Stochastic Dynamic Programming

Jan van der Wal
Department of Mathematics and Computing Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

August 31, 2010

Contents

Chapter 1

Preface

Many stochastic decision problems can be formulated as a Markov decision process. A Markov decision process can be seen as an extension of the Markov chain. The extension is that in each state the system has to be controlled by choosing one out of a number of possible decisions or actions.

Markov decision processes are theoretically well understood (although some questions remain to be answered) and practically easy to use and to solve and can hence be a very important tool.

The basic characteristic of the Markov decision process is that it is controlled continuously, although often only at discrete points in time, such as every day. In some decision processes the most important decision is made at the beginning, e.g., to buy or not to buy a machine, to have 4 warehouses or just one, to build a dam or not, etc. These decision problems are not addressed in this course, although such investment decisions have to be based on a good understanding about what is going to happen after the initial decision is taken. For that one might want to use the Markov decision process formulation again.

The standard approach for finding the best decisions in a sequential decision problem is known as dynamic programming, or *stochastic dynamic programming*.

In this course we first consider the case in which the number of decision epochs is finite, the so-called *finite horizon problem*. After that we consider, so-called, infinite horizon problems, for the criteria of discounted rewards and average reward per time unit . The text is concluded with chapters on semi-Markov decision processes and Markov decision processes with additional structure. Throughout this course we restrict ourselves (with some small exceptions) to problems with only finitely many states and finitely many actions.

Chapter 2

Deterministic dynamic programming

This chapter discusses finite horizon dynamic programming. Before presenting a somewhat more formal dynamic programming framework we consider some examples.

2.1 Examples

Example 2.1.1 Shortest path
Consider the following simple graph in which one has to find the shortest path from A to B. Clearly, the path first passes through one of the states {1,2,3} and after that through 4, 5 or

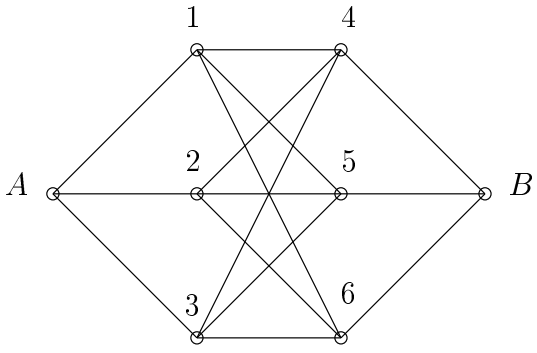

Figure 2.1: Shortest path

6. So the possible paths are $A14B, A15B, A16B, A24B, A25B, A26B, A34B, A35B, A36B$. If the lengths of the edges are given, the length of each path can be computed by merely adding up the lengths of the edges on that path. So, by comparing the lengths of these 9 paths one easily finds the shortest one. For the graph, in which after the set {4,5,6} the path goes to the set {7,8,9}, then to {10,11,12} and only then to B, the total number of paths is already 81 (3 to the power 4).

As we see the total number of possible path tends to explode. In a graph with T intermediate sets each consisting of K alternatives, the total number of paths is K^T. With $K = 10$ and $T = 20$ we have already 10^{20} paths. A PC that is able to evaluate and compare 10^7 paths per second, will need 300.000 years to find the optimal path if it would have to compare all paths.

Fortunately, this is not necessary. Instead one may profit form the obvious property that if the shortest path from A to B passes through i, the sub-paths from A to i and from i to B are shortest paths as well. This seemingly simple, but very powerful property, is known as *Bellman's optimality principle*.

Example 2.1.2 Knapsack
Suppose we are going on a hiking trip and we want to take at most 12 kilograms in our backpack. Of course there are a number of items that we have to take. Once we have packed all the absolutely necessary things, we still have 3 kilograms left. There are 12 items in front of us at the table that we really would like to pack, but together they weigh nearly 7 kilograms, so we have to make a choice. In the following table we list the 12 items, their weights and the value they have to us if we pack them.

Item	1	2	3	4	5	6	7	8	9	10	11	12
Weight	0.4	0.7	1.1	0.2	0.2	0.3	0.6	1.3	0.8	0.6	0.4	0.3
Value	5	9	12	3	3	5	7	15	9	8	7	5

The total number of possibilities is limited by 2 to the power 12, which is 4096. For various reasons, many of these possibilities make no sense at all. However, if the number of items is increased to 50, the number of alternatives explodes to 10^{15}.

Example 2.1.3 Production planning
In the next 5 weeks we have to deliver 500, 300, 300, 400 and 200 items of some product. In order to produce these items we have to start a production run. The set-up costs for a production run are 40 Euro (where we ignore the costs per item, as all 1700 items have to be produced anyway). There are also inventory costs, valued at 3 Euro per 100 items per week.

We have to find the optimal production schedule. From the cost structure it is clear that we never start a production run if we still have enough items to satisfy this weeks demand. Also, if a production run is started we produce the demand for the next k weeks, and not for $k + 0.5$ weeks.

2.2 The dynamic programming technique

As we have seen, there is a common structure in the examples we presented. In all examples one may define the following four characteristics.

- *States and stages.* There is a set of states for each of the stages of the problem. The characterization of the state is in most examples the same in each stage. For clarity reasons we begin by denoting the state-stage combination by (i, n).
- *Decisions.* In each of the states (i, n) there is a set of possible decisions which will be denoted by $K(i, n)$.
- *Transitions.* Given the combination state-stage (i, n) and the decision k there is a transition to particular state in the next stage, denoted by $F(i, n, k)$ (F for following).
- *Rewards.* If decision k is taken in (i, n) then there is an immediate reward $r(i, n, k)$. (We prefer to speak in terms of rewards and maximization although for many problems minimizing costs is the more natural formulation.)

The stages are numbered in reverse order. (The reason for this is the fact that for stochastic problems working backwards is the only way.) I.e., n denotes the number of stages *to go*. We will denote the total number of stages by T, numbering them $T, T-1, \ldots, 1$.

In several examples there is also some terminal payoff at the end of the planning period. For example, in a production planning problem, the items left to be scrapped or in a problem in which the status of a machine has to be controlled, the value of the machine. Let us denote this terminal payoff by $q(i, 0)$. In the cases in which there is no terminal payoff one defines $q(i, 0) = 0$.

Now the dynamic programming (DP) technique can be formulated.

Dynamic programming 1

- *Define* $v(i, 0) = q(i, 0)$ *for all* i *at stage 0.*
- *Compute for all* n, $n = 0, \cdots, T-1$ *and all* i *at the corresponding stage*

$$v(i, n+1) = \max_{k \in K(i, n+1)} \{r(i, n+1, k) + v(F(i, n+1, k))\} \ .$$

- *The maximizing actions in the various stages and states together constitute an optimal strategy for the problem.*

The formulation to be used in the sequel exploits the fact that the state is not really a pair (i, n). As n mainly indicates the period, and the dynamic programming consists of a sequence of decisions to be taken, an alternative formulation is the following.

Dynamic programming 2

- *Define* $v_0(i) = q(i)$ *for all* i *at stage 0.*

- *Compute for all n, $n = 0, \cdots, T-1$ and all i at the corresponding stage*

$$v_{n+1}(i) = \max_{k \in K(i)} \{r(i,k) + v_n(F(i,k))\} \ .$$

- *The maximizing actions in the various stages and states together constitute an optimal strategy for the problem.*

2.2.1 Shortest path; continued

For the shortest path problem the dynamic programming approach is straightforward. First define $v_0(B) := 0$ (once B is reached there are no more 'costs'). Then compute $v_1(i)$ for $i = 4, 5, 6$. (Note that from 4, 5 and 6 the shortest path is just the length of the edge to B.) Next compute $v_2(i)$ for $i = 1, 2, 3$. For instance, with $d(i,j)$ (d for distance) the length of the edge from i to j,

$$v_2(1) = \min\{d(1,4) + v_1(4), d(1,5) + v_1(5), d(1,6) + v_1(6)\} \ .$$

Etc.

2.2.2 Production planning; continued

Let us execute the algorithm for the production planning example. First we define the states. We already argued that we only produce whole weekly demands. So, a possible state description is the number of hundreds produced so far, so 0, 5, 8, 11, 15 or 17. Further it is clear that some states are not possible in some periods since we are not allowed to be late. Also the decisions differ per state-stage combination. If, with three periods to go, we have already produced 15 items, then we will not produce anything because we can (and thus it is best to) wait. On the other hand, if we have produced 15 at the beginning of the last period then we have to produce 2 more.
A complication arises from the inventory costs. It is more or less arbitrary in which period we account for these costs. Below we will compute the inventory costs at the beginning of the next period.
Note that for this problem, the number of periods to go already specifies which total productions amounts are possible, both for the beginning of the period and the end of the period.

With one period to go, the production sofar must be at least 15 and can be only 15 and 17, since, as has been argued before, production is for whole periods only. In this way one gets

$$v_1(15) = 40 \ \ (\textit{the setup costs only}) \ ,$$

and

$$v_1(17) = 6 \quad (\textit{the inventory costs}) \ .$$

With 2 periods to go there is only a decision to take if the total production sofar is 11:

$$v_2(11) = \min\{4 : 40 + v_1(15),\ 6 : 40 + v_1(17)\} = \min\{80, 46\} = 46 \ ,$$

and

$$v_2(15) = 12 + v_1(15) = 52 \ ,$$

$$v_2(17) = 18 + v_1(17) = 24 \ .$$

Continuing in this way yields

$$\begin{aligned}
v_3(8) &= \min\{3 : 40 + v_2(11),\ 7 : 40 + v_2(15),\ 9 : 40 + v_2(17)\} \\
&= \min\{86, 92, \mathbf{64}\} = 64 \ , \\
v_3(11) &= 9 + v_2(11) = 55 \ , \\
v_3(15) &= 21 + v_2(15) = 73 \ , \\
v_3(17) &= 27 + v_2(17) = 51 \ , \\
v_4(5) &= \min\{40 + v_3(8),\ 40 + v_3(11),\ 40 + v_3(15),\ 40 + v_3(17)\} \\
&= \min\{104, 95, 113, \mathbf{91}\} = 91 \ , \\
v_4(8) &= 9 + v_3(8) = 73 \ , \\
v_4(11) &= 18 + v_3(11) = 73 \ , \\
v_4(15) &= 30 + v_3(15) = 103 \ , \\
v_4(17) &= 36 + v_3(17) = 87 \ , \\
v_5(0) &= \min\{40 + v_4(5),\ 40 + v_4(8),\ \ldots\ ,\ 40 + v_4(17)\} \\
&= \min\{131, \mathbf{113}, \mathbf{113}, 143, 127\} = 113 \ .
\end{aligned}$$

This algorithm allows a simple description in the form of a matrix. The rows correspond to the amount already produced and the columns indicate the amount that has to be ready at the end of the period.

So, there are two optimal strategies. The first one prescribes to produce 8 items in the first period and the remaining 9 items in period 3. According to the other optimal strategy we make 11 items in period 1 and the other 6 in period 4.

This algorithm is known as '*Wagner and Whitin*'.

Wagner Whitin		Must be ready 5	8	11	15	17
	0	113				
	5		91			
Ready	8		73	64		
	11		73	55	46	
	15		103	73	52	40
	17		87	51	24	6

2.2.3 Knapsack; continued

For the knapsack problem, the DP structure is less obvious. What we can do is treat the selection problem as a sequential decision problem. First consider item 1, next item 2, and finally, in the last period, item 12. Then the state i can be defined as the number of ounces left, where i can have the values 0 up to 30.

This leads us to the following DP approach.

Define $v_0(i) := 0$ for all i, indicating that since all items have already been considered, the remaining weight has no value. Next compute $v_1(i)$ as the optimal way to use the remaining weight if only item 12 is left to consider. So,

$$v_1(i) = 0 \ , \ i = 0, 1, 2 \ .$$

$$v_1(i) = 5 \ , \ i = 3, \ldots, 30 \ .$$

Next, with 2 periods to go, thus with items 11 and 12 to consider, we get for instance,

$$v_2(5) = \max\{\textit{yes}, \ \textit{take } 11: \ 7 + v_1(1), \ \textit{no}, \ \textit{don't take } 11: \ 0 + v_1(5)\} = 7 \ .$$

Etc.

