Finite buffers \( (0 \leq n_i(t) \leq N_i) \).

Single closed loop – fixed population \( (\sum_i n_i(t) = N) \).

Focus is on the Buzacott model (deterministic processing time; geometric up and down times). Repair probability = \( r_i \); failure probability = \( p_i \). Many results are true for more general loops.

Goal: calculate production rate and inventory distribution.
• Limited pallets/fixtures.
• CONWIP (or hybrid) control systems.
• Extension to more complex systems and policies.
Special Case

Refer to *MSE* Section 5.6, page 205.

\[
P_{\text{loop}}(r_1, p_1, r_2, p_2, N_1, N_2) = P_{\text{line}}(r_1, p_1, r_2, p_2, N^{*})
\]

where

\[
N^{*} = \min(n, N_1) - \max(0, n - N_2)
\]
Two-Machine Loops

Special Case

<table>
<thead>
<tr>
<th>$r_1$</th>
<th>.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>.01</td>
</tr>
<tr>
<td>$r_2$</td>
<td>.1</td>
</tr>
<tr>
<td>$p_2$</td>
<td>.01</td>
</tr>
<tr>
<td>$N_1$</td>
<td>20</td>
</tr>
</tbody>
</table>

- $r_1 = 0.14$
- $p_1 = 0.01$
- $r_2 = 0.1$
- $p_2 = 0.01$
- $N_1 = 20$

Graph showing production rate vs population with different values of $N_2$ (10, 15, 20, 30, 40).

Copyright ©2007 Stanley B. Gershwin.
Expected population method

- Treat the loop as a line in which the first machine and the last are the same.
- In the resulting decomposition, one equation is missing.
- The missing equation is replaced by the expected population constraint \( \sum_i \bar{n}_i = N \).
Evaluate \( i, i - 1, i + 1 \) modulo \( k \) (ie, replace 0 by \( k \) and replace \( k + 1 \) by 1).

\[
\begin{align*}
 r_u(i) &= r_u(i - 1) X(i) + r_i(1 - X(i)); \\
 X(i) &= \frac{p_s(i - 1) r_u(i)}{p_u(i) E(i - 1)} \\
 p_u(i) &= r_u(i) \left( \frac{1}{E(i - 1)} + \frac{1}{e_i} - 2 - \frac{p_d(i - 1)}{r_d(i - 1)} \right), \quad \boxed{i = 1, \ldots, k} \\
 r_d(i) &= r_d(i + 1) Y(i + 1) + r_{i+1}(1 - Y(i + 1)); \\
 Y(i + 1) &= \frac{p_b(i + 1) r_d(i)}{p_d(i) E(i + 1)}.
\end{align*}
\]

\[
\begin{align*}
 p_d(i) &= r_d(i) \left( \frac{1}{E(i + 1)} + \frac{1}{e_{i+1}} - 2 - \frac{p_u(i + 1)}{r_u(i + 1)} \right), \quad \boxed{i = k, \ldots, 1}
\end{align*}
\]

This is \( 4k \) equations in \( 4k \) unknowns. But only \( 4k - 1 \) of them are independent because the derivation uses \( E(i) = E(i + 1) \) for \( i = 1, \ldots, k \). The first \( k - 1 \) are \( E(1) = E(2), E(2) = E(3), \ldots, E(k - 1) = E(k) \). But this implies \( E(k) = E(1) \), which is the same as the \( k \)th equation.
Therefore, we need one more equation. We can use

$$\sum_i \bar{n}_i = N$$

If we guess $p_u(1)$ (say), we can evaluate $\bar{n}^{\text{TOTAL}} = \sum_i \bar{n}_i$ as a function of $p_u(1)$. We search for the value of $p_u(1)$ such that $\bar{n}^{\text{TOTAL}} = N$. 
Expected population method

Behavior:

- Accuracy good for large systems, not so good for small systems.
- Accuracy good for intermediate-size populations; not so good for very small or very large populations.
**Expected population method**

*Hypothesis:* The reason for the accuracy behavior of the population constraint method is the correlation in the buffers.

- The number of parts in the system is actually *constant*.
- The expected population method treats the population as *random*, with a specified mean.
- If we know that a buffer is almost full, we know that there are fewer parts in the rest of the network, so probabilities of blockage are reduced and probabilities of starvation are increased. (Similarly if it is almost empty.)
- Suppose the population is smaller than the smallest buffer. *Then there will be no blockage.* The expected population method does not take this into account.
Loop Behavior

To construct a method that deals with the invariant (rather than the expected value of the invariant), we investigate how buffer levels are related to one another and to the starvation and blocking of machines.

