Control of In-process Inventory in a Serial Manufacturing System with Inspection and Rework

Solon Venâncio de Carvalho
Rita de Cássia Meneses Rodrigues
Nandamudi Lankalapalli Vijaykumar
Paulo Renato de Morais

Laboratório Associado de Computação e Matemática Aplicada (LAC)
Instituto Nacional de Pesquisas Espaciais (INPE)
Av. dos Astronautas, 1758 - Caixa Postal 515
12201-970 - São José dos Campos - SP, Brazil
e-mail: {solon, rita, vijay, prenato}@lac.inpe.br

ABSTRACT

A manufacturing system consisting of two machines $M_1$ and $M_2$ operating in series and an in-process inventory is considered. Products have to pass an inspection after being processed by the machines. After inspection, products are either discarded, reworked, or are allowed to proceed to the next stage. The machines are subject to failure during use. The time-to-failure, the repair time, the processing time of the machines, and the rework time of each product are considered to be exponentially distributed. The in-process inventory is controlled by a policy that decides dynamically whether to activate machine $M_1$ or not. The cost structure includes a storage cost, a machine restart cost, a machine $M_2$ starving cost, a processing cost, a repair cost and a reward for producing the products. A Markov Decision Model is used to maximize the long-run average revenue per unit time and to obtain measures of system performance. The model is represented using the Statecharts tool. Numerical results are presented.

RESUMO

Considera-se um sistema de manufatura com duas máquinas $M_1$ e $M_2$ operando em série e um estoque intermediário. Os produtos processados pelas máquinas passam por uma inspeção, onde podem ser aceitos, descartados ou enviados para reprocessamento. As máquinas estão sujeitas a falhas durante o uso. O tempo até a quebra, o tempo de reparo, o tempo de processamento das máquinas e o tempo para reprocessar cada produto são considerados exponencialmente distribuídos. O estoque intermediário é controlado por uma política que decide dinamicamente sobre o bloqueio ou não da máquina $M_1$. A estrutura de custos inclui um custo de estocagem, um custo de reativação das máquinas, um custo de ociosidade da máquina $M_2$, um custo de processamento, um custo de reparo e um ganho por produto fabricado. Utiliza-se um Modelo Markoviano de Decisão para obter uma política que maximiza a receita média do sistema a longo prazo por unidade de tempo. Algumas medidas de desempenho são obtidas. O modelo é representado usando a ferramenta “Statecharts”. Resultados numéricos são apresentados.
1. Introduction

Consider a manufacturing system with two machines $M_1$ and $M_2$ continuously producing a single product. Each product is processed first by machine $M_1$ and then by machine $M_2$. An in-process inventory is used to reduce machine $M_2$ idleness. The storage capacity $N$ of the in-process inventory is finite. After being processed by the machines, each product must be inspected. After inspection, a product either (a) is discarded if it presents major defects (with probability $p_{s1}$ for $M_1$ and $p_{s2}$ for $M_2$), (b) is sent back to the machine for rework if it presents minor defects (with probability $p_{r1}$ for $M_1$ and $p_{r2}$ for $M_2$), or (c) it proceeds to the next stage in case no defects are found (with probability $p_{g1}$ for $M_1$ and $p_{g2}$ for $M_2$). If the machine involved is $M_1$, the next stage would be storing the product in the in-process inventory or loading the product directly into the second machine $M_2$ (depending on the availability of $M_2$). In case the machine involved is $M_2$, the next stage would be sending the finished product to its final destination (see Figure 1).

![Figure 1 - Overview of the system](image)

As the machines may fail during use, the main objective is to devise a policy of optimal control of the in-process inventory. This control is achieved by blocking or unblocking machine $M_1$. The cost structure includes a storage cost, a machine restart cost, a machine $M_2$ starving cost, a processing cost, a repair cost and a reward for producing the products. A Continuous Time Markov Decision Model is used to maximize the long-run average revenue per unit time and to obtain measures of system performance. The control policies considered to block or unblock machine $M_1$ take into account not only the size of the in-process inventory but also the observed state for each machine (blocked, starved, processing, broken, reworking a product, or broken during rework). This paper extends some ideas of [Hwang and Koh, 1992] and [Gopalan and Kannan, 1994], using the mathematical approach presented in [Carvalho et al, 1993].