Note that in this deterministic problem there is no good reason for starting the computations with item 12 when there is only one period to go. We could have said that with one period to go, we consider item 1, just as well.

Chapter 3

Finite horizon stochastic dynamic programming

3.1 Introduction

There is little difference between the deterministic dynamic programming models and the stochastic problems. The only essential change is the fact that a decision in a state no longer leads to a specific state but defines a distribution for the next state. As a result of this randomness also the rewards sometimes are stochastic, as they may depend on the transition. One can easily cope with that by dealing with expected rewards.

A stochastic dynamic programming model is characterized by the following four elements.

- *States.* There is a, in our case, finite set of states $I := \{1, 2, \dots , N\}$. The characters i and j are used to denote the states.
- *Decisions.* In state i there is a set of possible decisions which will be denoted by $K(i)$.
- *Transitions.* If in state i decision k is taken, then the 'system' makes a transition to state j with probability p_{ij}^k.
- *Rewards.* If decision k is taken in i, then there is an expected immediate reward $r(i, k)$.

Remark 3.1.1 *Note that we ignore the fact that in some problems the stage might be an essential part of the state description. In those cases the extension is however straightforward (cf. section 2.2).*

Example 3.1.2 Machine replacement
Consider a machine that can be in four states: *Excellent, Good, Fair, Poor*. The machine is inspected weekly. From ample statistical information one has concluded that the probability that an Excellent machine is still Excellent one week later is equal to 0.8. With probability 0.2 the state has degraded to state Good. From state Good the machine goes to state Fair with probability 0.3 or it remains Good. From Fair it goes to Poor with probability 0.4 or stays Fair. A Poor machine wil be Poor forever.

The operating costs, maintenance and production losses, vary with the state. In the states Excellent, Good, Fair and Bad these costs are 100, 500, 1000 and 2000 per week, respectively. It is possible to trade the machine for a new one. The costs depend on the state and are 5000, 6000 and 8000, respectively, in the states Good, Fair and Poor.

Given that one will be using this type of machine for 12 more weeks before a much faster and more efficient one will be installed, what is the optimal replacement strategy?

3.2 Strategies

In the problems we treat in this course we encounter mainly two types of strategies

1. *Stationary strategies*. Recall that the state space is the same in each stage. A stationary strategy is fully characterized by the action it prescribes in state i, $i \in I$. So the action taken in state i is the same in each stage. Such a strategy will be denoted by the function f, with $f(i)$ the action prescribed in state i. We also use the term *policy* for the function f. The number of stationary strategies is given by $\Pi_i K_i$, thus usually very large.

2. *Markov stategies*. A Markov strategy is a sequence of policies f, one for each stage. Notation $(f_1, f_2, \ldots)$. If the number of stages is finite, T say, we write $(f_1, \ldots, f_T)$.

3.3 Finite horizon dynamic programming

In the finite horizon model one is controlling the system during a finite number of stages, T periods say. This makes it is necessary to model what happens when the systems 'ends' in a specific state. It has to be 'quantified' what it means if the process terminates in a specific state. For example, if one is dealing with an inventory problem and the state reflects the number of items in stock, then, at the and of the planning horizon, the value of the remaining parts has to be taken into account.
This is called this the *terminal payoff* and is denoted by $q(i)$ in state i.

To solve the finite horizon problem we use *dynamic programming*. Another term that is frequently used in literature is *successive approximation*.

Dynamic programming

- *Define*

$$v_0(i) = q(i) \ , \quad i \in I$$

- *Compute for* $n = 0, 1, \ldots, T-1$

$$v_{n+1}(i) = \max_{k \in K(i)} \{r(i,k) + \sum_{j \in I} p_{ij}^k v_n(j)\} \quad i \in I \tag{3.1}$$

- *Let* $f_{n+1}(i)$ *be the (a) maximizing action in the optimization problem above, then the policies* f_n *constitute a Markov strategy* $(f_T, \ldots f_1)$ *for the T-period problem. Note that the indices of the policies run in reversed order so that* f_n *is the policy to apply when there are still* n *periods to go.*

Theorem 3.3.1 *For the T-period problem the function* v_T *is the optimal reward and* $(f_T, \ldots f_1)$ *is the optimal (Markov) strategy.*

Proof: The proof of this result is straightforward if we consider Markov strategies only. Then we can easily show by induction that the Markov strategy $(f_T, \ldots f_1)$ is at least as good as any other Markov strategy. If however we consider more complicated strategies as well, e.g., strategies which randomize between actions and strategies which base the actions also on the complete history of the process, then the proof becomes notationally involved. The result, however, remains true.

3.3.1 The complexity

An important aspect of this dynamic programming algorithm is its complexity. As for each stage one has to consider each state, and in each state all possible actions and for each state-action combination all possible transitions, the complexity is given by

$$T \sum_i \sum_{k \in K(i)} F(i,k) \ ,$$

with $F(i,k)$ the number of transitions that are possible (with positive probability) when action k is taken in state i. If $|K(i)| = K$ for all i, and $F(i,k) = N$ for all i and k, then the complexity becomes TN^2K. For larger problems, often however, $F(i,k)$ is much smaller that N.

For example, for an intermediate size problem with 1000 states, in each state 20 possible decisions, per state-decision combination 50 transitions, with 100 periods, the complexity is only 10^8. So the computation time on a PC should only take a few seconds.

Chapter 4

Essentially one period problems

In this chapter we consider two stochastic decision problems that are dynamic and in principle have an infinite horizon, but have a special structure, as a result of which they are essentially one-period problems.

4.1 Periodic Review

In this first example we consider one of the most simple stochastic inventory control problems. In a warehouse we keep an inventory of a product in order to be able to satisfy customer demand immediately. Once a week, Friday afternoon say, we inspect the inventory and we have take a replenishment decision. Replenishment is free and immediate, i.e. costs no time. (E.g., the replenishment is ordered late Friday afternoon, the warehouse is closed in the weekend, and the replenishment is received at Monday morning before reopening.) The only costs we have to take into account are inventory costs and shortage costs. We assume that we have to pay inventory costs h for every unit that is found in the warehouse at the inspection Friday afternoon and that we pay shortage costs b for every unit of demand that we can not satisfy directly from stock. We assume that this demand is lost. This covers a lot of different possibilities of which lost demand is one. But it could also mean that we have to supply a more expensive product or that we have to fetch the lost demand from another warehouse and deliver it to the customer.

4.1.1 Periodic Review; continuous demand

Let us assume that the weekly demand is continuously distributed with distribution function F, density function f and mean d. It is not to hard to see that the optimal inventory control policy is completely characterized by an ideal Monday morning inventory level, S^* say. And although this is in fact an infinite (or at least very long) horizon problem, it

reduces to a one week problem because of the very simple cost structure: replenishment is free!

So we have to find the optimal replenishment level S^*. In order to do so we look at the expected one week costs $C(S)$ for a given replenishment level S,

$$C(S) = h \int_0^S (S - x) f(x) dx + b \int_S^\infty (x - S) f(x) dx \ ,$$

which we can rewrite as

$$C(S) = (b + h) \int_0^S (S - x) f(x) dx + b(d - S) \ ,$$

with d the expected one week demand. By differentiating $C(S)$ with respect to S and setting the derivative equal to zero we get the following expression for the optimal value of S. (See Exercises.)

$$F(S^*) = \frac{b}{b + h}. \tag{4.1}$$

Example 4.1.1 *The demand for a certain product is approximately normally distributed with mean 64 and variance 64, so standard deviation 8. Further* $h = 1$ *and* $b = 20$. *Then we have that the optimal* S *satisfies* $F(S^*) = 20/21 = 0.9524$. *For the standard normal distribution this corresponds to the value* 1.67. *So the optimal replenishment level is* $S^* = 64 + 8 * 1.67 = 77.4$.

4.1.2 Periodic review; discrete demand

It is fairly straightforward to extend the results of the previous section for continuous demand to the case of discrete demand. But for discrete demand a more intuitive reasoning is possible. Let us say that at the end of the week we have S items in stock. Do we want to order one more item, Yes or No? In order to answer this question we have to balance the *expected increase in the inventory costs* and the *expected decrease of the shortage costs* if we increase the inventory to $S + 1$.

The extra inventory costs are h if the item is not sold and the decrease of the shortage costs is b if the item is sold. So, with D the random variable denoting the demand, the expected increase in inventory costs is $hP[D \leq S]$, whereas the expected decrease in shortage costs is $bP[D > S]$. So we can define

$$S^* = \min \{ \, S \, | \, hP[D \leq S] > bP[D > S] \, \} \ .$$

With $F(S) = P[D \leq S]$, this inequality can be rewritten as $hF(S) > b(1 - F(S))$, or $F(S) > b/(b + h)$. So

$$S^* = \min \left\{ \, S \, | \, F(S) > \frac{b}{b + h} \, \right\}.$$

4.1.3 Discrete versus continuous

Let us consider the case of a Poisson distributed demand with an average of 64. Then the variance is 64 as well. And take again $h = 1$ and $b = 20$. Using a simple program one may compute the optimal replenishment level $S^* = 78$. It is well-known that the Poisson distribution (for somewhat larger average values) can be approximated very well by the Normal distribution. As we have seen in the continuous example, this would lead to an optimal value of 77.4.

4.2 Age replacement

Consider a part that is known to break down after a random time with distribution F and density function f. One has two options, either to wait until the part breaks down, or to replace it by a new one after a certain time. As long as the part is in operation no signs are received that it might break down soon. One only knows F (and f). (Note that the replacement by a new one could also be a complete revision after which the part is 'as new' again.) The costs of a voluntary replacement are C_v, the costs of an emergency replacement are C_e. Usually one will have $C_e >> C_v$.
Our aim is to find the optimal age replacement strategy. Often a part deteriorates with age, so that the probability that it will break down increases in time. So we want to replace the part when the risk that it will break down becomes too high. It is intuitively obvious that the optimal strategy will be of the form *replace a part as soon as it reaches age T if it did not break down before T*, but it might also be optimal never to replace it voluntarily. A strategy of this form is called an *age replacement strategy.*

4.2.1 Failure rate

When we speak about about the risk of breaking down, we mean a conditional risk. That is, what is the 'break down rate' of the part at time t given that it still works at time t? This is called the *failure rate* and will be denoted as $\lambda(t)$. So

$$\lambda(t) = \frac{f(t)}{1 - F(t)}.$$

Clearly, replacement becomes more interesting when the failure rate is high.

We can distinguish three classes of distribution functions:

1. Increasing Failure Rate (IFR),
2. Constant Failure Rate (CFR), and
3. Decreasing Failure Rate (DFR).

The only continuous CFR distribution is the exponential distribution. Examples of IFR distributions are the uniform distribution and the Erlang (or Gamma) distribution. The hyper exponential distribution is an example of a DFR distribution. (See Exercises.)

4.2.2 The optimal strategy

In order to find the optimal replacement age T^*, let us look at the average costs per unit time. For this we introduce the concept *cycle*. A cycle is the time between two successive starts with a new part, after a voluntary or emergency replacement. Further denote by $C(t)$ the average costs in a cycle and by $D(t)$ the average duration of a cycle for replenishment age t. The corresponding average costs per time unit is denoted by $G(t)$ and satisfies

$$G(t) = \frac{C(t)}{D(t)}.$$

In order to compute $G(t)$ one has to distinguish the two cases of voluntary and emergency replacement. Doing so, leads to

$$C(t) = F(t)C_e + (1 - F(t))C_v$$

and

$$D(t) = (1 - F(t))t + \int_0^t xf(x)dx = \int_0^t (1 - F(x))dx.$$

(The second equality sign can be obtained by partial integration.) So,

$$G(t) = \frac{F(t)C_e + (1 - F(t))C_v}{\int_0^t (1 - F(x))dx}.$$

Since $F(t)$ is differentiable the same holds for $G(t)$. Further it is clear that $G(t)$ tends to infinity if t tends to 0. Differentiating $G(t)$ with respect to t and setting the derivative equal to 0 leads to

$$G'(t) = \frac{C'(t)D(t) - C(t)D'(t)}{D^2(t)} = 0.$$

Or (see Exercises)

$$f(t)\,(\,C_e - C_v\,)\int_0^t (1 - F(x))dx - (\,1 - F(t)\,)\,(\,C_v + F(t\,)(\,C_e - C_v\,)\,) = 0 \qquad (4.2)$$

Dividing by $C_e - C_v$ and $1 - F(t)$, using the failure rate $\lambda(t) = f(t)/(1 - F(t))$ and defining

$$C_Q = \frac{C_v}{C_e - C_v}$$

this simplifies to

$$\lambda(t)\int_0^t (1-F(x))dx - F(t) - C_Q = 0. \tag{4.3}$$

There are distribution functions for which the derivative of $C(t)$ has 0, 1 or more than 1 zeros. Below we will consider two special cases: the exponential distribution, and the uniform distribution.

