International Journal of Pure and Applied Mathematics
Volume 51 No. 2 2009, 277-291

ANALYSIS OF MARKOV CHAIN WITH REWARDS BY Z-TRANSFORM AND THE DECISION OF ITS OPTIMAL POLICY

Yoshinori Uchimura[1], Yuko Hara-Mimachi[2] §

[1]Graduate School of Science and Technology
Meijo University
1-501, Shiogamaguchi, Tempaku-ku, Nagoya, 468-5802, JAPAN
e-mail: m0732008@ccmailg.meijo-u.ac.jp

[2]Department of Information Sciences
Meijo University
1-501, Shiogamaguchi, Tempaku-ku, Nagoya, 468-5802, JAPAN
e-mail: yuko@ccmfs.meijo-u.ac.jp

Abstract: This paper describes the analysis of Markov chain with rewards by z-transform. The decision method of its optimal policy is proposed by Ronald A. Howard. We have implemented Howard's Policy-Improvement algorithm by a high level programming language *Scilab* and added a counter of the iteration to this. It will be usually found this policy in a small number of iterations. On the other hand, we can obtain example of 6 time iterations by this program.

AMS Subject Classification: 60J22, 65C40, 65K05
Key Words: Markov chain, ergodic Markov chain, z-transform, dynamic programming, optimal policy

1. The z-Transform

A z-transform is applied to derive the sequence from a known difference equation or recurrence formula. Since the z-transform and the Discrete Fourier Transform (DFT) are closely related to each other, the z-transform is also applied to Digital Signal Processing (DSP). To analyze the Markov chain, we give some definitions and some theorems of z-transform in this section.

Received: February 5, 2009

§Correspondence author

Definition 1. (Two Side z-Transform) A two side z-transform of a sequence $\{x(n)\}$ is defined by

$$X_{\text{II}}(z) = \mathcal{Z}_{\text{II}}[x(n)] = \sum_{n=-\infty}^{\infty} x(n)z^{-n}, \tag{1}$$

where z is a continuous complex variable, and $\mathcal{Z}_{\text{II}}[\cdot]$ is a operator of the two side z-transform.

The two side z-transform is also called the bilateral z-transform.

Definition 2. (One Side z-Transform) An one side z-transform of a sequence $\{x(n)\}$ is defined by

$$X(z) = \mathcal{Z}[x(n)] = \sum_{n=0}^{\infty} x(n)z^{-n}, \tag{2}$$

where z is a continuous complex variable, and $\mathcal{Z}[\cdot]$ is the operator of the one side z-transform.

It is also called the unilateral z-transform. Clearly, the two side and the one side z-transforms are equivalent when $x(n) = 0$ for $n < 0$. In this paper, we apply only the one side z-transform. Therefore, the following theorems are concerned with the one side z-transform.

The z-transform is a linear transformation. Thus, the following theorem holds (see [2]).

Theorem 3. (Linearity) *If the z-transforms of two sequences $\mathcal{Z}[x_1(n)]$ and $\mathcal{Z}[x_2(n)]$ are denoted by $X_1(z)$ and $X_2(z)$, respectively, then*

$$\mathcal{Z}[\alpha x_1(n) + \beta x_2(n)] = \alpha X_1(z) + \beta X_2(z),$$

where α and β are constants.

Similarly, regarding the one side z-transform, the following theorem holds (see [2]).

Theorem 4. (Translation) *The z-transform of a delayed sequence $x(n-m)$ is*

$$\mathcal{Z}[x(n-m)] = z^{-m}X(z),$$

where m is a natural number.

For the analysis of Markov chain, we prepare the z-transform regarding the sequence of vector. We show below a definition of the z-transform regarding a vector.

Definition 5. (One Side z-Transform of a Vector) The one side z-transform of an N-vector $\boldsymbol{x}$ is defined by

$$\boldsymbol{X}(z) = \mathcal{Z}[\boldsymbol{x}(n)] = \begin{pmatrix} \mathcal{Z}[x_1(n)] \\ \mathcal{Z}[x_2(n)] \\ \vdots \\ \mathcal{Z}[x_N(n)] \end{pmatrix}. \tag{3}$$

It is clear that if we define the z-transform of a vector by (3), a linearity and a translation properties hold.