[Hwang and Koh, 1992] formulated a Markovian model to obtain a (s, S) policy that minimizes the average cost of in-process inventory between two machines without considering rework or inspection. [Carvalho et al, 1993] extended this results, considering control policies that take into account not only the size of the in-process inventory but also the state of each machine (blocked, starved, processing, or broken). This idea yielded a minimum average cost that is less than the one obtained in [Hwang and Koh, 1992]. [Gopalan and Kannan, 1994] discussed a similar model including inspection and rework but without considering the in-process inventory.

2. Model description

The set of possible states for each machine $M_1$ and $M_2$ are:

$E_{M1} = \{W, P, R, B, BR\}$ and $E_{M2} = \{W, P, R, B, BR\}$,
Proceedings of *International Conference on Industrial Logistics (ICIL)*, Ouro Preto, MG, Brazil, December 1995

where for machines $M_1$ and $M_2$, $P$ denotes “processing a product”, $R$ denotes “reworking a product”, $B$ denotes “broken”, and $BR$ denotes “broken during rework”. For machine $M_1$ the state $W$ denotes “waiting to be unblocked”, while for machine $M_2$ the state $W$ denotes “waiting”. The machine $M_2$ has to wait whenever the in-process inventory is empty.

The decision to block or unblock machine $M_1$ takes into account the state of the system. The state space of the system is defined as:

$$E = \{(m_1, n, m_2) / m_1 \in E_{M1}, n \in \{0, 1, ..., N\}, m_2 \in E_{M2}\},$$

where $m_i$ is the state of machine $M_i$, $i = 1, 2$, and $n$ is the size of the in-process inventory (including the product that eventually is being processed on machine $M_2$).

The dynamic behavior of the system is described by the change of its states. Each time the system changes its state, the new state configuration must be observed in order to decide what action is to be taken, i.e. whether the machine $M_1$ is to be blocked or not. $M_1$ must be unblocked whenever $M_2$ is in the $W$ state ($n = 0$), and must be blocked whenever the in-process inventory is at its maximum capacity ($n = N$). Therefore, for each state $i = (m_1, n, m_2) \in E$, the space of possible actions is:

$$A(i) = \begin{cases} 
\{D\} & \text{if } n = 0 \\
\{D, B\} & \text{if } 0 < n < N \\
\{B\} & \text{if } n = N 
\end{cases}$$

where $D$ and $B$ denote respectively unblock and block $M_1$.

For each machine $M_i$, $i = \{1, 2\}$, the processing time of each product, the time-to-failure, the repair time, the rework time, the time-to-failure during rework and the repair time for failures during rework are independent, exponentially distributed random variables with rates $\beta_i$, $\lambda_i$, $\mu_i$, $\beta_{ri}$, $\lambda_{ri}$, and $\mu_{ri}$ respectively.

Products have to be inspected after being processed by the machines. After inspection, products are either discarded with probability $p_{si}$, reworked with probability $p_{ri}$, or are allowed to proceed to the next stage with probability $p_{gi}$, for $i = 1, 2$.