4.2.3 The exponential distribution

The exponential distribution has the important property of being *memoryless*. Consider the exponential distribution with rate λ. Then the failure rate satisfies

$$\lambda(t) = \frac{f(t)}{1-F(t)} = \frac{\lambda e^{-\lambda t}}{1-(1-e^{-\lambda t})} = \lambda.$$

So the failure rate is constant. Also, in this case,

$$\lambda(t)\int_0^t (1-F(x))dx - F(t) - C_Q = \lambda\int_0^t e^{-\lambda x}dx - \lambda(1-e^{-\lambda t}) - C_Q = -C_Q\ .$$

So, equation (4.3) has no zero which again implies that a voluntary replacement does not make sense. The new part will have, statistically speaking, exactly the same future as the replaced one.
For the exponential distribution the optimal strategy is to wait till the part breaks down. Then the average costs per time unit are

$$G(\infty) = \frac{C_e}{1/\lambda} = \lambda C_e.$$

We also get this result by looking at the derivative of $G(t)$.

$$G^{'}(t) = \frac{\lambda\int_0^t e^{-\lambda x}dx - 1 + e^{-\lambda t} - C_Q}{(\int_0^t e^{-\lambda x}dx)^2} = \frac{-C_Q}{(\int_0^t e^{-\lambda x}dx)^2}.$$

So, the derivative is negative for all t. Hence $G(t)$ is decreasing in t and thus $G(t)$ is minimized in $t=\infty$.

4.2.4 The uniform distribution

The uniform distribution is not the most natural IFR distribution, but it is computationally simple. That is why this example is chosen. Consider the uniform distribution on

$[0, 1]$. It will clear that the closer the part will get to its maximal life time, the higher the failure rate will be. One easily sees

$$\lambda(t) = 1/(1-t), \quad 0 \leq t < 1.$$

If the part ages, we will have to replace it, but when? Looking for the 'zero' in equation (4.3) we get

$$\frac{1}{1-t}\int_0^t (1-x)dx - t - C_Q = 0.$$

Or

$$t^2/2 + C_Q t - C_Q = 0.$$

This quadratic equation has two zeroes, one of which is negative. The other one, denoted by T^*, satisfies

$$T^* = -C_Q + \sqrt{C_Q^2 + 2C_Q}.$$

Example. Take $C_e = 1000$ and $C_v = 100$, so $C_Q = 1/9$, then $T^* = (\sqrt{19} - 1)/9 = 0.373$. Thus the part is replaced when it has reached 37 percent of its maximal age. If $C_e = 2C_v$, so $C_Q = 1$, we get $T^* = (\sqrt{3} - 1) = 0.732$.

4.2.5 IFR and DFR distributions

In this section we will investigate the two classes of distributions DFR and IFR. In order to do so, we will look at the sign and possible changes of it in the function $G^{'}(t)$. For the sign the denominator $D^2(t)$ is not relevant. So let us look at the numerator. Define $H(t)$ by

$$H(t) := \lambda(t)\int_0^t (1-F(x))dx - F(t) - C_Q.$$

Assuming $F(0) = 0$ one gets $H(0) = -C_Q$. Now consider the derivative of $H(t)$ with respect to t.

$$H^{'}(t) = \lambda^{'}(t)\int_0^t (1-F(x))dx + \lambda(t)(1-F(t)) - f(t) = \lambda^{'}(t)\int_0^t (1-F(x))dx.$$

So the sign of the derivative of H is just the sign of the derivative of the failure rate. Thus we have the following result

Theorem 4.2.1 *If the age distribution has a DFR then it is optimal never to replace.*

Remark 4.2.2 *If the age distribution has an IFR then $H(t)$ increases. But $H(0)$ is negative, so even with $H(t)$ increasing it might never become 0. This will depend on*

$$H(\infty) = \lambda(\infty)L - 1 - C_Q,$$

with L the average life time of a part. So it is clear that if the failure rate tends to infinity, then the optimal strategy will prescribe to replace the part at some finite time.

4.3 Exercises

1. Derive Equation (4.1).
2. Derive Equation (4.2).
3. Show that the Erlang-2 distribution has an IFR and that a hyper exponential distribution with 2 phases has a DFR.

Chapter 5

Markov chains with rewards

Consider a finite Markov chain characterized by

- the finite set of states I ,
- the stages $0, 1, \ldots$,
- the transition probabilities p_{ij} from one stage to the next, and
- the rewards $r(i)$.

For the infinite horizon one considers two reward criteria: the *average reward* per period and the total expected *discounted reward*. We shortly recall these two criteria below.

5.1 Average reward per period

For an aperiodic Markov chain the average per period is defined as

$$\lim_{n\to\infty} \frac{1}{n} E \sum_{t=0}^{n-1} r(X_t) \ .$$

This average reward may depend of the initial state if there is more than one recurrent class. But if the chain has only one recurrent class one has

Theorem 5.1.1 *The average reward per period g in a Markov chain with only one recurrent chain is independent of the initial state and given by*

$$g = \sum_{i\in I} \pi_i r(i) \ ,$$

where π is the equilibrium distribution satisfying

$$\pi_i = \sum_{j\in I} \pi_j \, p_{ji} \, , \ i \in I \, , \qquad \sum_{i\in I} \pi_i = 1 \ .$$

5.2 Total expected discounted reward

A standard way of discounting is to correct the value of one unit of reward in period n by a factor β^n, for some $\beta < 1$. The total expected discounted reward for initial state i, denoted by $V(i)$ (suppressing β), is given by

$$V(i) = E_i \sum_{n=0}^{\infty} \beta^n r(X_n) \ .$$

Splitting the total expected reward in the reward in the first period and the expected reward from the next period onwards, then one gets the following the set of linear equations

$$V(i) = r(i) + \beta \sum_j p_{ij} V(j) \ , \ i \in I \ .$$

Using vector and matrix notation, this reads

$$V = r + \beta P V \ .$$

One easily shows that the solution to this system is unique:

$$V = (I - \beta P)^{-1} r \ .$$

Chapter 6

The infinite horizon discounted Markov decision process

6.1 Introduction

In chapter 3 we considered the finite horizon problem. In the next two chapters we consider the infinite horizon Markov decision process (Markov chain with decisions) further abbreviated as MDP.

Often in decision processes there is no clear limitation on the number of periods. E.g., although it is clear that the problem horizon is finite, it might be impossible to specify T. In those cases we usually prefer the infinite horizon model. No decision problem goes on forever without the rules of the game being changed, but for many problems the horizon is very long.
If we consider an infinite horizon, the sum of the rewards is not a very useful criterion. We will consider, as we did when we studied Markov chains, the discounted sum of the rewards and the average reward per period.
In this chapter we start with the expected discounted reward. So a reward of 1 at time n is worth only β^n at time 0. Formally $\beta < 1$ is merely a parameter, but practically one may give it a value that reflects the fact that one is less interested in what happens in one year from now than in what goes on today. If for instance, one values next year 25 percent less important than this year, and controls the system once a week, so 52 times a year, then one could take $\beta^{52} = 0.75$, so approximately $\beta = 0.9945$.

The decision points will be numbered $0, 1, 2, \ldots$ and are assumed to be equidistant with distance 1.

The following vector-matrix notation will be used. Let f be a stationary strategy, then

- $r(f)$ denotes the column vector of 1-period rewards, with i-th element $r(i, f(i))$,
- $P(f)$ denotes the 1-step transition matrix with ij-th element $p_{ij}^{f(i)}$.

- $V(f)$ denotes the total expected discounted reward when stationary strategy f is used, or, for the Markov chain in which in state i always action $f(i)$ is taken.

6.2 Preliminary analysis

Let $s = (f_0, f_1, \ldots)$ be a Markov strategy for the discounted MDP, where the policies are indexed with time (we can no longer use periods to go). This strategy, applied for a given initial state of the process, defines a stochastic process $\{X_n, B_n\}$ with X_n the state and B_n the decision at time n. We will write $E_{i,s}$ to indicate that the expectation is to be taken for initial state i ($P[X_0 = i] = 1$) and strategy s.

The total expected discounted reward for strategy s and for initial state i, will be denoted by $V(i, s)$, and satisfies

$$V(i,s) = E_{i,s} \sum_{n=0}^{\infty} \beta^n r(X_n, B_n) \ ,$$

or

$$V(i,s) = \sum_{n=0}^{\infty} \beta^n \sum_{j,k} P_{i,s}(X_n = j, B_n = k) r(j,k) \ .$$

And with $s = (f_0, f_1, \ldots)$,

$$V(i,s) = \sum_{n=0}^{\infty} \beta^n \sum_{j} P_{i,s}(X_n = j) r(j, f_n(j)) \ .$$

If, as we assumed, the number of states and the number of actions are finite, then the functions $V(i, s)$ are well defined. To see this, note that $|r(i, k)|$ is bounded, by A say, and hence the sum of the absolute rewards is bounded by $A/(1 - \beta)$.

Further define V^* as the optimal total expected discounted reward, so

$$V^*(i) = \sup_s V(i,s).$$

Our aim is to find a strategy that yields V^*. As will be shown hereafter, there exists a stationary strategy that achieves this maximal expected discounted reward for every initial state i.

As we have seen, a finite horizon problem can be solved by dynamic programming. Although we dealt with the non-discounted case only, it is easily seen that the same holds for the discounted finite horizon problem. Further, due to the discounting, what we do from time n onwards can cost us, compared to the optimal, at most $2A\beta^n/(1 - \beta)$. So,

for n sufficiently large, the optimal strategy for the finite horizon problem, extended with arbitrary decision rules for the later periods, will be nearly optimal for the infinite horizon problem as well.

Formally, let $s_n = (f_n, \ldots, f_1)$ be an optimal Markov strategy for the n-period discounted problem, and let $s_{n,f} = (f_n, \ldots, f_1, f, f, \ldots)$ be the extended strategy with policy f for the periods from n onwards. Then we have, with $V_n(i,s)$ the n-period reward for strategy s, and $V^*(i)$ the optimal expected discounted reward for initial state i,

Lemma 6.2.1 *For all i and n,*

$$V_n(i,s_n) - A\frac{\beta^n}{(1-\beta)} \leq V(i,s_{n,f}) \leq V^*(i) \leq V_n(i,s_n) + A\frac{\beta^n}{1-\beta} \ .$$

So, for n sufficiently large the strategy $s_{n,f}$ is nearly optimal for any f.

From this we also see that $V_n(i,s_n) = v_n(i)$ converges to $V^*(i)$. But this implies

Theorem 6.2.2 *For all $i \in I$*

$$\max_k \{r(i,k) + \beta \sum_j p_{ij}^k V^*(j)\} = V^*(i) \ .$$

Let $f^*(i)$ be the (a) maximizing action in state i, $i \in I$, then the policy f^* is a stationary optimal strategy for the discounted infinite horizon problem, thus $V(f^*) = V^*$.