2. Analysis of Markov Chain by z-Transform

Consider an N-state discrete-time ergodic Markov chain whose state space is $S = \{1, 2, \cdots, N\}$. We also assume that this Markov chain is time-homogeneous. Let p_{ij} $(i, j \in S)$ be *a trasition probability of going from state i to state j*, where it does not depend on time n. Note that

$$0 \leq p_{ij} \leq 1,$$

and for any i $(1 \leq i \leq N)$

$$\sum_{j=1}^{N} p_{ij} = 1.$$

A transition matrix $\boldsymbol{P}$ is defined by

$$\boldsymbol{P} = [p_{ij}] = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{pmatrix}.$$

We denote *the stationary distribution of the chain* after n transitions by $\boldsymbol{\pi}(n)$. $\boldsymbol{\pi}(n)$ is a row vector that satisfies the equation

$$\boldsymbol{\pi}(n+1) = \boldsymbol{\pi}(n)\boldsymbol{P} \quad (n = 0, 1, \cdots). \tag{4}$$

Since the Markov chain is a time-homogeneous, we have

$$\boldsymbol{\pi}(n) = \boldsymbol{\pi}(0)\boldsymbol{P}^n, \tag{5}$$

if the initial stationary distribution $\boldsymbol{\pi}(0)$ is known.

When the vector z-transform of the vector $\boldsymbol{\pi}(n)$ is given by the symbol $\boldsymbol{\Pi}(z)$, the transform of (4) is

$$\mathcal{Z}[\boldsymbol{\pi}(n+1)] \quad = \quad \mathcal{Z}[\boldsymbol{\pi}(n)\boldsymbol{P}],$$

$$z(\boldsymbol{\Pi}(z) - \boldsymbol{\pi}(0)) = \boldsymbol{\Pi}(z)\boldsymbol{P}.$$

Through rearrangement we have

$$\begin{aligned} \boldsymbol{\Pi}(z) - \boldsymbol{\pi}(0) &= z^{-1}\boldsymbol{\Pi}(z)\boldsymbol{P}, \\ \boldsymbol{\Pi}(z)(\boldsymbol{I} - z^{-1}\boldsymbol{P}) &= \boldsymbol{\pi}(0), \end{aligned}$$

and finally

$$\boldsymbol{\Pi}(z) = \boldsymbol{\pi}(0)(\boldsymbol{I} - z^{-1}\boldsymbol{P})^{-1}, \tag{6}$$

where the matrix $\boldsymbol{I}$ is identity. On the other hand, the z-transform of (5) is

$$\begin{aligned} \mathcal{Z}[\boldsymbol{\pi}(n)] &= \mathcal{Z}[\boldsymbol{\pi}(0)\boldsymbol{P}^n], \\ \boldsymbol{\Pi}(z) &= \boldsymbol{\pi}(0)\mathcal{Z}[\boldsymbol{P}^n]. \end{aligned}$$

Thus a substitution of $\boldsymbol{\pi}(0)\mathcal{Z}[\boldsymbol{P}^n]$ into the left side of (6) gives

$$\mathcal{Z}[\boldsymbol{P}^n] = (\boldsymbol{I} - z^{-1}\boldsymbol{P})^{-1}.$$

On the other hand, if *a limit stationary distribution* is represented by $\bar{\boldsymbol{\pi}}$, $\boldsymbol{P}^n$ is written by

$$\boldsymbol{P}^n = \boldsymbol{u}\bar{\boldsymbol{\pi}} + \boldsymbol{\Phi}(n),$$

where

$$\boldsymbol{u} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$$

and $\boldsymbol{\Phi}(n)$ is a $N \times N$ matrix, which converges exponentially to zero matrix $\boldsymbol{O}$. Thus we obtain

$$\mathcal{Z}[\boldsymbol{P}^n] = (\boldsymbol{I} - z^{-1}\boldsymbol{P})^{-1} = \frac{1}{1 - z^{-1}}\boldsymbol{u}\bar{\boldsymbol{\pi}} + \mathcal{Z}[\boldsymbol{\Phi}(n)]. \tag{7}$$