The Statecharts representation of this model is shown in Figure 2. This tool has been developed to specify complex reactive systems by extending the conventional state-transition diagrams to include the concepts of hierarchy, orthogonality, and communication. The definition, syntax, and semantics of the Statecharts are given in [Harel, 1987], [Harel et al, 1987], and [Hooman et al, 1992]. In the components $M_1$ and $M_2$ shown in Figure 2, inspections are modeled by a probabilistic selector represented by the letter p in a circle. It is important to clarify that the association of probabilities with the transitions is not a part of Statecharts formalism. This tool has been used to represent the system behavior in a clear manner. Some ideas of associating Statecharts with Markov chains to calculate performance measures are shown in [Vijaykumar et al, 1995].
In order to obtain a control policy that maximizes the long-run average revenue per unit time, this system is modeled as a Continuous Time Markov Decision Process. Given that at a decision epoch the system is in state $i \in E$ and action $a \in A(i)$ is chosen, denote by $\tau(i,a)$ the expected time until the next decision epoch, $p(i,j,a)$ the probability that in the next decision epoch the state will be $j \in E$, and $R(i,a)$ the expected revenue obtained until the next decision epoch.

$\Lambda_{i,j}(a)$ is defined as the transition rate from state $e_i$ to state $e_j$ ($e_i, e_j \in E$) when the last chosen action was $a \in A(i)$. The algorithm used to obtain transition rates $\Lambda_{i,j}(a)$ that describe the system behavior is as follows:

for each state $e_i = (m_1, p, m_2) \in E$ and for each action $a \in A(i)$ do

\[ e_r \leftarrow e_i \]
if ($e_i, m_1 = P \land a = B$) then $e_r, m_1 \leftarrow W$
if ($e_i, m_1 = W \land a = B$) then $e_r, m_1 \leftarrow P$

// machine $M_1$ is processing
if ($e_r, m_1 = P$) then {
    // may finish processing the product
    // with probability $p_{g1}$ product defectless
    $e_f \leftarrow e_r$
    $e_j, p \leftarrow e_j, p + 1$
    if ($e_j, p = N$) then $e_j, m_1 \leftarrow W$ // maximum capacity
    if ($e_j, m_2 = W$) then $e_j, m_2 \leftarrow P$
    create transition ($e_j, a, e_j$) with rate ($p_{g1} \beta_1$)
    // with probability $p_{g1}$ product fails in inspection
    $e_j \leftarrow e_r$
    create transition ($e_j, a, e_j$) with rate ($p_{g1} \beta_1$)
    // with probability $p_{g1}$ product to be reprocessed

\[ e_j \leftarrow e_i \]
$e_j, m_1 \leftarrow R$
create transition ($e_i, a, e_j$) with rate ($p_{g1} \beta_1$)
// in case of breakdown
$e_r \leftarrow e_i$
$e_j, m_1 \leftarrow B$
create transition ($e_j, a, e_j$) with rate $\lambda_1$
}

// machine $M_1$ reworking on the product
if ($e_r, m_1 = R$) then {
    // end of rework
    $e_j \leftarrow e_i$
    $e_j, m_1 \leftarrow P$
    $e_j, p \leftarrow e_j, p + 1$
    if ($e_j, p = N$) then $e_j, m_1 \leftarrow W$ // maximum capacity
    if ($e_j, m_2 = W$) then $e_j, m_2 \leftarrow P$
    create transition ($e_j, a, e_j$) with rate $\beta_1$
    // breakdown during rework

$\Lambda_{i,j}(a)$ is the rate of transition from state $e_i$ to state $e_j$ when the last chosen action was $a \in A(i)$.

Figure 2 - Statecharts representation of the model

The statecharts are used to model the decision process in the system. Each state represents a possible state of the system, and the transitions between states represent the possible actions that can be taken. The rate $\Lambda_{i,j}(a)$ is the rate at which a transition from state $e_i$ to state $e_j$ occurs when the last chosen action was $a \in A(i)$. The statecharts include a control mechanism that allows the system to be controlled based on the current state and the chosen action.
Using the transition rates $\lambda_{ij}(a)$, it is easy to obtain the total rate of output from each state given by $\Lambda_i(a) = \sum_{j \neq i} \lambda_{ij}(a)$, and so the transition probabilities are given by $p(i,j,a) = \lambda_{ij}(a)/\Lambda_i(a)$, and the expected time between transitions is given by $\tau(i,a) = 1/\Lambda_i(a)$. The expected revenue is given by:

$$R(i,a) = G(i,a) - C_b(i,a) - C_{p1}(i,a) - C_{p2}(i,a) - C_{b1}(i,a) - C_{b2}(i,a) - C_{r1}(i,a) - C_{r2}(i,a) - C_{l1}(i,a) - C_{l2}(i,a) - C_s(i,a),$$

where $G(i,a), C_b(i,a), C_{p1}(i,a), C_{p2}(i,a), C_{b1}(i,a), C_{b2}(i,a), C_{r1}(i,a), C_{r2}(i,a), C_{l1}(i,a), C_{l2}(i,a), C_s(i,a)$ represent respectively the expected reward for producing the products, the expected storage cost (cost to keep a product in the inventory), the expected processing cost for machine $M_1$, the expected processing cost for machine $M_2$, the expected repair cost for machine $M_2$, the expected restart cost for machine $M_1$, the expected restart cost for
machine M₂, the expected cost of loosing a product by machine M₁, the expected cost of loosing a product by machine M₂, and the expected starving cost for machine M₂, incurred until the next decision epoch, given that at the decision epoch the system is in state i ∈ E and action a ∈ A(i) is chosen. The terms given to describe R(i,a) are given by:

\[
C_b(i,a) = \begin{cases} 
  c_b(n-1)\tau(i,a) & \text{if } n > 1 \\
  0 & \text{otherwise}
\end{cases}
\]

\[
C_{p1}(i,a) = \begin{cases} 
  c_{p1}\tau(i,a) & \text{if } m_1 = P \text{ or } m_1 = R \\
  0 & \text{otherwise}
\end{cases}
\]

\[
C_{p2}(i,a) = \begin{cases} 
  c_{p2}\tau(i,a) & \text{if } m_2 = P \text{ or } m_2 = R \\
  0 & \text{otherwise}
\end{cases}
\]

\[
C_{b1}(i,a) = \begin{cases} 
  c_{b1}\tau(i,a) & \text{if } m_1 = B \text{ or } m_1 = BR \\
  0 & \text{otherwise}
\end{cases}
\]

\[
C_{b2}(i,a) = \begin{cases} 
  c_{b2}\tau(i,a) & \text{if } m_2 = B \text{ or } m_2 = BR \\
  0 & \text{otherwise}
\end{cases}
\]

\[
C_{s}\frac{\sum_{e_j \in E \atop e_j \neq W} \Lambda_j(a)}{\Lambda_i(a)} = \begin{cases} 
  c_s & \text{if } e_jm_k = W \\
  0 & \text{otherwise}
\end{cases} 
\]

\[
C_{il}(i,a) = \begin{cases} 
  c_{il}p_{s1}\beta_1 & \text{if } m_1 = P \\
  0 & \text{otherwise}
\end{cases}
\]

\[
C_{l2}(i,a) = \begin{cases} 
  c_{l2}p_{s2}\beta_2 & \text{if } m_2 = P \\
  0 & \text{otherwise}
\end{cases}
\]

\[
C_s(i,a) = \begin{cases} 
  c_s \tau(i,a) & \text{if } n = 0 \\
  0 & \text{otherwise}
\end{cases}
\]

\[
G(i,a) = \begin{cases} 
  g p_{s2}\beta_2 & \text{if } m_2 = P \\
  g \beta_2 & \text{if } m_2 = R \\
  0 & \text{otherwise}
\end{cases}
\]

where \(c_b, c_s, c_{p1}, c_{p2}, c_{b1}, c_{b2}, c_{l1}, c_{l2}, c_{s2}, g\) are positive constants provided to the model.