Proof:
From

$$r(f^*) + \beta P(f^*)V^* = V^*$$

we get

$$V(f^*) = \lim_{N\to\infty} \sum_{n=0}^{N} \beta^n P^n(f^*) r(f^*) = \lim_{N\to\infty} \left(\ V^* - \beta^{N+1} P^{N+1}(f^*) V^* \ \right) = V^*.$$

Theorem 6.2.3 *V^* is the unique solution of the optimality equation*

$$\max_f \{ \, r(f) + \beta P(f) v \, \} = v \ . \tag{6.1}$$

Proof:
Using the same reasoning as in the previous proof, we get for a solution v of (6.1) and a policy f_v which maximizes (6.1)

$$V(f_v) = v \ .$$

And for any other policy

$$V(f) \leq v \ .$$

So

$$V^* = V(f^*) \leq v = V(f_v) \leq V^* \ .$$

Hence $v = V^*$.

One may easily show that in the infinite horizon a stationary strategy is optimal if and only if it satisfies the optimality equation. See Exercise 1.

6.3 Stationary strategies

Although we know that an optimal stationary strategy exists, we cannot easily find one, because V^* is not known. And from a practical point of view, the Markovian strategies obtained by dynamic programming are not satisfactory.

In the next three sections we will present three different algorithms by which the discounted problem can be 'solved'. The first one is called *value iteration* or *successive approximation*. In this approach we compute the value of finite horizon problems and we show how from that we can derive a nearly optimal stationary strategy. In the second algorithm, called *policy iteration* we iterate in the space of stationary strategies. Given a stationary strategy we compute a better one, until we have found an optimal one. The third algorithm is based on a linear programming formulation.

6.4 Successive approximation

The successive approximation or value iteration approach is based on the fact that it is possible to give upper and lower bounds on the expected infinite horizon reward of a stationary strategy by looking at a 1-period problem.

Before formulating the result, some notation is introduced.

- $\min(y)$ and $\max(y)$ denote the value of the minimal and maximal element of the vector y,
- the *span* of a vector, i.e., the difference between the largest and smallest component, is denoted by $sp(y)$, so $sp(y) = \max(y) - \min(y)$,
- the column vector of ones is denoted by e, so $e(i) = 1, \ i \in I$.

Then

Lemma 6.4.1 *Let v and w be arbitrary column vectors.*
a) If $r(f) + \beta P(f)v \leq w$, then

$$V(f) \leq w + \frac{\beta}{1-\beta} \max(w - v)e \ .$$

b) If $r(f) + \beta P(f)v \geq w$, then

$$V(f) \geq w + \frac{\beta}{1-\beta} \min(w - v)e \ .$$

Proof: We only prove a), the proof of b) being similar.
First write

$$r(f) \leq w - \beta P(f)v \ ,$$

and

$$V(f) = \sum_{n=0}^{\infty} \beta^n P^n(f) r(f) \ .$$

Substituting the first expression into the second one gives

$$V(f) \leq \sum_{n=0}^{\infty} \beta^n P^n(f)(w - \beta P(f)v) = w + \sum_{n=1}^{\infty} \beta^n P^n(f)(w - v) \ .$$

The result then directly follows by bounding $w - v$ from above by the constant vector with its largest component, and using $P(f)e = e$.

From this Lemma we immediately get the following theorem.

Theorem 6.4.2 *Consider a successive approximation process. Let v_{n+1} be the last computed value vector and let f_{n+1} satisfy*

$$r(f_{n+1}) + P(f_{n+1})v_n = \max_f \{ r(f) + P(f)v_n \} = v_{n+1} \ .$$

Then

$$v_{n+1} + \frac{\beta}{1-\beta} \min(v_{n+1} - v_n)e \leq V(f_{n+1}) \leq V^* \leq v_{n+1} + \frac{\beta}{1-\beta} \max(v_{n+1} - v_n)e \ . \quad (6.2)$$

So,

$$V(f_{n+1}) \geq V^* - \frac{\beta}{1-\beta} sp(v_{n+1} - v_n)e \ . \quad (6.3)$$

From this we see that if $sp\,(v_{n+1} - v_n)$ is sufficiently small, then the stationary strategy f_{n+1} is nearly optimal. It is clear that $sp\,(v_{n+1} - v_n)$ decreases at least at the rate β. But one may show that, if the Markov decision process is unichained and not periodic, i.e., the stationary strategies of interest are aperiodic and have only one recurrent chain, then the convergence goes much faster. Formally,

$$sp(v_{n+1} - v_n) = O(\alpha^n) \quad (n \to \infty) \ ,$$

where α is problem dependent and usually considerably smaller than β.

Summarizing, we have the following algorithm to solve the discounted Markov decision process.

Successive approximation algorithm

- *Choose the value of the initial vector v_0, e.g., $v_0 = 0$.*
- *Compute v_{n+1} and f_{n+1} such that*

$$r(f_{n+1}) + \beta P(f_{n+1})v_n = v_{n+1} = \max_f \{r(f) + \beta P(f)v_n\}.$$

- *Until $sp(v_{n+1} - v_n)$ is sufficiently small.*

6.5 Policy iteration

The second technique used to solve the discounted Markov decision process is a form of strategy improvement. The improvement step is based on Lemma 6.4.1, with v replaced by $V(f_n)$.

Theorem 6.5.1 *Let f_{n+1} satisfy*

$$r(f_{n+1}) + \beta P(f_{n+1})V(f_n) = \max_f \{r(f) + \beta P(f)V(f_n)\} \geq r(f_n) + \beta P(f_n)V(f_n) = V(f_n) \tag{6.4}$$

then

$$V(f_{n+1}) \geq r(f_{n+1}) + \beta P(f_{n+1})V(f_n) \geq V(f_n) \ . \tag{6.5}$$

This leads us to the following algorithm.

Policy iteration

- *Choose an initial policy* f_0 *and determine* $V(f_0)$.
- *For* $n = 0, 1, \ldots$ *determine*

$$r(f_{n+1}) + \beta P(f_{n+1})V(f_n) = \max_f\{r(f) + \beta P(f)V(f_n)\} \quad \text{and} \quad V(f_{n+1}) \ .$$

- *Until*

$$r(f_{n+1}) + \beta P(f_{n+1})V(f_n) = V(f_n) \ .$$

As long as f_n is not optimal, i.e., $\max_f\{r(f) + \beta P(f)V(f_n)\} \neq V(f_n)$, we can improve f_n. Formally, since the number of policies is finite, the algorithm will converge. However, the number of policies can be tremendously large. Fortunately, in practice a few, 3 up to 20, iterations suffice. The complexity of the algorithm is determined mainly by the complexity of solving $V(f_n)$.

An interesting mixture of successive approximation and policy iteration is an algorithm in which the improvement step in which f_{n+1} is determined is not followed by the computation of $V(f_{n+1})$. Note that, particularly when the number of states is large, the computation of $V(f_{n+1})$ becomes expensive. We will call this variant *Policy approximation*. We need some additional notation. Let f be a stationary strategy, then the operator $L(f)$ is defined by

$$L(f)v := r(f) + \beta P(f)v \ .$$

Policy approximation

- *Choose* v_0 *such that* $\max_f\{r(f) + \beta P(f)v_0\} \geq v_0$ *and some* $m \geq 1$.
- *Compute for* $n = 0, 1, ...$

$$r(f_{n+1}) + \beta P(f_{n+1})v_n = \max_f\{r(f) + \beta P(f)v_n\} \quad \text{and} \quad v_{n+1} = L^m(f_{n+1})v_n \ ,$$

- *Until*

$$sp(r(f_{n+1}) + \beta P(f_{n+1})v_n - v_n)$$

is sufficiently small.

6.6 Linear Programming

There is a third technique for solving Markov decision processes which seems to be of a quite different nature, linear programming. We will first derive the LP-formulation of the problem.

As we have seen already, the total expected discounted reward for a strategy and a given initial state i_0 can be written as

$$V(i_0, s) = \sum_{n=0}^{\infty} \beta^n \sum_{j,k} P_{i_0,s}(X_n = j, B_n = k) r(j,k) \ .$$

It is not certain that an optimal strategy for initial state i is also optimal for initial state j, because when starting in state i you might never reach a transient state j in which case the actions prescribed in state j are irrelevant for initial state i. Therefore, we look at a slightly different problem formulation in which not an initial state is given, but an initial distribution, $\pi(0)$, say with $\pi_i(0) = 1/N$ for all i.
So, we want to optimize

$$V(\pi(0), s) = \sum_{n=0}^{\infty} \beta^n \sum_{i,k} P_{\pi(0),s}(X_n = i, B_n = k) r(i,k) \ .$$

Writing

$$x_i^k(n, s) = P_{\pi(0),s}(X_n = i, B_n = k)$$

the object function becomes

$$\sum_{n=0}^{\infty} \beta^n \sum_{i,k} x_i^k(n, s) r(i,k) \ .$$

Note that we have suppressed the dependence on s. In other words, whatever s may be, there are constraints. Clearly, $\sum_{i,k} x_i^k(n) = 1$ for all n. But also the transition probabilities limit the freedom of the $x_i^k(n)$. In order to reach state j at time $n+1$ the system has to make a transition from its state at time n into state j.

$$\sum_k x_j^k(n+1, s) = \sum_{i,l} x_i^l(n, s) p_{ij}^l \ , \ j \in I \ , \ n \geq 0 \ .$$

Further we have,

$$\sum_k x_i^k(0, s) = \pi_i(0) \ .$$

This suggests the following LP-formulation

LP-1

Maximize

$$\sum_{n=0}^{\infty} \beta^n \sum_{i,k} x_i^k(n) r(i,k) \ ,$$

Subject to

$$\sum_k x_j^k(n+1) = \sum_{i,l} x_i^l(n) p_{ij}^l \ , \quad j \in I, \ n \geq 0, \tag{6.6}$$

$$\sum_k x_i^k(0) = \pi_i(0) \ , \quad i \in I \ , \tag{6.7}$$

$$x_i^k(n) \geq 0 \ , \quad i \in I, \ k \in K(i), \ n \geq 0 \ . \tag{6.8}$$

Although we now have a LP-formulation, it is not a solvable problem. The number of variables as well as the number of constraints is not finite, but this is caused by the time parameter n only. Therefore let us use the substitution

$$x_i^k = \sum_{n=0}^{\infty} \beta^n x_i^k(n) \ .$$

Then the object function becomes $\sum_{i,k} x_i^k r(i,k)$. The constraints have to be 'transformed' as well. A natural way to try to achieve this, is by multiplying the constraints by powers of β and summing them. This way we get

$$\sum_{n=0}^{\infty} \beta^{n+1} \sum_k x_j^k(n+1) = \beta \sum_{n=0}^{\infty} \beta^n \sum_{i,l} x_i^l(n) p_{ij}^l \ ,$$

so

$$\sum_k x_j^k - \pi_j(0) = \beta \sum_{i,l} x_i^l p_{ij}^l \ .$$

This leads to the following simpler LP-formulation.

LP-2

Maximize

$$\sum_{i,k} x_i^k r(i,k) \ ,$$

Subject to

$$\sum_k x_j^k - \beta \sum_{i,l} x_i^l p_{ij}^l = \pi_j(0) \ , \quad j \in I, \tag{6.9}$$

$$x_i^k \geq 0 \ , \quad i \in I, \ k \in K(i) \ . \tag{6.10}$$

This second LP-problem has N constraints, and hence any basic solution has at most N nonnegative variables. On the other hand, since we have taken $\pi_j(0)$ positive for all j, at least one of the x_j^k must be positive for each j. So, we can conclude that for each j there is exactly one k for which x_j^k is positive. In other words, each basic solution seems to correspond with a stationary strategy.

In order to show that this LP-problem is equivalent to the original discounted Markov decision process we still have to show one more thing. We already know that we can restrict ourselves to stationary strategies. What remains to be shown is that if we take a stationary strategy f and construct the corresponding $x_i^{f(i)}(n)$ and from that the $x_i^{f(i)}$, thus obtaining a vector $x(f)$ say, that then this vector is precisely the basic solution of LP-2 that corresponds to f. And also we have to show that the corresponding values of the object function coincide. This is left as an exercise (Exercise 2).