Consider an N-state Markov chain with rewards. Let r_{ij} be the reward associated with a transition from state i to state j. We denote the set of rewards for the chain by a reward matrix $\boldsymbol{R}$ with element r_{ij}. Thus, we get

$$\boldsymbol{R} = [r_{ij}] = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1N} \\ r_{21} & r_{22} & \cdots & r_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ r_{N1} & r_{N2} & \cdots & r_{NN} \end{pmatrix}.$$

Let us denote *the expected total earnings* in the next n transitions by $v_i(n)$

if a state i is occupied. We define

$$v_i(n) = \sum_{j=1}^{N} p_{ij}\{r_{ij} + v_j(n-1)\} \quad (i = 1, 2, \cdots, N)$$

that is,

$$v_i(n) = \sum_{j=1}^{N} p_{ij} r_{ij} + \sum_{j=1}^{N} p_{ij} v_j(n-1) \quad (i = 1, 2, \cdots, N).$$

By putting

$$w_i = \sum_{j=1}^{N} p_{ij} r_{ij} \quad (i = 1, 2, \cdots, N),$$

then we have

$$v_i(n) = w_i + \sum_{j=1}^{N} p_{ij} v_j(n-1) \quad (i = 1, 2, \cdots, N). \tag{8}$$

w_i is called *the expected immediate reward for state i*. It can be interpreted as the reward to be expected in the next transition out of state i. A column vector $\boldsymbol{v}(n)$ with N components $v_i(n)$ is called *a total-value vector* when

$$\boldsymbol{v}(n) = \boldsymbol{w} + \boldsymbol{P}\boldsymbol{v}(n-1), ,$$

where

$$\boldsymbol{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}.$$

Let us denote the z-transform of the totla-value vector $\boldsymbol{v}(n)$ by $\boldsymbol{V}(z)$. From (2), we get

$$\boldsymbol{v}(n+1) = \boldsymbol{w} + \boldsymbol{P}\boldsymbol{v}(n) \quad (n = 0, 1, \cdots).$$

If we take the z-transform of this equation, we obtain

$$\mathcal{Z}[\boldsymbol{v}(n+1)] = \mathcal{Z}[\boldsymbol{w} + \boldsymbol{P}\boldsymbol{v}(n)]$$

and

$$z(\boldsymbol{V}(z) - \boldsymbol{v}(0)) = \frac{1}{1 - z^{-1}}\boldsymbol{w} + \boldsymbol{P}\boldsymbol{V}(z).$$

Through rearrangement, we have

$$\boldsymbol{V}(z) - \boldsymbol{v}(0) = \frac{z^{-1}}{1 - z^{-1}}\boldsymbol{w} + z^{-1}\boldsymbol{P}\boldsymbol{V}(z),$$

$$\begin{aligned} \boldsymbol{V}(z) - z^{-1}\boldsymbol{P}\boldsymbol{V}(z) &= \frac{z^{-1}}{1-z^{-1}}\boldsymbol{w} + \boldsymbol{v}(0), \\ (\boldsymbol{I} - z^{-1}\boldsymbol{P})\,\boldsymbol{V}(z) &= \frac{z^{-1}}{1-z^{-1}}\boldsymbol{w} + \boldsymbol{v}(0), \end{aligned}$$

and finally

$$\boldsymbol{V}(z) = \frac{z^{-1}}{1-z^{-1}}(\boldsymbol{I} - z^{-1}\boldsymbol{P})^{-1}\boldsymbol{w} + (\boldsymbol{I} - z^{-1}\boldsymbol{P})^{-1}\boldsymbol{v}(0). \tag{9}$$

The substitution of $\boldsymbol{v}(0) = \boldsymbol{o}$ will reduce (9) to

$$\boldsymbol{V}(z) = \frac{z^{-1}}{1-z^{-1}}(\boldsymbol{I} - z^{-1}\boldsymbol{P})^{-1}\boldsymbol{w}, \tag{10}$$

where $\boldsymbol{o}$ is zero vector. By substituting of the right-hand side of (7) into (10), we give

$$\boldsymbol{V}(z) = \frac{z^{-1}}{(1-z^{-1})^2}\boldsymbol{u}\bar{\boldsymbol{\pi}}\boldsymbol{w} + \frac{z^{-1}}{1-z^{-1}}\mathcal{Z}[\boldsymbol{\Phi}(n)]\boldsymbol{w}.$$