In a line, every downstream machine could block a given machine, and every upstream machine could starve it. In a loop, blocking and starvation are more complicated.
• The *range of blocking of a machine* is the set of all machines that could block it if they stayed down for a long enough time.

• The *range of starvation of a machine* is the set of all machines that could starve it if they stayed down for a long enough time.
• All buffer sizes are 10.
• Population is 37.
• If $M_4$ stays down for a long time, it will block $M_1$.
• Therefore $M_4$ is in the range of blocking of $M_1$.
• Similarly, $M_2$ and $M_3$ are in the range of blocking of $M_1$. 
If $M_5$ stays down for a long time, it will starve $M_1$.

Therefore $M_5$ is in the range of starvation of $M_1$.

Similarly, $M_6$ is in the range of starvation of $M_1$. 
● The range of blocking of a machine in a line is the entire downstream part of the line.

● The range of starvation of a machine in a line is the entire upstream part of the line.
In an acyclic network, if $M_j$ is downstream of $M_i$, then the range of blocking of $M_j$ is a *subset* of the range of blocking of $M_i$.

Similarly for the range of starvation.
In an acyclic network, if $M_j$ is downstream of $M_i$, any real machine whose failure could cause $M_d(j)$ to appear to be down could also cause $M_d(i)$ to appear to be down.

Consequently, we can express $r_d(i)$ as a function of the parameters of $L(j)$. This is not possible in a network with a loop because some machine that blocks $M_j$ does not block $M_i$. 
Ranges of blocking and starvation of adjacent machines are *not* subsets or supersets of one another in a loop.
$M_5$ can block $M_2$. Therefore the parameters of $M_5$ should directly affect the parameters of $M_d(1)$ in a decomposition. However, $M_5$ cannot block $M_1$ so the parameters of $M_5$ should not directly affect the parameters of $M_d(6)$. Therefore, the parameters of $M_d(6)$ cannot be functions of the parameters of $M_d(1)$.
Multiple Failure Mode Line Decomposition

- To deal with this issue, we introduce a new decomposition.
- In this decomposition, we do not create failures of virtual machines that are mixtures of failures of real machines.
- Instead, we allow the virtual machines to have distinct failure modes, each one corresponding to the failure mode of a real machine.
• There is an observer in each buffer who is told that he is actually in the buffer of a two-machine line.
Multiple Failure Mode Line Decomposition

- Each machine in the original line may and in the two-machine lines must have multiple failure modes.
- For each failure mode downstream of a given buffer, there is a corresponding mode in the downstream machine of its two-machine line.
- Similarly for upstream modes.
Multiple Failure Mode Line Decomposition

- The downstream failure modes appear to the observer after propagation through *blockage*.
- The upstream failure modes appear to the observer after propagation through *starvation*.
- The two-machine lines are more complex than in earlier decompositions but the decomposition equations are simpler.
Form a Markov chain and find the steady-state probability distribution. The solution technique is very similar to that of the two-machine-state model. Determine the production rate, probability of starvation and probability of blocking in each down mode, average inventory.
• A set of decomposition equations are formulated.
• They are solved by a Dallery-David-Xie-like algorithm.
• The results are a little more accurate than earlier methods, especially for machines with very different failures.
In the upstream machine of the building block, failure mode 4 is a *local* mode; modes 1, 2, and 3 are *remote* modes. Modes 5, 6, and 7 are local modes of the downstream machine; 8, 9, and 10 are remote modes.

For *every* mode, the repair probability is the same as the repair probability of the corresponding mode in the real line.

*Local modes:* the probability of failure into a local mode is the same as the probability of failure in that mode of the real machine.
Multiple Failure Mode Line Decomposition

Remote modes: $i$ is the building block number; $j$ and $f$ are the machine number and mode number of a remote failure mode. Then

$$p_{jf}^u(i) = \frac{P_{s,jf}(i - 1)}{E(i)} r_{jf}; \quad p_{jf}^d(i - 1) = \frac{P_{b,jf}(i)}{E(i - 1)} r_{jf}$$

where $p_{jf}^u(i)$ is the probability of failure of the upstream machine into mode $jf$; $P_{s,jf}(i - 1)$ is the probability of starvation of line $i - 1$ due to mode $jf$; $r_{jf}$ is the probability of repair of the upstream machine from mode $jf$; etc.

Also, $E(i - 1) = E(i)$.