Now let us look at this LP-2 problem in some more detail. In the simplex algorithm we move from one basic solution to the next, so from stationary strategy to stationary strategy, just like in the policy iteration algorithm. Thus, we want to understand what the similarities between the two algorithms are. In order to investigate this we first construct the following LP-tabloid.

x_1^1	x_1^2	$x_1^{K_1}$	x_2^1	$x_1^{K_2}$	x_N^1	$x_N^{K_N}$	V	
$1-\beta p_{11}^1$	$1-\beta p_{11}^2$	$1-\beta p_{11}^{K_1}$	$-\beta p_{21}^1$	$-\beta p_{21}^{K_2}$	$-\beta p_{N1}^1$	$-\beta p_{N1}^{K_N}$	0	$\pi_1(0)$
$-\beta p_{12}^1$	$-\beta p_{12}^2$	$-\beta p_{12}^{K_1}$	$1-\beta p_{22}^1$	$1-\beta p_{22}^{K_2}$	$-\beta p_{N2}^1$	$-\beta p_{N2}^{K_N}$	0	$\pi_2(0)$
$-\beta p_{1N}^1$	$-\beta p_{1N}^2$	$-\beta p_{1N}^{K_1}$	$-\beta p_{2N}^1$	$-\beta p_{2N}^{K_2}$	$1-\beta p_{NN}^1$	$1-\beta p_{NN}^{K_N}$	0	$\pi_N(0)$
r(1,1)	r(1,2)	$r(1,K_1)$	r(2,1)	$r(2,K_2)$	r(N,1)	$r(N,K_N)$	-1	0

So we have added an equation for the value function, $V = \sum_{i,k} x_i^k r(i,k)$. This equation will be changed using pivot operations until the value equation has the form

$$V = V^* - \sum x_i^k d(i,k) \ ,$$

with all $d(i,k)$ nonnegative, and the $d(i,k)$ corresponding to the present basic solution equal to 0. Then it is clear that for any other basic solution the value will be at most v_0, and that the present basic solution is optimal.

Now, suppose we have a basic solution, f say. Then, in order to rewrite the value equation such such the $d(i,k)$ corresponding to the basic variables are 0, we have to add a

linear combination of the constraints to the first value equation $V - \sum_{i,k} x_i^k r(i,k) = 0$. If constraint j is added v_j times, then these v_j should satisfy

$$d(i, f(i)) = v_i - r(i, f(i)) - \beta \sum_i p_{ij}^{f(i)} v_j = 0 \ , i \in I \ .$$

But as we know, this set of equations, which in matrix-vector notation reads, $v = r(f) + \beta P(f)v$ has the unique solution $v = V(f)$. So, finding the vector v is just the same as computing the value of a stationary strategy.

Now, in order to verify the optimality of f we have to look at the other constants $d(i,k)$ in the value equation. For these we get

$$d(i,k) = v_i - r(i,k) - \beta \sum_i p_{ij}^k v_j \ .$$

So, we see that all $d(i,k)$ are nonnegative if v satisfies the optimality equation, i.e.,

$$v_i = \max_k \{r(i,k) + \beta \sum_j p_{ij}^k v_j\} \ , \quad i \in I \ .$$

If not, then there is at least one state i and one action $k \in K(i)$ for which, now writing $V(f)$ instead of v,

$$r(i,k) + \beta \sum_j p_{ij}^k V_j(f) > V_i(f) \ .$$

So, we see that there is a lot of similarity between the LP approach and the policy iteration algorithm. There are, however, some important differences. In LP we normally change basic variables one at the time. In policy iteration we use the structure of the problem to find a new action, basic variable) in each state. If we do this in the LP approach as well, then there are no differences any more.

6.6.1 Additional constraints

An advantage of the LP approach is the fact that we can add constraints. For instance, if we have an inventory control model, then we can use costs, but we can also add a constraint stating than at least 95 percent of all demand should be satisfied directly from stock. Note however, that if we do so, then the number of constraints will be larger than N, and hence a basic solution may have more than N non zero variables. Thus in at least one state there are 2 actions which are used. This means that for the problem with additional constraints the basic solutions correspond to *randomized stationary strategies*. I.e., in one or more states a randomized action is used.

It is not possible to adapt the successive approximation and policy iteration algorithms so that they can deal with additional constraints of this type.

6.7 Exercises

1. Show that in the infinite horizon a stationary strategy is optimal if and only if it satisfies the optimality equation.

2. Show that if for a stationary strategy f we construct the corresponding $x_i^{f(i)}(n)$ and from that the $x_i^{f(i)}$, thus obtaining a vector $x(f)$ say, that then this vector is precisely the basic solution of LP-2 that corresponds to f. And also show that the corresponding values of the object function coincide.

Chapter 7

The average reward problem

7.1 Introduction

At first sight there seems to be a big difference between the discounted case and the average reward case. In the discounted model the first periods are important and what happens very far in the future has hardly any influence on the total payoff. In the average reward case we are only interested in the long run average, which means that in most cases the actions we take in the beginning are not important at all. Although this is true, we will see that if we are using stationary strategies only, the differences become very small. In this chapter we spend some time on the relation between these two cases. The main topic of the chapter is (as in the discounted case) the computation of stationary (nearly) optimal strategies.
Here we concentrate on the case that for each (relevant) stationary strategy the resulting Markov chain has a *single recurrent chain*. We also assume that these chains are *aperiodic*. However, much of what follows can be translated to the general case of possibly more than one recurrent chain, i.e., with an average reward per time unit which may depend on the initial state.

7.2 The average reward for a stationary strategy

Let us consider a stationary strategy f. If the Markov chain for strategy f is aperiodic, then we have

$$\lim_{t\to\infty} P^t(f) = P^*(f)$$

If $P(f)$ has a single recurrent chain, then $P^*(f)$ has identical rows (the equilibrium distribution).
The average reward per unit time for f is denoted $g(f)$ and defined as

$$g(f) = P^*(f)r(f) \ .$$

So, if $P(f)$ has a single recurrent chain, then $g(f)$ is a constant vector.
Then, the n-period $V_n(f)$ reward satisfies

$$\begin{aligned} V_n(f) &= \sum_{t=0}^{n-1} P^t(f) r(f) = \sum_{t=0}^{n-1} \{ P^*(f) - (P^t(f) - P^*(f)) \} r(f) && (7.1) \\ &= ng(f) + d(f) + o(1) \quad (n \to \infty) , && (7.2) \end{aligned}$$

where we define $d(f)$, the so-called *relative value function*, by

$$d(f) = \sum_{t=0}^{\infty} \{ P^t(f) - P^*(f) \} r(f) .$$

Since $P^t(f)$ converges to $P^*(f)$ exponentially fast, the above sum is well defined.
In the literature, this relative value function is often denoted by the letter V. In order to avoid confusion we here use the letter d, say for difference.

In case the Markov chain is periodic the results change slightly. We can define the 'average limiting matrix' by

$$P^*(f) = \lim_{t \to \infty} \frac{1}{t} \sum_{n=0}^{t-1} P^n(f) .$$

This matrix plays the same role as the limit matrix in the aperiodic case. Also the definition of $d(f)$ has to be changed from a limit to an average limit.

The following result holds for both periodic and aperiodic chains.

Theorem 7.2.1 *For any stationary strategy f we have*

$$\begin{aligned} P(f)g(f) &= g(f) , && (7.3) \\ r(f) + P(f)d(f) &= g(f) + d(f) , && (7.4) \\ P^*(f)d(f) &= 0 . && (7.5) \end{aligned}$$

Proof:
The first equation has no importance in case $g(f)$ is a constant vector. The second equation follows directly from the recursion which expresses V_{n+1} in $V_n(f)$,

$$V_{n+1}(f) = r(f) + P(f)V_n(f) = r(f) + P(f)\{ng(f) + d(f) + o(1)\}$$

and

$$V_{n+1}(f) = (n+1)g(f) + d(f) + o(1) .$$

Multiplying the previous equation with $P^*(f)$, the third one follows from

$$P^*(f) = P^*(f)P(f) = P(f)P^*(f) \ ,$$

and

$$P^*(f)g(f) = g(f) \quad \textit{and} \quad P^*(f)V_n(f) = ng(f).$$

We have even more: $g(f)$ and $d(f)$ not only satisfy this set of equations, they are the unique solution.

Theorem 7.2.2 *The vectors $g(f)$, $d(f)$ are the unique solution of the set*

$$\begin{aligned} P(f)g &= g \ , && (7.6)\\ r(f) + P(f)d &= g + d \ , && (7.7)\\ P^*(f)d &= 0 \ . && (7.8) \end{aligned}$$

Proof:
By iteration the first equation gives $P^*(f)g = g$. Multiply the second equation with $P^*(f)$ to get $g = P^*(f)r(f)$, so $g = g(f)$, i.e., the solution for g is unique. Now suppose there are two solutions for d, d_1 and d_2 say. If we substitute them into the second equation and subtract the two expressions, we get $P(f)(d_1 - d_2) = d_1 - d_2$, hence by iteration $P^*(f)(d_1 - d_2) = d_1 - d_2$, which, because of third equation, is equal to 0. Hence also $d_1 = d_2$.

7.3 Discounted versus average reward

The total expected discounted reward for a strategy f can be written as

$$\begin{aligned} V_\beta(f) &= \sum_{n=0}^{\infty} \beta^n P^n(f) r(f) && (7.9)\\ &= \sum_{n=0}^{\infty} \beta^n \left\{ P^*(f) - (P^n(f) - P^*(f)) \right\} r(f) && (7.10)\\ &= \frac{1}{1-\beta} g(f) + d(f) + \sum_{n=0}^{\infty} (1-\beta^n) \left\{ P^n(f) - P^*(f) \right\} r(f) \ . && (7.11) \end{aligned}$$

Since in the aperiodic case $P^n(f) - P^*(f)$ converges to zero, it is clear that the last term vanishes if β goes to 1. (In the periodic case this term also vanishes because the average of $P^n(f) - P^*(f)$ converges to 0.) So, just like in the average reward case, the payoff is governed by the two terms, $g(f)$ and $d(f)$. So, if β is sufficiently close to 1, a stationary

strategy that is optimal in the discounted problem, will be optimal for the average reward case as well.

Now consider a sequence of discount factors β_n tending to 1. Let f_n be optimal for discount factor β_n. Then, as there are only finitely many policies, the sequence of policies $\{f_n\}$ contains at least one policy infinitely often. Let us denote this policy (one of these policies) by f^*. Then we have

Theorem 7.3.1 *For all f and for all i*

$$g(f^*) \geq g(f) \ ,$$

and

$$if \quad g(f^*) = g(f) \quad then \quad d(i, f^*) \geq d(i, f) \ .$$

Corollary 7.3.2 *So f^* is superior within the class of stationary strategies. As we will show, f^* is also average optimal within the set of all strategies. We will denote this optimal average value by g^*, so $g^* = g(f^*)$.*

From the optimality equation for the sequence of discount factors for which f^* is optimal, we also have

Theorem 7.3.3 *Let β_m be a subsequence of discount factors for which f^* is optimal. Then for all f*

$$\begin{aligned} r(f) \quad &+ \quad \beta_m P(f) \{ \frac{1}{1-\beta_m} g(f^*) + d(f^*) + o(1) \} \\ &\leq \quad r(f^*) + \beta_m P(f^*) \{ \frac{1}{1-\beta_m} g(f^*) + d(f^*) + o(1) \} \\ &= \quad \frac{1}{1-\beta_m} g(f^*) + d(f^*) + o(1) \quad (\beta_m \uparrow 1) \ . \end{aligned}$$

or

$$\begin{aligned} r(f) \quad &+ \quad P(f) d(f^*) + o(1) \\ &\leq \quad r(f^*) + P(f^*) d(f^*) + o(1) \\ &= \quad g(f^*) + d(f^*) + o(1) \quad (\beta_m \uparrow 1) \ . \end{aligned}$$

Hence

$$\max_f \{ r(f) + P(f) d(f^*) \} = g(f^*) + d(f^*) \ . \tag{7.12}$$

Denote the subset of maximizing policies in (7.12) by F^*. Then we have

Corollary 7.3.4 *For all $h \in F^*$*

$$V_n(h) = ng^* + d^* - P^n(h)d^* \ ,$$

so any policy $h \in F^*$ is average optimal.