When $\boldsymbol{\phi}(n)$ is defined by

$$\boldsymbol{\phi}(n) = \boldsymbol{\Phi}(n)\boldsymbol{w},$$

we obtain

$$\boldsymbol{V}(z) = \frac{z^{-1}}{(1-z^{-1})^2}\boldsymbol{u}\bar{\boldsymbol{\pi}}\boldsymbol{w} + \frac{z^{-1}}{1-z^{-1}}\mathcal{Z}[\boldsymbol{\phi}(n)].$$

By using Theorem 4, we have

$$\frac{z^{-1}}{1-z^{-1}}\mathcal{Z}[\boldsymbol{\phi}(n)] = \frac{1}{1-z^{-1}}\mathcal{Z}[\boldsymbol{\phi}(n-1)].$$

Since $\boldsymbol{\phi}(n-1)$ also converges exponentially to zero vector $\boldsymbol{o}$, we have

$$\frac{z^{-1}}{1-z^{-1}}\mathcal{Z}[\boldsymbol{\phi}(n)] = \frac{1}{1-z^{-1}}\mathcal{Z}[\boldsymbol{\phi}(n)].$$

Therefore, we get

$$\boldsymbol{V}(z) = \frac{z^{-1}}{(1-z^{-1})^2}\boldsymbol{u}\bar{\boldsymbol{\pi}}\boldsymbol{w} + \frac{1}{1-z^{-1}}\mathcal{Z}[\boldsymbol{\phi}(n)].$$

Since $\lim_{n\to\infty}\boldsymbol{\phi}(n) = 0$, a summation of $\boldsymbol{\phi}(n)$ can be expressed by

$$\sum_{k=0}^{n}\boldsymbol{\phi}(k) = \boldsymbol{c} + \boldsymbol{\phi}(n),$$

where

$$\boldsymbol{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{pmatrix}.$$

By taking the z-transform of this equation, we obtain

$$\begin{aligned} \mathcal{Z}\left[\sum_{k=0}^{n} \boldsymbol{\phi}(k)\right] &= \mathcal{Z}[\boldsymbol{c} + \boldsymbol{\phi}(n)], \\ \frac{1}{1-z^{-1}}\mathcal{Z}[\boldsymbol{\phi}(n)] &= \frac{1}{1-z^{-1}}\boldsymbol{c} + \mathcal{Z}[\boldsymbol{\phi}(n)]. \end{aligned}$$

Thus, $\boldsymbol{V}(z)$ can be written by

$$\boldsymbol{V}(z) = \frac{z^{-1}}{(1-z^{-1})^2}\boldsymbol{u}\bar{\boldsymbol{\pi}}\boldsymbol{w} + \frac{\boldsymbol{c}}{1-z^{-1}} + \mathcal{Z}[\boldsymbol{\phi}(n)].$$

If we take the inverse z-transform of this equation, we obtain

$$\boldsymbol{v}(n) = \boldsymbol{u}\bar{\boldsymbol{\pi}}\boldsymbol{w}n + \boldsymbol{c} + \boldsymbol{\phi}(n)$$

and

$$v_i(n) = (\bar{\boldsymbol{\pi}}\boldsymbol{w})n + c_i + \phi(n). \tag{11}$$

From this result, if n is large enought, then $v_i(n)$ can approximate to $(\bar{\boldsymbol{\pi}}\boldsymbol{w})n + c_i$.

We have implemented a simulation program using Monte Carlo method. By this program, a total earning by the simulation is provided. Also, a total expected earning is calculated by a transition probability matrix and a reward matrix. A transition probability matrix and a reward matrix can be decided freely. Figure 1 shows an example of the simulation result.

3. The Decision Method of Optimal Policy

In this section, we deal with the decision method of optimal policy proposed by Ronald A. Howard. By this method, we can find the optimal policy from some policy with respect to the Markov chain with rewards.

