AN APPLICATION OF THE LUMPING METHOD TO A LOSS SYSTEM WITH TWO TYPES OF CUSTOMERS

Tsukasa Sato
Nissan Diesel Co. Ltd.

Masao Mori
Tokyo Institute of Technology

(Received May 27, 1982; Revised November 29, 1982)

Abstract This paper applies the lumping method, proposed by Takahashi[4], to get stationary probabilities numerically for a loss system with two types of input streams. By using the method we can evaluate numerically the effect of introducing several servers who are capable to serve both types of customers.

1. Introduction

In a telephone service system for ticket reservation in an airline company, some reception clerks serve only for domestic line passengers and others serve only for international line passengers. In this manner of service there often happens such an unbalanced situation that all clerks for domestic lines are hard at work, while some servers for international lines are idle, or vice-versa. So it is expected that we could improve service availability by introducing several well-trained servers who are capable to deal with any service requirements for both types of customers. In this paper we would examine how these commonly usable servers could decrease the total loss probability.

In the section 2, we will introduce a model to analyze above phenomena, which is described as a Markov process with a large state space. Such the largeness of state space makes it almost impossible even for a powerful computer to get stationary probabilities for this model by using ordinary Gauss-Seidel method or Gauss elimination method. So we provide an abbreviated algorithm of lumping method, originally proposed in Takahashi[4] and Takahashi & Takami[5], to get stationary probabilities in the section 3. In the section 4, we will examine some numerical examples.
2. Model

We consider a loss system with $S$ servers, into which two types of customers are arriving. Each input stream is a Poisson process with rate $\lambda_k$ ($k=1,2$). $S_k$ servers can only deal with the type $k$ customers ($k=1,2$), and the rests $S_{\bar{s}}=S-S_1-S_2$, called commonly usable servers, are highly educated to be able to serve both types of customers. And we assume that commonly usable servers deal with customers overflowed from their own assignmented $S_1$ or $S_2$ servers. A type $k$ customer who finds each $S_k$ and $S_{\bar{s}}$ servers are all busy should be lost.

Here each service time is assumed to be independently and exponentially distributed. $S_k$ servers serve with rate $\mu_k$ ($k=1,2$). Commonly usable $S_{\bar{s}}$ servers may be somewhat less efficient on account of complex job, so they are assumed to server with rate $\mu_{\bar{s}}$ for type 1 customers and with rate $\mu_4$ for type 2 customers.

By setting above, we can easily formulate the model as a Markov process having an appropriate infinitesimal generator $Q$ on the state space $\mathcal{S}=(i_1, i_2, i_3, i_4)$; $0 \leq i_1 \leq S_1$, $0 \leq i_2 \leq S_2$, $0 \leq i_3 \leq S_{\bar{s}}$, $i_3, i_4 \geq 0$, where $i_k$ denotes the number of servers serving among $S_k$ servers ($k=1,2$) and $i_3$ and $i_4$ denote the number of servers among commonly usable $S_{\bar{s}}$ servers serving type 1 and type 2 customers respectively. Note that the total number of states is $N=(S_1+1)(S_2+1)(S_{\bar{s}}+1)(S_{\bar{s}}+2)/2$, which is fairly large. For example, $N=29,106$ in the case of $S_1=S_2=20$ and $S_{\bar{s}}=10$.

Clearly the process is regular, that is, irreducible and aperiodic. Thus the stationary distribution exists and is given by the unique stochastic vector $\pi$ satisfying $Q\pi=\pi$, where $0$ is a $N$-dimensional zero vector. However the largeness of state space makes it difficult to get $\pi$ numerically in an ordinaly manner. In the next section we will contrive to calculate $\pi$ by using lumping method.