From equation (7.12), and g^* being a constant vector, we immediately get

Theorem 7.3.5 *For any strategy s*

$$V_n(s) \leq ng^* + d^* - E_s(d^*(X_n)) \ ,$$

hence

$$g(s) \leq g^* \ .$$

Exploiting the analogy between the two reward structures we again derive the three algorithms to obtain good or optimal stationary strategies: successive approximation, policy iteration and linear programming.

7.4 Successive approximation

In the discounted case, the successive approximation approach was based on a Lemma about upper and lower bounds. In the average reward model we use the following simpler result.

Lemma 7.4.1 *Let v be an arbitrary vector, then*

$$\min\left(\, r(f) + P(f)v - v \,\right) e \leq g(f) \leq \max\left(\, r(f) + P(f)v - v \,\right) e \ .$$

Proof
This result follows directly from

$$g(f) = P^*(f)r(f) = P^*(f)\left(\, r(f) + P(f)v - v \,\right) \ .$$

And also

Lemma 7.4.2 *Let v be an arbitrary vector, and let*

$$r(f_v) + P(f_v)v - v = \max_f \{ r(f) + P(f)v - v \} .$$

then

$$\min (r(f_v) + P(f_v)v - v) e \leq g(f_v) \leq g^* \leq \max (r(f_v) + P(f_v)v - v) e.$$

Proof
The third inequality follows directly from the fact that

$$r(f^*) + P(f^*)v - v \leq r(f_v) + P(f_v)v - v$$

implies

$$g^* = g(f^*) = P^*(f^*) (r(f^*) + P(f^*)v - v) \leq P^*(f^*) (r(f_v) + P(f_v)v - v) .$$

This gives us the average reward version of the successive approximation algorithm.

Successive approximations

- *Choose an initial vector v_0.*
- *Compute for $n = 0, 1, \ldots$*

$$v_{n+1} = \max_f \{ r(f) + P(f)v_n \} ,$$

- *Until $sp (v_{n+1} - v_n)$ is sufficiently small .*
- *Then the stationary strategy f_{n+1} is (nearly) optimal and*

$$\min (v_{n+1} - v_n) \leq g_{f_{n+1}} \leq \max (v_{n+1} - v_n) .$$

Remark 7.4.3 So far we paid no attention to two special cases that deserve some attention. Firstly, in the merely theoretical case of a Markov decision process with more than one recurrent chain, the above algorithm makes no sense at all. Although the result remains true, the span that indicates the quality of the stationary strategy does not converge to 0 as $sp(g^*)$ will be positive. Secondly, if the Markov decision process is periodic, then the $sp(v_{n+1} - v_n)$ will not converge to 0 as well. In that case we can adapt the algorithm such that we look at the reward in one cycle.

Example 7.4.4 Consider the following inventory problem. A product is delivered from stock. Once a week it is decided whether or not to order a new batch of 5. The maximal inventory is 7. The demand is stochastic. With $d(i)$ the probability that the demand equals i, we have $d(0) = 0.3$, $d(1) = 0.4$, $d(2) = 0.25$ and $d(3) = 0.05$. Further we have inventory costs of 1 per unit for every item that we have in stock at the end of a period and shortage costs of 10 for every item that we cannot deliver from stock. The fixed costs for ordering a batch are 20.

Table 7.4 shows the results of the successive approximation algorithm for example 7.4.4.

n	0	1	2	3	4	5	6	7	sp	g^*
1	10.50	3.80	1.50	1.95	2.95	3.95	4.95	5.95	9.000	6.000
2	21.00	12.29	6.62	4.61	5.18	6.88	8.85	10.85	8.270	6.365
3	29.57	22.19	14.70	10.10	8.62	9.57	11.88	14.72	7.206	6.294
4	33.53	31.15	23.66	17.89	14.36	13.53	15.00	17.94	5.847	6.044
5	39.41	36.62	31.12	26.24	21.88	19.41	19.35	21.43	4.866	5.924
6	46.64	42.37	37.30	33.39	29.62	26.64	25.30	26.06	3.107	6.185
7	54.01	49.16	43.63	39.81	36.64	34.01	32.27	32.03	1.396	6.665
8	60.94	56.35	50.46	46.34	43.23	40.94	39.38	38.80	0.668	6.857
9	67.63	63.37	57.46	53.17	49.89	47.63	46.27	45.74	0.360	6.833
10	74.36	70.15	64.37	60.11	56.72	54.36	53.01	52.58	0.212	6.833
11	81.19	76.90	71.18	66.99	63.61	61.19	59.78	59.36	0.148	6.819
12	88.06	83.70	77.97	73.80	70.47	68.06	66.61	66.15	0.077	6.828
13	94.90	90.55	84.79	80.61	77.29	74.90	73.46	72.98	0.047	6.828
14	101.73	97.40	91.63	87.43	84.11	81.73	80.30	79.82	0.032	6.831
15	108.55	104.23	98.47	94.27	90.93	88.55	87.13	86.65	0.017	6.830
16	115.38	111.05	105.30	101.10	97.76	95.38	93.96	93.48	0.010	6.829
17	122.21	117.88	112.12	107.93	104.60	102.21	100.78	100.31	0.006	6.830
18	129.04	124.71	118.95	114.76	111.43	109.04	107.61	107.14	0.003	6.830
19	135.87	131.54	125.78	121.59	118.26	115.87	114.44	113.97	0.002	6.830
20	142.70	138.37	132.61	128.42	125.08	122.70	121.27	120.80	0.001	6.830

Table 7.1: Successive approximations 7.4.4

The first column gives the number of the iteration, the next 8 columns give the v_n values for the different initial states. Column number 10 gives the span of $v_{n+1} - v_n$ and the last column gives the average of $min(v_{n+1} - v_n)$ and $max(v_{n+1} - v_n)$.
As we see the algorithm converges fast. The optimal strategy 'order in state 0 only' appears already in the third iteration. And after 20 iterations the span is nearly 0.

7.5 Policy iteration

Suppose we have a stationary strategy f_n, and that we have computed the constant vector $g(f_n)$ and the relative value vector $d(f_n)$. Then one might consider the following attempt to improve a stationary strategy.

Policy Improvement Step

- *Find a policy f_{n+1} satisfying*

$$r(f_{n+1}) + P(f_{n+1})d(f) = \max_f\{r(f) + P(f)d(f)\}$$

  with f_{n+1} equal (!!) to f_n if possible, i.e., in all states i where $f_n(i)$ is a maximizer.

- *Compute $g(f_{n+1})$ and $d(f_{n+1})$.*

Define γ by

$$r(f_{n+1}) + P(f_{n+1})d(f_n) = g(f_n) + d(f_n) + \gamma \ . \tag{7.13}$$

Clearly $\gamma \geq 0$ and, as long as f_{n+1} is not equal to f_n, also $\gamma \neq 0$.

That f_{n+1} is indeed an improvement is stated in the following theorem.

Theorem 7.5.1 *If $\gamma \neq 0$ then f_{n+1} is better than f_n, i.e., either $g(f_{n+1}) > g(f_n)$ or $g(f_{n+1}) = g(f_n)$ and $d(f_{n+1}) \geq d(f_n)$ with inequality in at least one state. So f_{n+1} is 'lexicographically' better than f_n.*

Proof.
In order to show that f_{n+1} improves f_n multiply both sides in equation (7.13) with $P^*(f_{n+1})$. This gives

$$g(f_{n+1}) = g(f_n) + P^*(f_{n+1})\gamma \geq g(f_n) \ .$$

So if γ is positive in at least one of the recurrent states for policy f_{n+1}, then $g(f_{n+1}) > g(f_n)$ and we are done. So assume this is not the case. Then the two policies are the same on the recurrent states and γ is positive in at least one of the transient states under f_{n+1}. Also

$$r(f_{n+1}) + P(f_{n+1})d(f_{n+1}) = g(f_{n+1}) + d(f_{n+1}) \ . \tag{7.14}$$

Subtracting (7.13) from (7.14) and writing $\Delta d = d(f_{n+1}) - d(f_n)$ gives

$$P(f_{n+1})\Delta d = \Delta d - \gamma \ .$$

So by iteration

$$P^*(f_{n+1})\Delta d \leq \Delta d - \gamma \ .$$

But the two policies are equal in the recurrent states, hence the d's must be equal in the recurrent states as well. So $P^*(f_{n+1})\Delta d = 0$. Hence $\Delta d \geq \gamma$. So $d(f_{n+1}) \geq d(f_n)$ and $d(f_{n+1}) \neq d(f_n)$, which completes the proof.

Remark 7.5.2 *Note that we used the fact that the two policies are the same whenever possible. Without that, cycling may occur and Theorem 7.5.1 is no longer true.*

What remains to be shown is that if one cannot find a real improvement anymore, the policy is average optimal. So, let f_n be the policy that cannot be improved. Let f be an arbitrary policy, then we can define for these two policies the non-positive vector δ satisfying

$$r(f) + P(f)d(f_n) = g(f_n) + d(f_n) + \delta \ .$$

Since δ is non-positive, multiplication with $P^*(f)$ immediately gives $g(f) \leq g(f_n)$. As f is arbitrary, this holds for all f and thus f_n is average reward optimal.

So we can formulate the following average reward version of the policy iteration algorithm.

Policy Iteration Algorithm

Policy Improvement Step

- *Find a policy f_{n+1} satisfying*

$$r(f_{n+1}) + P(f_{n+1})d(f_n) = \max_f\{r(f) + P(f)d(f_n)\}$$

with f_{n+1} equal (!!) to f_n if possible, i.e., in all states i where $f_n(i)$ is a maximizer.

Policy Evaluation Step

- *Compute $g(f_{n+1})$ and $d(f_{n+1})$.*

Stop

- *Until $f_{n+1} = f_n$.*

7.6 The equations for g and d

We have seen that $g(f)$ and $d(f)$ are uniquely determined by

$$\begin{aligned} r(f) + P(f)d &= g + d \ , & (7.15) \\ P^*(f)d &= 0 \ . & (7.16) \end{aligned}$$

The first equation already specifies the vector d up to a constant. That constant is fixed by the second equation, but for many computations that constant is not important. Therefore the latter relation is often simplified to

$$d_N = 0 \ . \tag{7.17}$$

7.7 Linear programming

As in the discounted case the decision problem can also be solved by linear programming. We will do so for the case that all Markov chains corresponding to stationary strategies are *irreducible*, i.e., one recurrent class and *no transient states*.

Let f be an arbitrary stationary strategy. Then the corresponding average reward g_f satisfies

$$g_f = \sum_{i \in I} p_i^{(\infty)} r(i, f(i)) \ ,$$

where the $p_i^{(\infty)}$ are the unique solution of the equations

$$\begin{aligned} x_i &= \sum_{j \in I} x_j p_{ji}^{f(i)} \ , \ i \in I \ , & (7.18) \\ \sum_{i \in I} x_i &= 1 \ , & (7.19) \\ x_i &\geq 0 \ , \ i \in I \ . & (7.20) \end{aligned}$$

A stationary strategy f can be characterized by the set $\{d_i^k(f)\}$ with $d_i^k(f) = 1$ if $k = f(i)$ and 0 otherwise for all $i \in I$. Clearly there is a one to one correspondence between the stationary strategies and the sets $\{d_i^k\}$ with $d_i^k \in \{0, 1\}$ for all i and k and $\sum_k d_i^k = 1$ for all i. So one may write

$$p_{ij}^{f(i)} = \sum_k p_{ij}^k d_i^k(f) \ .$$

This enables us to write the problem of finding the optimal g_f as a mathematical programming problem.