Let g be defind by

$$g = \bar{\boldsymbol{\pi}}\boldsymbol{w},$$

where $\bar{\boldsymbol{\pi}}$ and $\boldsymbol{w}$ are the limiting state probability vector and the expected immediate reward vector, respectively. Then, g means a slope of the asymptote of the total-value. g is criterion to decision of the optimal policy. From (8), we

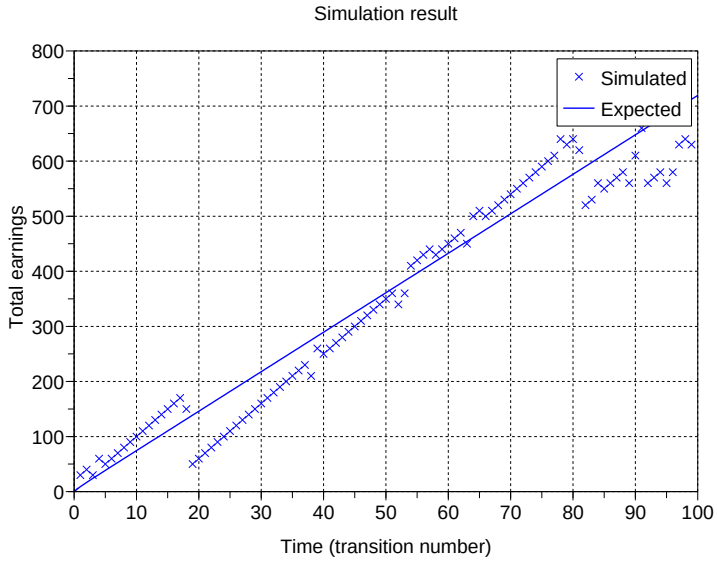

Figure 1: An example of the simulation result

have

$$v_i(n+1) = w_i + \sum_{j=1}^{N} p_{ij} v_j(n) \quad (n = 0, 1, \cdots). \tag{12}$$

On the other hand, it was shown in before section that for ergodic Markov chain, the value $v_i(n)$ is approximated as

$$v_i(n) = (\bar{\boldsymbol{\pi}}\boldsymbol{w})n + c_i \quad \text{for large } n. \tag{13}$$

Therefore, substituting (13) into (12) we get

$$\begin{aligned} g \cdot (n+1) + c_i &= w_i + \sum_{j=1}^{N} p_{ij}(gn + c_j) && (14) \\ &= w_i + \sum_{j=1}^{N} p_{ij} gn + \sum_{j=1}^{N} p_{ij} c_j + \sum_{j=1}^{N} p_{ij} && (15) \\ &= w_i + gn \sum_{j=1}^{N} p_{ij} + \sum_{j=1}^{N} p_{ij} c_j. \end{aligned}$$

Since $\sum_{j=1}^{N} p_{ij} = 1$, for any $1 \leq i \leq N$, these equations become

$$g + c_i = w_i + \sum_{j=1}^{N} p_{ij} c_j.$$

Therefore, the following simultaneous equation holds.

$$g\boldsymbol{u} + \boldsymbol{c} = \boldsymbol{w} + \boldsymbol{P}\boldsymbol{c}, \tag{16}$$

where g and $\boldsymbol{c}$ are unknown values. To complete the solution we set

$$c_N = 0,$$

because it suffices just to find the value g.

Let A and B be policies. We use superscripts A and B to indicate the quantities relevant to policies A and B. As noted previously, these policies are compared by g^A and g^B. If

$$g^A \geq g^B,$$

it is decided that policy A is superior to policy B. Here, we discuss an important theorem regarding (17).

Theorem 6. *If we have*

$$\boldsymbol{w^A} + \boldsymbol{P^A c^B} \geq \boldsymbol{w^B} + \boldsymbol{P^B c^B}, \tag{17}$$

it follows that

$$g^A \geq g^B.$$

Proof. From the (16) we obtain the following equations regarding A and B.

$$g^A\boldsymbol{u} + \boldsymbol{c^A} = \boldsymbol{w^A} + \boldsymbol{P^A c^A}, \tag{18}$$

$$g^B\boldsymbol{u} + \boldsymbol{c^B} = \boldsymbol{w^B} + \boldsymbol{P^B c^B}. \tag{19}$$