3. Application of the Lumping Method

Let us assign a number to each state of the Markov process in the lexicographical order, i.e. $1=(0,0,0,0)$, $2=(0,0,0,1)$, $\ldots$, $S_1+1=(0,0,0,S_1)$, $S_2+2=(0,0,1,0)$, $\ldots$, $N-1=(S_1+S_2,S_2-1,1)$ and $N=(S_1+S_2,S_2,0)$. And let us partition the state space $\mathcal{S}$ into groups (or blocks) indexed by $(i_1, i_2)$ as follows:

$A_1=\{(i_1,0,0,i_4): 0 \leq i_3+i_4 \leq S_3, i_3, i_4 \geq 0\} \equiv \{1, 2, \ldots, m\}$

$A_2=\{(i_1,0,i_3,i_4): 0 \leq i_3+i_4 \leq S_3, i_3, i_4 \geq 0\} \equiv \{m+1, m+2, \ldots, 2m\}$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
An Application of the Lumping Method

\[ A_p = \{(S_1, S_2, i_1, i_2) : 0 \leq i_1 \leq S_1, \quad 0 \leq i_2 \leq S_2, \quad i_1, i_2 \geq 0\} \ni \{N-m+1, N-m+2, \ldots, N\} \]

where \( r = (S_1+1)(S_2+1) \) is the number of groups (or blocks). We call each \( A_i \) a lumped state. And group size (or block size) of each lumped state is \( m = (S_1+1)(S_2+2)/2 \).

In the manner of above numbering of the state space, the infinitesimal generator \( Q \) is written as

\[
Q = \begin{pmatrix}
00 & 01 & \cdots & 0S_2 & 10 & \cdots & 1S_2 & \cdots & 0S_2 + 1S_2 & \cdots & SS_2
\end{pmatrix}
\]

where each entry designated by blank is zero and \( Q_{ii} \) for \( i = (i_1, i_2) \) is represented by

\[
Q_{ii} = \begin{cases}
T-(i_1 u_1 + i_2 u_2)I & \text{for } 0 \leq i_1 \leq S_1 - 1 \text{ and } 0 \leq i_2 \leq S_2 - 1 \\
T + \Lambda_1 -(S_1 u_1 + i_2 u_2)I & \text{for } 0 \leq i_2 \leq S_2 - 1 \text{ and } i_1 = S_1 \\
T + \Lambda_2 -(i_1 u_1 + S_2 u_2)I & \text{for } 0 \leq i_1 \leq S_1 - 1 \text{ and } i_2 = S_2 \\
T + \Lambda_1 + \Lambda_2 -(i_1 u_1 + S_2 u_2)I & \text{for } i_1 = S_1 \text{ and } i_2 = S_2
\end{cases}
\]

Here \( Q_{ii} \), \( T \), \( \Lambda_1 \) and \( \Lambda_2 \) are \( m \times m \) submatrices representing transition rate within each group, \( I \) is the \( m \times m \) identical matrix. \( T \), \( \Lambda_1 \) and \( \Lambda_2 \) are just defined for computational convenience. For instance, in the case of \( S_2 = 3 \), \( T \), \( \Lambda_1 \) and \( \Lambda_2 \) are given by the following matrices:

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
An Application of the Lumping Method

\[
\begin{align*}
T &= \begin{pmatrix}
(0,0) & (0,1) & (0,2) & (0,3) & (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (3,0) \\
(0,0) & -\lambda & & & & & & & & \\
(0,1) & \mu & -\lambda & & & & & & & \\
(0,2) & 2\mu & -\lambda & 2& & & & & & \\
(0,3) & 3\mu & -\lambda & 3& & & & & & \\
(1,0) & \mu & & -\lambda & & & & & & \\
(1,1) & \mu & & \mu & -\lambda & & & & & \\
(1,2) & \mu & & \mu & 2& -\lambda & & & & \\
(2,0) & 2\mu & & & & & & & & \\
(2,1) & 2\mu & & \mu & -\lambda & & & & & \\
(3,0) & 3\mu & & \mu & & & & & & \\
\end{pmatrix} \\
\Lambda_1 &= \begin{pmatrix}
(0,0) & (0,1) & (0,2) & (0,3) & (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (3,0) \\
0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\
\lambda_1 & 0 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & 0 \\
\lambda_1 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\
\lambda_1 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\
\lambda_1 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix} \\
\Lambda_2 &= \begin{pmatrix}
(0,0) & (0,1) & (0,2) & (0,3) & (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (3,0) \\
0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\end{align*}
\]