MPP

Maximize

$$\sum_{i,k} x_i d_i^k r(i,k)$$

subject to

$$\begin{aligned}
x_i &= \sum_{j,k} x_j d_j^k p_{ji}^k \ , \ i \in I \ , && (7.21)\\
\sum_i x_i &= 1 \ , && (7.22)\\
\sum_k d_i^k &= 1 \ , \ i \in I \ , && (7.23)\\
x_i &\geq 0 \ , \ i \in I \ , && (7.24)\\
d_i^k &\in \{0,1\} \ , \ i \in I \ , \ k \in K(i) \ . && (7.25)
\end{aligned}$$

Unfortunately, this formulation is not linear (because of the products $x_i d_i^k$) and partly integer valued (due to the variables d_i^k). The nonlinearity is not really a problem, because it is possible to replace the product $x_i d_i^k$ by a new variable x_i^k. The interpretation of these new variables is clear: x_i^k is just the equilibrium probability that the system is in state i and that action k is taken. If we allow randomized stationary strategies as well, then we can also forget about the constraint that the d_i^k are 0 or 1. They only have to sum up to 1 in each state.
This brings us to the second problem formulation which now is a linear programming one.

LP

Maximize

$$\sum x_i^k r(i,k)$$

subject to

$$\begin{aligned}
\sum_k x_i^k &= \sum_{j,l} x_j^l p_{ji}^l \ , \ i \in I \ , && (7.26)\\
\sum_{i,k} x_i^k &= 1 \ , && (7.27)\\
x_i^k &\geq 0 \quad . && (7.28)
\end{aligned}$$

Clearly the first N equations are dependent, as the sum of them is just an identity. So we can eliminate one of them, the N-th one say. As a result we have only N constraints. But that implies that, any basic solution will have at most N nonzero variables. On the other hand, any stationary strategy, also any randomized stationary strategy, has only one recurrent chain and no transient states. From that one may show that a basic solution must have at least one nonzero x_i^k in each state i. Hence, there is a one to one correspondence between stationary strategies and basic solutions. So the the LP formulation is indeed equivalent to the Markov decision problem.

7.8 The relation between LP and Policy iteration

As in the discounted case, there is strong relation between the linear programming approach and policy iteration. As we have seen, it is possible to omit one of the first N constraints and we skipped constraint number N. For this LP problem we construct the LP matrix.

x_1^1	$x_1^{K_1}$	$\cdots$	x_{N-1}^1	$x_{N-1}^{K_{N-1}}$	x_N^1	$x_N^{K_N}$	v	
$1-p_{11}^1$	$1-p_{11}^{K_1}$	$\cdots$	$-p_{N-11}^1$	$-p_{N-11}^{K_{N-1}}$	$-p_{N1}^1$	$-p_{N1}^{K_N}$	0	0
$-p_{1N-1}^1$	$-p_{1N-1}^{K_1}$	$\cdots$	$1-p_{N-1N-1}^1$	$1-p_{N-1N-1}^{K_2}$	$-p_{NN-1}^1$	$-p_{NN-1}^{K_N}$	0	0
1	1	$\cdots$	1	1	1	1	0	1
$-r(1,1)$	$-r(1,K_1)$	$\cdots$	$-r(N-1,1)$	$-r(N-1,K_{N-1})$	$-r(N,1)$	$-r(N,K_N)$	1	0

Now consider a basic solution, which as we have seen corresponds to a stationary strategy. In order to see whether this strategy is optimal one has to find the linear combination of the constraints that, if added to the optimality equation, will give zeroes for the basic variables. So one has to find values d_1 up to d_{N-1} and g (the reason for the peculiar naming of these variables will become clear), such that, if the first $N-1$ constraints are multiplied by the corresponding d_i and the N-th by g, one gets these zeroes. So, we have to find the d_i and g that satisfy

$$\begin{array}{rcrcrcrcl}
-r(1,f(1)) & + & (1-p_{11}^{f(1)})d_1 & + \cdots + & (-p_{1N-1}^{f(1)})d_{N-1} & + & g & = & 0 \\
 & & & \cdots & & & & & \\
-r(N-1,f(N-1)) & + & (-p_{N-11}^{f(N-1)})d_1 & + \cdots + & (1-p_{N-1N-1}^{f(N-1)})d_{N-1} & + & g & = & 0 \\
-r(N,f(N)) & + & (-p_{N1}^{f(N)})d_1 & + \cdots + & (-p_{NN-1}^{f}(N))d_{N-1} & + & g & = & 0
\end{array}$$

In vector-matrix notation with an additional parameter d_N this reads,

$$r(f) + P(f)d = d + g.e \tag{7.29}$$

$$d_N = 0 \tag{7.30}$$

This is just the set of equations (7.15) and (7.17) derived before, with instead of the normalizing condition $P^*(f)d = 0$ the alternative $d_N = 0$. So the solution for $g.e$ is $g(f)$ and the solution for d is the relative value vector $d(f)$ plus a constant vector.

Now in order to verify whether the present basic solution is optimal one has to look at the other variables in the optimality equation. The present solution is optimal if

$$r(i,k) + \sum_j p_{ij}^k d_j \leq d_i + g$$

for all i and k. But this is just the same question as encountered for the optimality in the policy iteration algorithm.

So the linear programming approach is very similar to policy iteration. Normally in the simplex method we change one variable at a time. Whereas in the policy iteration approach we allow in each iteration a change of action in each state.

Remark 7.8.1 *Successive approximations and policy iteration are not capable of dealing with additional constraints. Linear programming can. In some problems constraints are natural. For instance, in inventory control one may want at least a certain fraction of the demand to be satisfied from stock.*

Chapter 8

Semi-Markov decision processes

8.1 Introduction

In quite a number of Markov decision processes the time points at which the system is controlled are not equidistant. The simplest example is the case that the underlying stochastic processes are not Markov chains but continuous time Markov processes.

Example 8.1.1 *Consider an $M|M|1$ queueing system with costs for waiting and rewards for accepting and completing a job. Possibly there are also costs for rejecting a job. It will be clear that if the queue is very long then it is optimal to reject a new arrival. In this model the time the system stays in a state is both random and action dependent.*

Example 8.1.2 *In an $M|M|s$ model there are at most s machines active. One may also decide to use temporarily less than s machines in operation. For instance for maintenance reasons. Again the time the system stays in a state is both random and action dependent.*

Even worse, the decision process need not be Markov process any more. It might only be a semi-Markov process. In a semi-Markov process the mean and distribution of the time the system spends in a state depend on that state and possibly on the action taken. The transition from that state to the next state is random. It may depend on the state, the action and the realization of the time needed for the transition. But it does not depend on the history of the process before the state was reached .

Example 8.1.3 *In a production system with Poisson arrivals and general service times one can decide to operate the machine at different speeds. The higher the speed the more damage to the machine. If the state of the machine is very bad, a maintenance job has to be inserted. The machine speed can be changed but not while a job is being processed. Actions are taken immediately after the completion of a job.*

8.2 The model

The semi-Markov decision process ($SMDP$) is characterized by the following elements:

- a discrete, usually finite, set of *states* denoted by I,
- in state i a finite set of *actions* denoted by $K(i)$,
- *transition probabilities* p_{ij}^k, by which, if in state i action k is taken, the system makes a transition to state j,
- the (expected) *transition time* τ_i^k needed to complete the transition if in state i action k is taken,
- the (expected) *immediate reward* $r(i,k)$ if in state i action k is taken.

Because of the random duration of the transitions, the discounted problem becomes more involved. It will not be discussed here. Below the average reward case is treated for which the situation is easier.

We will look at the problem in two different ways. For that we will consider the case of a fixed stationary strategy, f say. First we will have a look at the so-called *embedded* process, looking at transition moments only. If we merely count these transition epochs, then this process looks very much like a Markov chain. After that we will follow a different approach, by using a kind of time transformation.

8.3 The embedded chain

So, let us consider stationary strategy f used in the $SMDP$. Looking at transition instants only, thus at the points in time at which an action is taken, we get a Markov chain with transition probabilities $p_{ij}^{f(i)}$. The embedded equilibrium distribution $\pi(f, emb)$ can be computed from

$$\pi_i(f,emb) = \sum_j \pi_j(f,emb)\; p_{ji}^{f(j)}, \qquad \sum_i \pi_i(f,emb) = 1\ . \tag{8.1}$$

Then $g(f)$, the average reward per unit time for this strategy, can be seen to be equal to the mean reward per period divided by the mean time per period, thus

$$g(f) = \frac{\sum_i \pi_i(f,emb) r(i,f(i))}{\sum_i \pi_i(f,emb)\tau_i^{f(i)}}\ . \tag{8.2}$$

8.4 A time transformed Markov process

Before we consider the general $SMDP$ let us first have a look at a continuous time Markov chain. For a given stationary strategy f there are transition rates denoted by $q_{ij}^{f(i)}$ for $j \neq i$ (the transition rate from i to itself is 0). Then

$$p_{ij}^{f(i)} = q_{ij}^{f(i)} / \sum_{l \neq i} q_{il}^{f(i)}, \quad j \neq i\,, \quad p_{ii}^{f(i)} = 0\,.$$

The mean time $\tau_i^{f(i)}$ needed to make the transition satisfies

$$\tau_i^{f(i)} = 1/\sum_{l \neq i} q_{il}^{f(i)}.$$

Now we can construct an equivalent Markov process in which the times between the transition have the same mean τ^f, with

$$\tau^f = \min_i \tau_i^{f(i)}\ .$$

We do this by adding an artificial (or fictitious) transition rate $q_{ii}^{f(i)}$ from state i to itself so that

$$\sum_{j \neq i} q_{ij}^{f(i)} + q_{ii}^{f(i)} = 1/\tau^f\ .$$

These artificial rates lead to fake transitions (nothing changes) but the result is that the mean time between transitions (fake transitions included) now is state independent and equal to τ^f. We can now look at the corresponding embedded Markov chain with transition probabilities

$$p_{ij}^{f(i)}(MC) = \frac{q_{ij}^{f(i)}}{\sum_{l \neq i} q_{il}^{f(i)} + q_{ii}^{f(i)}} = \frac{q_{ij}^{f(i)}}{\sum_{l \neq i} q_{il}^{f(i)}} \frac{\tau^f}{\tau_i^{f(i)}} = \frac{\tau^f}{\tau_i^{f(i)}} p_{ij}^{f(i)},\ j \neq i \tag{8.3}$$

and

$$p_{ii}^{k}(MC) = 1 - \sum_{j \neq i} p_{ij}^{f(i)}(MC) = 1 - \sum_{j \neq i} \frac{\tau^f}{\tau_i^{f(i)}} p_{ij}^{f(i)} = 1 - \frac{\tau^f}{\tau_i^{f(i)}}\ . \tag{8.4}$$

Of course we also have to correct the immediate rewards. As at the new next decision epoch the real transition has been completed with a probability less than 1, to be precise, with probability $\tau^f/\tau_i^{f(i)}$, the reward will be received with that probability as well, so

$$r(i, f(i), MC) = \frac{\tau^f}{\tau_i^{f(i)}} r(i, f(i))\ .$$

Note that there still is a difference between the continuous time average reward and the average reward in the Markov chain due to the fact that in the Markov chain we talk about average reward per period and a period is not one time unit but τ^f time units.

In this Markov chain the equilibrium distribution is obtained form

$$\pi_i(f, MC) = \sum_j \pi_j(f, MC) p_{ji}^{f(j)}(MC) \ , \quad \sum_i \pi_i(f, MC) = 1 \ , \tag{8.5}$$

so that the average reward per period satisfies

$$g(f, MC) = \sum_i \pi_i(f, MC) r(i, f(i), MC) \ .$$

The average reward per unit time thus should satisfy

$$g(f) = g(f, MC) \ / \ \tau^f \ . \tag{8.6}$$

8.5 The equivalence

According to our intuition the two averages from equations (8.2) and (8.6) will be equal. With the use of equations (8.3) and (8.4), equation (8.5) can be rewritten as

$$\pi_i(f, MC) = \sum_{j \neq i} \pi_j(f, MC) \frac{\tau^f}{\tau_j^{f(j)}} p_{ji}^{f(j)} + \pi_i(f, MC)(1 - \frac{\tau^f}{\tau_i^{f(i)}})$$

or

$$\frac{\pi_i(f, MC)}{\tau_i^{f(i)}} = \sum_{j \neq i} \frac{\pi_j(f, MC)}{\tau_j^{f(j)}} p_{ji}^{f(j)} \ .$$

From this we see, by comparing with equation (8.1), that

$$\alpha \frac{\pi_i(f, MC)}{\tau_i^{f(i)}} = \pi_i(f, emb), \tag{8.7}$$

for some constant α. If we now substitute this expression into equation (8.2) for the average reward of the embedded process we get

$$\frac{\sum_i \pi_i(f, emb) r(i, f(i))}{\sum_i \pi_i(f, emb) \tau_i^{f(i)}} = \frac{\alpha \sum_i \pi_i(f, MC) \frac{r(i, f(i))}{\tau_i^{f(i)}}}{\alpha \sum_i \pi_i(f, MC)} = \sum_i \pi_i(f, MC) r(i, f(i), MC) / \tau^f \ ,$$

which is just the average reward per unit time for the MC case.