Subtracting (19) from (18), we obtain

$$g^\Delta\boldsymbol{u} + \boldsymbol{c^\Delta} = \boldsymbol{w^B} - \boldsymbol{w^A} + \boldsymbol{P^B c^B} - \boldsymbol{P^A c^A}, \tag{20}$$

where $g^\Delta = g^B - g^A$ and $c^\Delta = c^B - c^A$. Put

$$\boldsymbol{d} = \boldsymbol{w^B} - \boldsymbol{w^A} + \boldsymbol{P^B c^A} - \boldsymbol{P^A c^A}. \tag{21}$$

Note that $\boldsymbol{d} \geq 0$. From (20), (21) , we have

$$g^\Delta\boldsymbol{u} + \boldsymbol{c^\Delta} = \boldsymbol{d} + \boldsymbol{P^B c^B} - \boldsymbol{P^B c^A}. \tag{22}$$

If we multiply both side of (22) by $\bar{\boldsymbol{\pi}}^{\boldsymbol{B}}$, we obtain

$$g^\Delta\bar{\boldsymbol{\pi}}^{\boldsymbol{B}}\boldsymbol{u} + \bar{\boldsymbol{\pi}}^{\boldsymbol{B}}\boldsymbol{c^\Delta} = \bar{\boldsymbol{\pi}}^{\boldsymbol{B}}\boldsymbol{d} + \bar{\boldsymbol{\pi}}^{\boldsymbol{B}}\boldsymbol{P^B c^\Delta},$$

where $\bar{\boldsymbol{\pi}}^{\boldsymbol{B}}$ is limit stationary distribution of policy B. From $\bar{\boldsymbol{\pi}}^{\boldsymbol{B}}\boldsymbol{u} = 1$ and

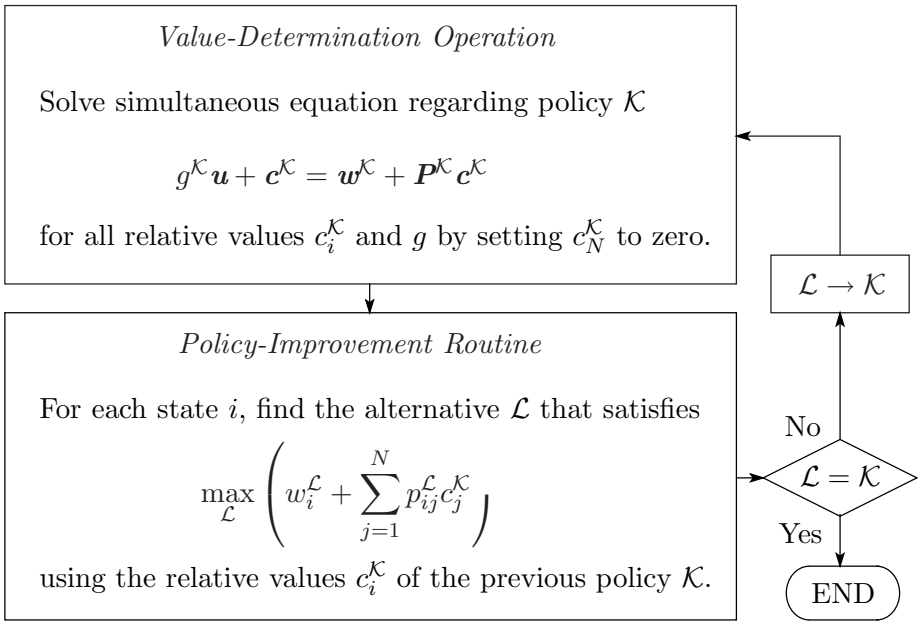

Figure 2: Howard's policy-improvement algorithm

$\bar{\boldsymbol{\pi}}^{B}\boldsymbol{P}^{B} = \bar{\boldsymbol{\pi}}^{B}$ we have

$$g^{\Delta} = \bar{\boldsymbol{\pi}}^{B}\boldsymbol{d}.$$

Thus, note $\boldsymbol{d} \geq 0$ we obtain

$$g^{B} - g^{A} = g^{\Delta} = \bar{\boldsymbol{\pi}}^{B}\boldsymbol{d} \geq 0. \qquad \square$$