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.
An Application of the Lumping Method

Here we write $Q$ concisely as

\[(3.2) \quad Q = (Q_{ij})\]

where $Q_{ij}$ is an $m \times m$ submatrix representing transition rate from the group $A_i$ to the group $A_j$. And also we split the stationary distribution vector $\varphi$ into

\[(3.3) \quad \varphi = (\varphi_1, \varphi_2, \ldots, \varphi_m)\]

where each $\varphi_i$ is a $m$-dimensional row vector corresponding to $A_i$.

Let $\pi$ be the $m$-dimensional column vector with all entries equal to 1, and $\pi = (\pi_i)$ be the $r$-dimensional row vector whose $i$-th entry is $\pi_i = X_i \cdot e$. The entry $\pi_i$ denotes the probability that the process is in the group $A_i$ in the steady state. Let $\beta_i = \frac{1}{\pi_i} X_i \equiv (\beta_{ik} | k \in A_i)$. Then $\beta_i \cdot e = 1$ and the entry $\beta_{ik}$ denotes the conditional probability that the process is in the state $k$ in the steady state given that the process is in the group $A_i$.

In the general lumping method both the vector $\pi$ and $\beta_i$ should be calculated iteratively. And the total number of iterations increases rapidly as the number of groups $r$ and the group size $m$ increase.

However in our model the marginal distribution $\pi = (\pi_i)$ is given by

\[(3.4) \quad \pi_i = \left( \frac{1}{i!} \sum_{j=0}^{\infty} \lambda_i^j \frac{1}{j!} \sum_{j=0}^{\infty} \lambda_j^j \right) \cdot \left( \sum_{j=0}^{\infty} \frac{1}{j!} \left( \frac{\lambda_2}{\mu_2} \right)^j \right) \cdot \left( \sum_{j=0}^{\infty} \frac{1}{j!} \left( \frac{\lambda_1}{\mu_1} \right)^j \right)
\]

for the lumped state $A_i = \{ (i_1, i_2, i_3, i_4) | 0 \leq i_3 + i_4 \leq S_2, i_3, i_4 \geq 0 \}$, which is easily obtained by considering the independent behaviour of the solely usable $S_1$ and $S_2$ servers for type 1 and type 2 customers respectively. So we can omit the procedure to calculate the vector $\pi$. Thus the abbreviated algorithm is given in the following. Here we will designate the conditional probability vectors at the $n$th iteration as $\beta_i(n)$.

Algorithm

\begin{itemize}
  \item \textbf{Step 1} \quad Start with arbitrary initial stochastic vectors $\beta_i(0)$, $i = 1, 2, \ldots, r$.
  \item \textbf{Step 2} \quad At the $n$th iteration, solve the equation
    \[(3.5) \quad \beta_i(n)Q_{ij} + \sum_{d=0}^{S_1} \omega_{id} \beta_i(n)Q_{ij} + \sum_{d=S_2}^{r} \omega_{j+1} \beta_i(n-1)Q_{ij} = 0_m\]
    for $\beta_i(n)$, where $0_m$ is the $m$-dimensional zero vector.
    
    Normalize $\beta_i(n)$ as
    \[(3.6) \quad \beta_i(n) = \beta_i(n) / (\beta_i(n) \cdot e)\]
  \item \textbf{Step 3} \quad For a given sufficiently small $\varepsilon > 0$, if
    \[(3.7) \quad |\beta_{ik}(n-1) - \beta_{ik}(n)| < \varepsilon \quad \text{for all } i, k\]
  \item \textbf{Step 4} \quad Set the stationary distribution as $X = (\omega_i \beta_i(n))$.
\end{itemize}
In this algorithm the total computational time depends on how to solve the equation (3.5) effectively. Ordinarily we solve it by Gauss-Seidel iteration method. Fortunately, however, many of diagonal matrices $Q_{ii}$ are triangular, so in such a case we can directly solve the equation by Gauss elimination method. By this little idea we can reduce computational time much.