$p_{jf}^u(i), p_{jf}^d(i)$ are used to evaluate $E(i), P_{s,jf}(i), P_{b,jf}(i)$ from two-machine line $i$ in an iterative method.
Consider

\[ p_{jf}^u(i) = \frac{P_{s,jf}(i - 1)}{E(i)} r_{jf}; \quad p_{jf}^d(i - 1) = \frac{P_{b,jf}(i)}{E(i - 1)} r_{jf} \]

In a line, \( jf \) refers to all modes of \( all \) upstream machines in the first equation; and all modes of \( all \) downstream machines in the second equation.

We can interpret the upstream machines as the range of starvation and the downstream machines as the range of blockage of the line.
- Use the multiple-mode decomposition, but adjust the ranges of blocking and starvation accordingly.
- However, this does not take into account the local information that the observer has.
The $B_6$ observer knows how many parts there are in his buffer. If there are 5, he knows that the modes he sees in $M^d(6)$ could be those corresponding to the modes of $M_1$, $M_2$, $M_3$, and $M_4$. 

Copyright ©2007 Stanley B. Gershwin.
• However, if there are 8, he knows that the modes he sees in $M^d(6)$ could only be those corresponding to the modes of $M_1$, $M_2$, and $M_3$; and not those of $M_4$.

• The transition probabilities of the two-machine line therefore depend on whether the buffer level is less than 7 or not.
Thresholds

- This would require a new model of a two-machine line.
- The same issue arises for starvation.
- In general, there can be more than one threshold in a buffer.
- Consequently, this makes the two-machine line very complicated.
Transformation

- **Purpose:** to avoid the complexities caused by thresholds.
- **Idea:** Wherever there is a threshold in a buffer, break up the buffer into smaller buffers separated by perfectly reliable machines.
Transformation

- When $M_1$ fails for a long time, $B_4$ and $B_3$ fill up, and there is one part in $B_2$. Therefore there is a threshold of 1 in $B_2$.
- When $M_2$ fails for a long time, $B_1$ fills up, and there is one part in $B_4$. Therefore there is a threshold of 1 in $B_4$.
- When $M_3$ fails for a long time, $B_2$ fills up, and there are 18 parts in $B_1$. Therefore there is a threshold of 18 in $B_1$. 
When $M_4$ fails for a long time, $B_3$ and $B_2$ fill up, and there are 13 parts in $B_1$. Therefore there is a threshold of 13 in $B_1$.

- **Note:** $B_1$ has two thresholds and $B_3$ has none.
- **Note:** The number of thresholds equals the number of machines.
• Break up each buffer into a sequence of buffers of size 1 and reliable machines.
• Count backwards from each real machine the number of buffers equal to the population.
• Identify the reliable machine that the count ends at.
• Collapse all the sequences of unmarked reliable machines and buffers of size 1 into larger buffers.
Transformation

• Ideally, this would be equivalent to the original system.
• However, the reliable machines cause a delay, so transformation is not exact for the discrete/deterministic case.
• This transformation *is* exact for continuous-material machines.
Small populations

Transformation

- If the population is smaller than the largest buffer, at least one machine will *never* be blocked.
- However, that violates the assumptions of the two-machine lines.
- We can reduce the sizes of the larger buffers so that no buffer is larger than the population. This does not change performance.
Numerical Results

• Many cases were compared with simulation:
  
  ★ *Three-machine cases*: all throughput errors under 1%; buffer level errors averaged 3%, but were as high as 10%.
  
  ★ *Six-machine cases*: mean throughput error 1.1% with a maximum of 2.7%; average buffer level error 5% with a maximum of 21%.
  
  ★ *Ten-machine cases*: mean throughput error 1.4% with a maximum of 4%; average buffer level error 6% with a maximum of 44%.
Numerical Results

- Convergence reliability: almost always.
- Speed: execution time increases rapidly with loop size.
- Maximum size system: 18 machines. Memory requirements grow rapidly also.
Numerical Results

The Batman Effect

- Error is very small, but there are apparent discontinuities.
- This is because we cannot deal with buffers of size 1, and because we do not need to introduce reliable machines in cases where there would be no thresholds.
• All buffer sizes 10. Population 15. Identical machines except for $M_1$.

• Observe average buffer levels and production rate as a function of $r_1$. 
• Production rate vs. $r_1$.
• Usual saturating graph.
• When \( r_1 \) is small, \( M_1 \) is a bottleneck, so \( B_4 \) holds 10 parts, \( B_3 \) holds 5 parts, and the others are empty.

• As \( r_1 \) increases, material is more evenly distributed. When \( r_1 = 0.1 \), the network is totally symmetrical.
Applications

- Design system with pallets/fixtures. The fixtures and the space to hold them in are expensive.
- Design system with tokens/kanbans (CONWIP). By limiting population, we reduce production rate, but we also reduce inventory.
2.852 Manufacturing Systems Analysis
Spring 2010

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.