Based on the values of \(\tau(i,a), p(i,j,a)\) and \(R(i,a)\) given above, the Value-Iteration Algorithm [Tijms, 1986] has been used to obtain the policy that maximizes the long-run average
Proceedings of *International Conference on Industrial Logistics (ICIL)*, Ouro Preto, MG, Brazil, December 1995

revenue per unit time. Performance measures have been obtained by using a post-optimality analysis. Some of these measures are shown in the next section.

### 3. Numerical Results

As an illustration, consider the input data shown in Table 1. Besides these values, the following data have been considered:

- Storage cost: $c_h = 5$
- Sale price of a product: $g = 50$

<table>
<thead>
<tr>
<th></th>
<th>Machine $M_1$</th>
<th>Machine $M_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing rate</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Breakdown rate</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Repair rate</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Product rework rate</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>Breakdown-during-rework rate</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Repair-during-rework rate</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Restarting cost</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Processing cost</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Repair cost</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Starving cost</td>
<td>-</td>
<td>20</td>
</tr>
<tr>
<td>Inspection: Cost of losing a product</td>
<td>50</td>
<td>70</td>
</tr>
<tr>
<td>Inspection: Probability of losing a product</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Inspection: Rework probability</td>
<td>0.05</td>
<td>0.05</td>
</tr>
<tr>
<td>Inspection: Probability of no defect</td>
<td>0.9</td>
<td>0.9</td>
</tr>
</tbody>
</table>

**Table 1 - Input data**

Figure 3 shows the variation of the minimum average revenue in terms of the maximum size of the inventory ($N$). As can be seen, the size of the inventory up to a certain maximum limit is very important to improve the average revenue. In this example, it is not worth to use an inventory size larger than 8.
Figure 3 - Average revenue in terms of maximum size of the inventory

Based on the input values shown in Table 1 and a inventory size of 8, a post-optimal analysis yielded some performance measures for each machine using optimal policy shown in Table 2. Other performance measures obtained are:

Expected storage cost = 11.37
Average size of the inventory = 2.27
Expected reward for producing the products = 133.22

<table>
<thead>
<tr>
<th></th>
<th>Machine M₁</th>
<th>Machine M₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing rate</td>
<td>2.80</td>
<td>2.66</td>
</tr>
<tr>
<td>Rate of losing a product</td>
<td>0.15</td>
<td>0.14</td>
</tr>
<tr>
<td>Availability</td>
<td>0.87</td>
<td>0.71</td>
</tr>
<tr>
<td>Idleness</td>
<td>0.37</td>
<td>0.14</td>
</tr>
<tr>
<td>Restart rate</td>
<td>0.28</td>
<td>0.45</td>
</tr>
<tr>
<td>Starving cost</td>
<td>-</td>
<td>2.75</td>
</tr>
<tr>
<td>Restart cost</td>
<td>2.79</td>
<td>4.53</td>
</tr>
<tr>
<td>Processing cost</td>
<td>25.22</td>
<td>28.75</td>
</tr>
<tr>
<td>Repair cost</td>
<td>6.30</td>
<td>14.37</td>
</tr>
<tr>
<td>Cost of losing a product</td>
<td>7.38</td>
<td>9.82</td>
</tr>
</tbody>
</table>

Table 2 - Performance measures

4. Conclusions

The models discussed in [Hwang and Koh, 1992] and [Gopalan and Kannan, 1994] have been extended using a Semi-Markov Decision Model approach. The present paper considered inspection, rework and in-process inventory. Satisfactory results have been obtained. This paper also shows that in-process inventory is essential whenever failures of production machines, in a transfer line system, must be considered.
Proceedings of *International Conference on Industrial Logistics (ICIL)*, Ouro Preto, MG, Brazil, December 1995

This work is a part of an ongoing research project dealing with the application of Markov models in the study of performance measures of systems. This project is partially funded by the Brazilian Research Council (CNPq) under grant n° 500756/90-2.

The model discussed has been implemented in C++. A first prototype of a class structure to deal with Markov and Semi-Markov Decision Processes has been used.

5. Bibliography


