Mauris, H. Prade, “Probability-Possibility Transformation, Triangular Fuzzy Sets and Probabilistic Inequalities”, 10(2004), pp.273-297.
17. A. Neumaier, “Clouds, Fuzzy Sets and Probability Intervals”, *Reliable Computing* 10: 249-272, 2004.

## **Vita**

Syed Irshad Ahmed was born in Hyderabad, Andhra Pradesh, India. He received his schooling in St Georges Grammar School and Shadan Junior College in Hyderabad, Andhra Pradesh, India. He received his Bachelor of Engineering from Osmania University. His research interest is in fuzzy control, stochastic processes. He joined Louisiana State University in 2003 and is expected to graduate the degree of Master of Science in Electrical Engineering in May 2005.