4. Numerical Examples

In our problem, we are interested mainly in the values of the following loss probabilities:

- $P_1$: probability a type 1 customer is lost,
- $P_2$: probability a type 2 customer is lost,
- $P_t = \frac{\lambda_1 P_1 + \lambda_2 P_2}{\lambda_1 + \lambda_2}$: probability an arbitrary customer is lost.

We call $P_t$ the total loss rate of the system.

Table 1 shows some values of loss probabilities for the case of $S=10$.

For a given set of parameters $\lambda_i$ and $\mu_i$, the optimal assignment of servers to minimize the total loss rate $P_t$ without introducing commonly usable servers is $(S_1=6, S_2=4)$. Each loss probabilities in this case are $(P_1=0.05216, P_2=0.08524, P_t=0.06939)$. Then we assume 3 servers to be trained as commonly usable servers, but their service speed is assumed to be not so high as that of other servers dealing with only one type of jobs: here we set $\mu_3$ and $\mu_4$ as 90% of $\mu_1$ and $\mu_2$ respectively. The optimal assignment for the case turns to be $(S_1=4, S_2=3, S_3=3)$, where loss probabilities are $(P_1=0.03898, P_2=0.03432, P_t=0.03712)$. It is noteworthy that these values are all smaller than corresponding probabilities for the case of $(S_1=6, S_2=4, S_3=0)$. That is to say, we can improve system availability/performance both for customers and the manager of the system by training a few as commonly usable servers.

Here we mention computational times. We used HITAC 8350, which was installed about ten years ago. In getting probabilities we set $\varepsilon=10^{-6}$. For the case of $(S=10; S_1=5, S_2=2, S_3=3; \text{no. of blocks}=(S_1+1)(S_2+1)=18, \text{block size}=(S_3+1)(S_3+2)/2=10)$, CPU time is 72 seconds. For the case of $(S=30; S_1=17, S_2=10, S_3=3; \text{no. of blocks}=(17+1)(10+1)=198, \text{block size}=(3+1)(3+2)/3=10)$, CPU time swells up to 59 minutes on account of increasing number of iterations in our algorithm. However, when our algorithm was checked by using ACOS 1000 at Tohoku University, it was reported that CPU times for $S=10$ are negligible small and the latter case takes 80 seconds. In practice we could set $\varepsilon$ as $10^3 \sim 10^5$ according to the order of numerical value of probabilities actually
An Application of the Lumping Method

Table 1. Loss probabilities

\[ S_1 + S_2 + S_3 = 10, \]
\[ \lambda_1 = 3.0, \lambda_2 = 2.0, \]
\[ \mu_1 = 1.0, \mu_2 = 1.0, \mu_3 = 0.9, \mu_4 = 0.9 \]