8.6 DP and different period length

Consider the following trivial $SMDP$ with only one state (called 1) and two decisions (called 1 and 2) with the following characteristics.

$$r(1,1) = 3, \quad p_{11}^1 = 1, \quad \tau_1^1 = 3$$

$$r(1,2) = 2, \quad p_{11}^2 = 1, \quad \tau_1^2 = 1 \ .$$

So clearly it is optimal to use action 2 as this action yields 2 per time unit where action 1 only gives 1 per time unit. The straightforward implementation of the DP equation might result in

$$v_{n+1}(1) = \max_k \{r(1,k) + v_n(1)\} = 3 + v_n(1) \ ,$$

which suggests that action 1 is optimal.

Apparently we need to correct for the fact that the two actions lead to different mean transition times.

8.7 The general SMDP

Having seen how we can transform the Markov process and the problems different period lengths may cause for the DP approach a logical step is to use the following data transformation for the general $SMDP$.

States and actions are the same in the two models. The difference is in the transitions and the rewards. First the minimal expected period length τ is defined by

$$\tau = \min_{i,k} \tau_i^k \ .$$

Next the transition probabilities for the MDP version of the $SMDP$ are constructed as follows,

$$p_{ij}^k(MDP) = \frac{\tau}{\tau_i^k} \, p_{ij}^k \ , \ for \ j \neq i \ , \tag{8.8}$$

$$p_{ii}^k(MDP) = 1 - \frac{\tau}{\tau_i^k} + \frac{\tau}{\tau_i^k} \, p_{ii}^k \ . \tag{8.9}$$

Note that this is a little different from what we have seen before because p_{ii}^k is allowed to be positive, i.e., we now don't exclude the possibility of transition from a state to itself (for instance a repair that failed or a demand of 0 in an inventory problem).

Finally the rewards are defined as

$$r(i,k,MDP) = \frac{r(i,k)}{\tau_i^k} \ .$$

As we have seen, with these definitions the average rewards in the two models should be equal. The ordering based on the average reward of the stationary strategies is exactly the same in the two models.

Without proof we state that also in the average reward $SMDP$ we can restrict ourselves to stationary strategies.

Conclusion. *We can solve the average reward $SMDP$ by solving the equivalent MDP by any of the three methods introduced before, successive approximations, policy iteration and linear programming.*

8.8 An example

Consider an $M|M|1$ queueing system with jobs arrive according to a Poisson process at rate λ and exponential service times with mean $1/\mu$. The only decision is to accept of reject new arrivals. There are two types of cost: the cost of waiting is c_w per job per time unit, and the cost of rejecting a job denoted by c_r. It is clear that the system will no longer accept jobs if the number of jobs in the system m satisfies $mc_w/\mu > c_r$. So the $M|M|1$ system with infinite state space can be reduced to an $M|M|1|N$ system with finite space, with $N = \min\{m|mc_w/\mu > c_r\}$. However, it is also clear that if λ is much larger than μ, the system does not have to fear idle time, i.e., loss of capacity, so that the maximal number of jobs in the system may be very small.

8.8.1 A solution

We first detail the problem formulation by specifying the

- **states**, the set of possible states is $\{0, 1, \ldots, N\}$,
- **decisions**, in the states $0, 1, ..., N-1$ we can either accept ($k = 1$) or reject ($k = 0$) new arrivals. In state N we can only reject.
- **transitions**, a possible transition structure is the following.

  $$p_{00}^0 = 1, \ p_{i,i-1}^0 = 1, \ i = 1, \ldots, N \ ,$$

  $$p_{01}^1 = 1, \ p_{i,i+1}^1 = \frac{\lambda}{\lambda+\mu}, \ p_{i,i-1}^1 = \frac{\mu}{\lambda+\mu}, \ i = 1, \ldots, N-1 \ ,$$

  and the

- **mean time per transition**

  $$\tau_0^0 = 1/\lambda,\ \tau_0^1 = 1/\lambda,\ \tau_i^0 = 1/\mu,\ \tau_i^1 = 1/(\lambda+\mu)\ .$$

  So, we get $\tau = 1/(\lambda+\mu)$.

Then the data transformation gives:

$$p_{00}^0(MDP) = 1,\ p_{i,i-1}^0(MDP) = \frac{\mu}{\lambda+\mu},\ \ p_{i,i}^0(MDP) = \frac{\lambda}{\lambda+\mu},\ i = 1,\ldots,N\ ,$$

$p_{00}^1(MDP) = \dfrac{\mu}{\lambda+\mu}$, $p_{i,i+1}^1(MDP) = \dfrac{\lambda}{\lambda+\mu}$, $i < N$, $p_{i,i-1}^1(MDP) = \dfrac{\mu}{\lambda+\mu}$, $i = 1,\ldots N{-}1$.

Etc.

Chapter 9

Structured Markov decision processes

9.1 Introduction

So far Markov decision processes have been studied without any additional structure. All that was used is that the underlying processes are (semi-)Markovian. In many problems there is additional structure and, as a result of that, the optimal strategy is of a special form.

9.2 Examples

In this section first a couple of examples is given.

Example 9.2.1 *Order acceptation*
Jobs arrive according to a Poisson process. Upon arrival it has to be decided whether the job is accepted or rejected. The rejection costs are c per rejected job. For all jobs in the system waiting costs are paid. These costs are 1 per job per time unit.

Example 9.2.2 *Inventory model*
Once a week the inventory of a certain product can be adjusted by a replenishment order. Replenishment is free and takes time 0. The demand per week is random. There are two types of costs, inventory costs and shortage costs, both linear. The optimal replenishment strategy is an 'order up to' policy.

Example 9.2.3 *Inventory model 2*
The problem is the same as in the previous example except for the fact that there are fixed replenishment costs. The optimal strategy is of (s, S) type.

Example 9.2.4 *Maintenance model*
Consider a machine which produces one type of items. A quick weekly inspection shows the state of the machine. The machine does not get better. The costs of operation increase if the machine state decreases. One has the option to replace the machine by a new one. The optimal strategy is expected to be of the following type: if the state of the machine drops below a certain level it is replaced.

Example 9.2.5 *Car insurance*
Car insurances are characterized by premium and by the so-called 'bonus-malus' system. The premium per year is expressed in a percentage of the basic premium. If one does not claim this year, the percentage to pay next year is lowered. If one claims the percentage goes up. The optimal strategy will be of the following type. Given the present percentage there is limit for the damage, such that if the damage is below that level claiming is not optimal, above that limit it is optimal to claim.

In all these examples the additional structure of the problem leads to an optimal strategy with additional structure. One advantage of a structured strategy is that it is easy to apply it in practice. (If decisions are not intuitively appealing people will use their personal judgement to 'improve' the strategy.) Another advantage is that computations may become simpler, hence faster.

9.3 Proof of structured optimal strategies

There is no general proof for the existence of an optimal strategy with a *threshold* or *control limit* structure. One has to look at the specific structure of the problem. The line of proof, however, is intuitive, and hence one can try to copy it or adapt it.

In this section the optimality of a structured strategy is established for the order acceptance model of Example 9.2.1.

First, the model is transformed into an ordinary MDP. Let the arrival rate be λ and the service rate be μ. The state is fully characterized by the number of jobs in the system. In each state there are two decisions: accept or reject the next job. Decisions are taken after each state change. Following Chapter 8 we define $\tau = 1/(\lambda + \mu)$. Further, define $p = \lambda\tau$ and $q = \mu\tau$. So, if in state i it is decided to accept the next job, the MDP makes a transition to state $i + 1$ with probability p and a transition to state $i - 1$ with probability q. If the decision is to reject the next job the transition is to state i with probability p or to $i - 1$ with probability q.

We will prove that the optimal strategy is of the form: accept jobs if the number of jobs in the system is at most $M - 1$ and reject jobs otherwise, for some M. It is obvious that we will not accept a job if it is clear that its expected waiting costs will exceed c. (These

costs are equal to the expected waiting time because the cost rate for waiting is equal to 1.)

In order to prove this, we will use successive approximations and we will show that in each step the optimal policy is of the control limit type.

In step $n+1$ we have the following optimization.

$$\begin{aligned} v_{n+1}(i) &= \min\{i + pv_n(i+1) + qv_n(i-1)\ ,\ i + p(c + v_n(i)) + qv_n(i-1)\} \\ &= i + qv_n(i-1) + p\min\{v_n(i+1)\ ,\ c + v_n(i)\}\ . \end{aligned}$$

In order to prove the control limit result it has to be shown that if it is optimal to reject in state i, then it is optimal to reject in state $i+1$ as well.

Lemma 9.3.1 *If v_n is convex, then there is an optimal policy f_{n+1} of control limit type.*

Proof: Suppose it is optimal to reject in state i. Then $c + v_n(i) \geq v_n(i+1)$, or $v_n(i+1) - v_n(i) \geq c$. From the convexity we have $v_n(i+2) - v_n(i+1) \geq v_n(i+1) - v_n(i)$, so it is optimal to reject in state $i+1$ as well.

Define $v_0 = 0$, then v_0 is convex, and there is an optimal control limit policy in the 1-step problem with terminal reward v_0. In order to construct an induction proof, it has to be shown that v_1 is convex. Below the result is formulated for general n.

Lemma 9.3.2 *If v_n is convex and non-decreasing, then v_{n+1} is convex and non-decreasing as well.*

Proof: Assuming that v_n is convex it has been have shown already that there is an optimal control limit for period $n+1$. Let i_{n+1} be the control limit, i.e., it is optimal to accept in $i < i_{n+1}$ and to reject otherwise. Then

$$v_{n+1}(i) = i + pv_n(i+1) + qv_n(i-1) \quad i < i_{n+1}\ .$$

So v_{n+1} is non-decreasing and convex for $0 < i < i_{n+1}$. Similarly, we see that v_{n+1} is convex for $i > i_{n+1}$. So it remains to show the convexity in 0 and in i_{n+1}.
For state 0 the fact is used that v_n is non-decreasing to show that $v_{n+1}(2) - v_{n+1}(1) \geq v_{n+1}(1) - v_{n+1}(0)$. This is left to the reader. In state i_{n+1} two inequalities have to be verified. The first one is

$$v_{n+1}(i_{n+1}+1) - v_{n+1}(i_{n+1}) \geq v_{n+1}(i_{n+1}) - v_{n+1}(i_{n+1}-1)\ .$$

The left hand side (lhs) is equal to

$$1 + q(v_n(i_{n+1}) - v_n(i_{n+1}-1)) + p(\ v_n(i_{n+1}+1) - v_n(i_{n+1})\ )\ .$$

The right hand side (rhs) is

$$1 + q(v_n(i_{n+1} - 1) - v_n(i_{n+1} - 2) + pc \ .$$

The first two terms in the lhs are clearly at least equal to these terms on the rhs. The inequality for the third term follows from the fact that rejection is optimal in state i_{n+1}. The other inequality we have to establish is the one with i_{n+1} replaced by $i_{n+1} - 1$. The verification process is almost identical.

Lemmas 9.3.1 and 9.3.2 together complete the proof that the optimal strategy is of threshold type for all finite n. One may show (by letting n tend to ∞) that this implies that also in the average reward problem the optimal strategy is of this simple form.

The above proof was not so difficult. In most problems, however, the intuitively obvious monotonicity results are very hard to prove. One of the well known elegant but more involved proofs is the one by Scarf for the optimality of (s, S) strategies using K-convexity.