Figure 2 shows Howard's policy-improvement algorithm. We may enter the iteration cycle in either box. If the value-determination operation is chosen as the entrance point, an initial policy must be selected. If the cycle is to start in the policy-improvement routine, then a starting set of values is necessary. The upper box, the value-determination operation, yields the g and $\boldsymbol{c}$ of policy $\mathcal{K}$ corresponding to given matrices $\boldsymbol{P}^{\mathcal{K}}$ and $\boldsymbol{R}^{\mathcal{K}}$. The lower box finds the alternative $\mathcal{L}$ that maximizes

$$w_i^{\mathcal{L}} + \sum_{j=1}^{N} p_{ij}^{\mathcal{L}} c_j^{\mathcal{K}}$$

using the relative value $c_i^{\mathcal{K}}$. In other wards, the value-determination operation yields values as a function of policy, whereas the policy-improvement routine yields the policy as a function of the values.

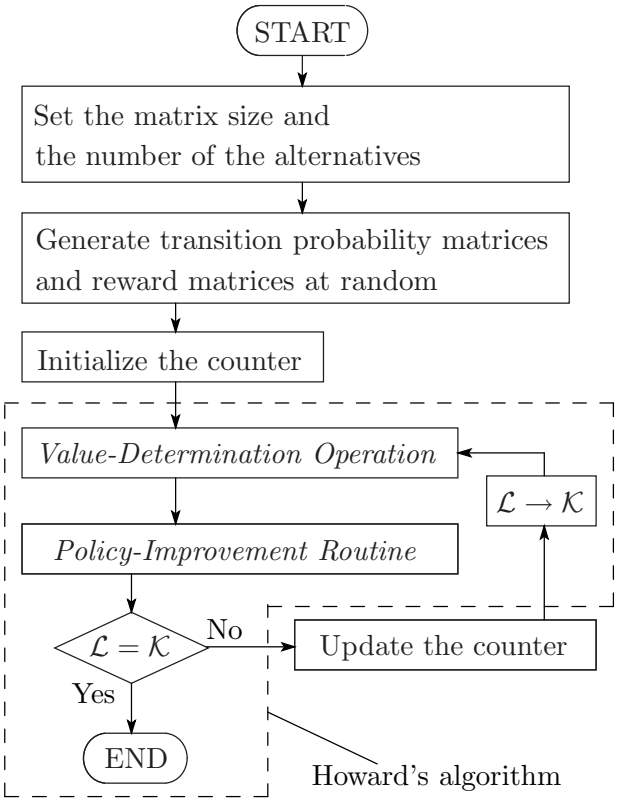

Figure 3: A flowchart of the implemented program

We have implemented Howard's policy-improvement algorithm by a high level programming language *Scilab* and added a counter of the iteration to this. *Scilab* is available for download at no cost (see [5]). Figure 3 shows a flowchart of the implemented program and Appendix shows its *Scilab* script. It will usually find this policy in a small number of iterations. On the other hand, we can obtain example of 6 time iterations by this program.

References

[1] Ronald A. Howard, *Dynamic Programming and Markov Processes*, The Massachusetts Institute of Technology Press (1960).

[2] Eliahu Ibrahim Jury, *Theory and Application of the z-Transform Method*, Krieger Pub. Co. (1973).

[3] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck, *Discrete-Time Signal Processing*, Second Edition, Prentice Hall International (1999).

[4] Hiroshi Yanai, *Z Henkan to Sono Ouyou*, Nikkagiren Shuppan (1999), In Japanese.