<table>
<thead>
<tr>
<th>( S_1 )</th>
<th>( S_2 )</th>
<th>( S_3 )</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( P_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0.00270</td>
<td>0.16667</td>
<td>0.26829</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0.00557</td>
<td>0.41740</td>
<td>0.17030</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>0.00976</td>
<td>0.23754</td>
<td>0.10087</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>0.01453</td>
<td>0.12722</td>
<td>0.05961</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>0.01887</td>
<td>0.07094</td>
<td>0.03970</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
<td>0.02238</td>
<td>0.04615</td>
<td>0.03188</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0.02532</td>
<td>0.03560</td>
<td>0.02943</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7</td>
<td>0.02809</td>
<td>0.03068</td>
<td>0.02913</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>8</td>
<td>0.03089</td>
<td>0.02811</td>
<td>0.02977</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>9</td>
<td>0.03382</td>
<td>0.02667</td>
<td>0.03069</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0.00813</td>
<td>0.40000</td>
<td>0.16488</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>1</td>
<td>0.01324</td>
<td>0.22383</td>
<td>0.09748</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>2</td>
<td>0.01878</td>
<td>0.11724</td>
<td>0.05816</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0.02364</td>
<td>0.06372</td>
<td>0.03968</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>0.02750</td>
<td>0.04055</td>
<td>0.03272</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>5</td>
<td>0.03076</td>
<td>0.03078</td>
<td>0.03077</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>0.03385</td>
<td>0.02625</td>
<td>0.03081</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>7</td>
<td>0.03701</td>
<td>0.02387</td>
<td>0.03175</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>8</td>
<td>0.04033</td>
<td>0.02252</td>
<td>0.03321</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0.02186</td>
<td>0.21053</td>
<td>0.09323</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1</td>
<td>0.02870</td>
<td>0.10695</td>
<td>0.06000</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>0.03446</td>
<td>0.05596</td>
<td>0.04306</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>3</td>
<td>0.03898</td>
<td>0.03432</td>
<td>0.03712</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0.04285</td>
<td>0.02539</td>
<td>0.03587</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>0.04662</td>
<td>0.02129</td>
<td>0.03649</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>0.05054</td>
<td>0.01913</td>
<td>0.03798</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>7</td>
<td>0.05470</td>
<td>0.01789</td>
<td>0.03998</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>0</td>
<td>0.05216</td>
<td>0.09524</td>
<td>0.06939</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>0.05925</td>
<td>0.04688</td>
<td>0.05430</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>0.06484</td>
<td>0.03703</td>
<td>0.04972</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3</td>
<td>0.06982</td>
<td>0.01913</td>
<td>0.04954</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>4</td>
<td>0.07485</td>
<td>0.01562</td>
<td>0.05116</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>5</td>
<td>0.08022</td>
<td>0.01379</td>
<td>0.05365</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>6</td>
<td>0.08598</td>
<td>0.01273</td>
<td>0.05668</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0</td>
<td>0.10005</td>
<td>0.03670</td>
<td>0.08071</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>0.11697</td>
<td>0.01914</td>
<td>0.07784</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>0.12352</td>
<td>0.01260</td>
<td>0.07915</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
<td>0.13048</td>
<td>0.00986</td>
<td>0.08223</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>0.13808</td>
<td>0.00849</td>
<td>0.08624</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>5</td>
<td>0.14635</td>
<td>0.00770</td>
<td>0.09089</td>
</tr>
</tbody>
</table>
required. Then the number of iterations calculating $\beta_2$ (Step 2→Step 4) might decrease, which leads to reduce the computational time much. So the lumping method would be applicable to a problem with proper size of state space.

5. Concluding Remarks

The lumping method enables us to calculate stationary probabilities of a model even with a fairly large state space. Especially it is very efficient for a model of which transition rate matrix is block tridiagonal([4]). However, in solving a Markov model with a general form of infinitesimal generator, usually this method requires a great number of iterations and takes much computational times, even though it makes possible to get probabilities. In our model, fortunately, the marginal probability vector $\pi$ is available, which makes it handy to apply the method.

In another paper [3], we tried to partition the state space into groups indexed by $i_1$ only. So the number of groups is $S_1+1$ and each group size is $(S_2+1)(S_3+1)(S_3+2)/2$. Grouping this way, the generator $Q$ is represented as a block tridiagonal matrix. This generator is similar to one of Ph/M loss system, but those are slightly different. So we cannot apply the method of Neuts[2] directly. We transformed $Q$ into a block didiagonal matrix, which enable us easily to solve that system of linear equations. However, largeness of block size requires large memory of computer.

Machihara[1] studies a similar problem appeared in telephone exchange network. He concentrates to analyze the phenomena happened at commonly usable servers, into which two types of interruped Poisson streams arrive. The approach is reported very efficient to get distributions.

Acknowledgements

The authors sincerely express their thanks to Professor Y. Takahashi of Tohoku University for his suggesting the problem and troublesome testing of our computer program using ACOS 1000. And they are also grateful to the referees for their kind comments.
References


Masao MORI: Dept. of Management Science, Tokyo Institute of Technology, Oh-okayama, Meguro-ku Tokyo 152, Japan.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.