[5] *Scilab*, http://www.scilab.org/

Appendix: *Scilab* Script

```
clear

//-------------------------------------------------------------

function[Pi_list] = Pi_pattern_list(msize,dp,maxnum)

[lhs,rhs] = argn(0);
if rhs <= 2, maxnum = 1; end
if rhs <= 1, dp = 0.1; end
for i = maxnum:-dp:0,
  if msize-1 == 0,
    b = maxnum;
    break;
  else
    Pi_list = Pi_pattern_list(msize-1,dp,maxnum-i);
    Pi_list = [i*ones(size(Pi_list,1),1),Pi_list];
  end
  if i == maxnum,
    b = Pi_list;
  else
    b = [b;Pi_list];
  end
end
Pi_list = b;
endfunction

//-------------------------------------------------------------

function[Ri_list] = Ri_pattern_list(msize,dr,minnum,maxnum)
```

```
[lhs,rhs] = argn(0);
if rhs <= 3, maxnum = 1; end
  if rhs <= 2, minnum = 0; end
if rhs <= 1, dr = 0.25; end
Rij_pattern = minnum:dr:maxnum;
Rsy = length(Rij_pattern)^(msize);
Ri_list = zeros(Rsy,msize);
for i = 1:msize,
  for j = 1:length(Rij_pattern),
    if j==1,
      rj = Rij_pattern(j)*ones(Rsy/(length(Rij_pattern)^i),1);
    else rj=[rj;Rij_pattern(j)*ones(Rsy/(length(Rij_pattern)^i),1)]
    end
  end
  Rj=rj;
  for k=1:(Rsy/length(rj))-1,
      Rj=[Rj;rj];
  end
  Ri_list(:,i)=Rj;
end
endfunction

//---------------------------------------------------------------

//===========================================
// Value-Determination Operation
//===========================================
function[w,g,c]=value_determination(P,R)
[sy,sx] = size(P);
w = sum(P.*R,2);
A = [ones(sy,1),eye(sy,sx-1)-P(1:sy,1:sx-1)];
[c,kerA] = linsolve(A,w);
g = -c(1);
c = -[c(2:$);0];
endfunction

//---------------------------------------------------------------

msize = 5;  // matrix size
kmax = 500; // number of the alternatives
dp = 0.05;
dr = 0.25;
maxr = 1;
minr = -1;
```

```
a = round(Pi_pattern_list(msize,dp)*100)/100;
b = Ri_pattern_list(msize,dr,minr,maxr);
sa = size(a,1); sb = size(b,1);

for i = 1:kmax,
  eigen = 1;
  while eigen > 0,
    for j=1:msize,
      execstr(['P_l.No'+string(i)+'(j,:)'+'=a(ceil(sa*rand(1)),:)']);
      execstr(['R_l.No'+string(i)+'(j,:)'+'=b(ceil(sb*rand(1)),:)']);
    end
    execstr(['eigen=spec(P_l.No'+string(i)+')']);
    eigen = max(abs(imag(eigen(abs(eigen)==1))));
  end
end

k = kmax*ones(1,msize);
k_now = ones(1,msize);
count = 1;

while 1,
  P=zeros(msize,msize);
  R=zeros(msize,msize);

  for j=1:msize,
    execstr(['P_k=P_l.No'+string(k(j))']);
    execstr(['R_k=R_l.No'+string(k(j))']);
    P(j,:) = P_k(j,:);
    R(j,:) = R_k(j,:);
  end

  [w,g,c] = value_determination(P,R);
  S = w+P*c;
  disp('k = '); disp(k)
  disp('c = '); disp(c')
  disp('g = '); disp(g)

  //----- Policy-Improvement Routine -----
  k_now = ones(1,msize); // initialization
  for i = 1:kmax
    execstr(['Pk=P_l.No'+string(i)']);
    execstr(['Rk=R_l.No'+string(i)']);
    w = value_determination(Pk,Rk);
    S_now = w+Pk*c;
```

```
    k_now(S_now >= S & sum(Pk,2)==1) = i;
    S(S_now >= S & sum(Pk,2)==1) = S_now(S_now >= S & sum(Pk,2)==1);
  end
  if sum(abs(k-k_now)) == 0; break;
  else k = k_now;
  end
  //-------------------------------------
  count = count+1;
end

for i = 1:length(k),
  execstr(['Pk=P_1.No'+string(k(i))']);
  execstr(['Rk=R_1.No'+string(k(i))']);
  P(i,:) = Pk(i,:);
  R(i,:) = Rk(i,:);
end

disp('-------- OPTIMAL POLICY --------')
disp('k = '); disp(k); // optimal policy
disp('P=');   disp(P);
disp('R=');   disp(R);
disp('count = '+string(count))
disp('--------------------------------')
```