Analysis of Loop Networks by Decomposition

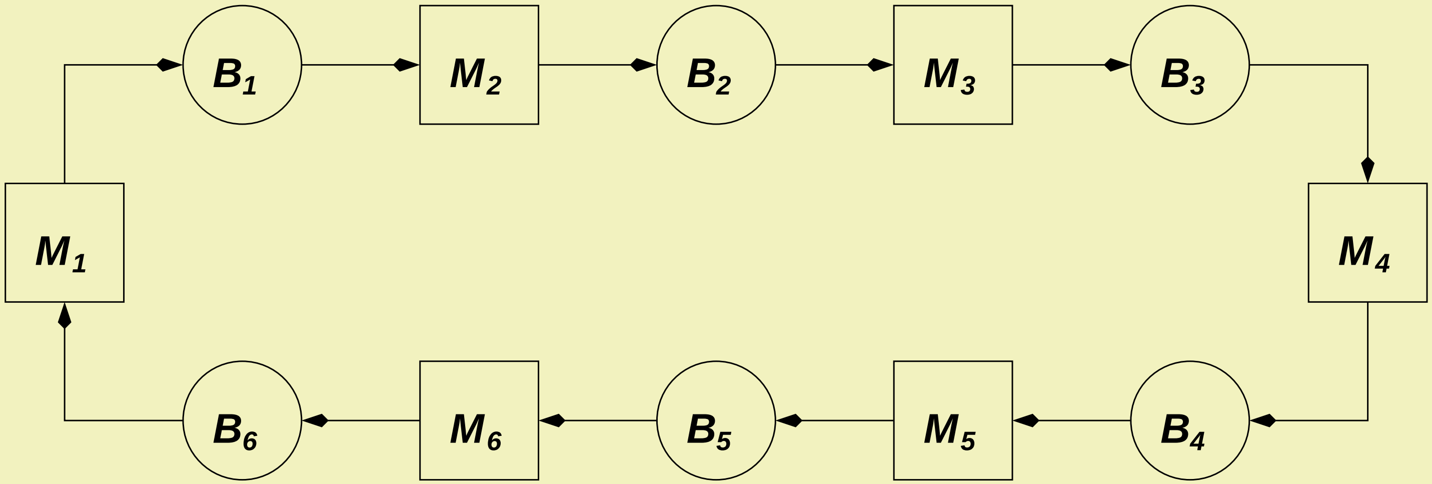
Stanley B. Gershwin Nicola Maggio
Andrea Matta Tullio Tolio Loren Werner

May 22, 2001

Outline

- Introduction
- Tolio Decomposition of a Line
- Extension to Loops
- Thresholds
- Transformation
- Numerical Results
- Conclusions

Introduction



- Finite buffers ($0 \leq n_i(t) \leq N_i$).
- Closed loop – fixed population ($\sum_i n_i(t) = N$).
- Buzacott model (deterministic processing time; geometric up and down times). Repair probability = r_i failure probability = p_i.
- *Goal:* calculate production rate and inventory distribution.

Introduction

- Limited pallets/fixtures.
- CONWIP (or hybrid).
- Extension to more complex systems and policies.

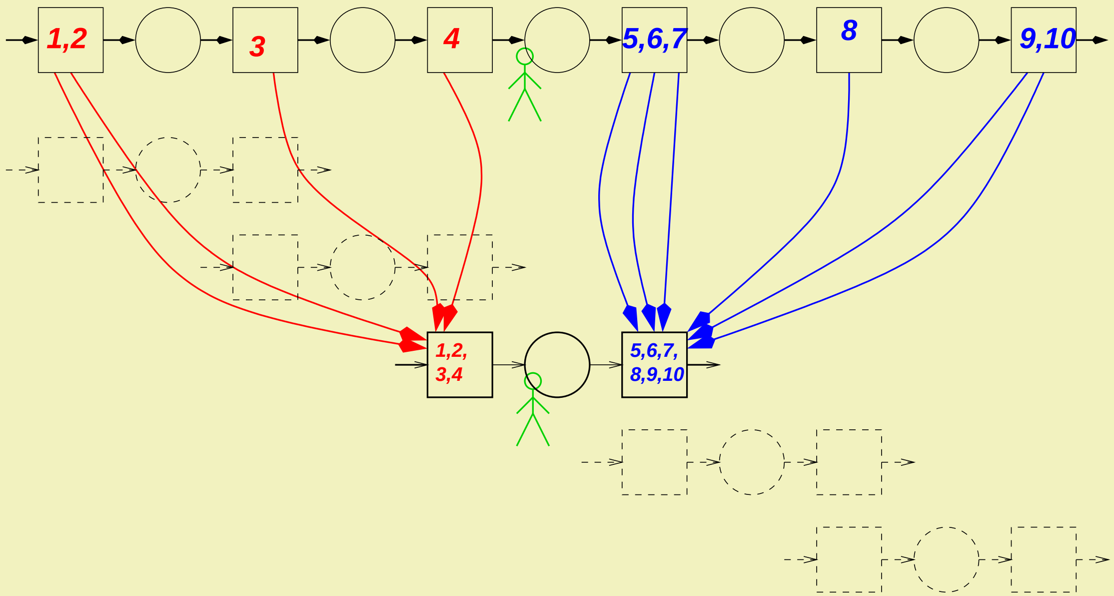
Introduction

- Frein, Commault, Dallery (1996):
  - ⋆ 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 population constraint $(\sum_i \bar{n}_i(t) = N)$.
  - ⋆ 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.

Introduction

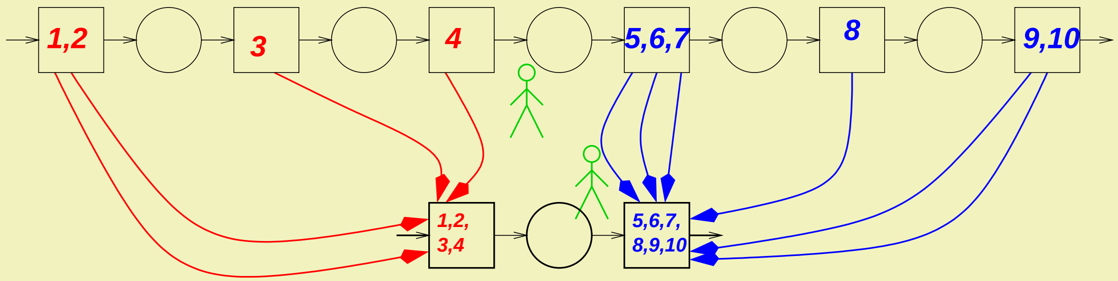
- *Hypothesis:* The reason for the accuracy behavior of the Frein-Commault-Dallery method is the correlation in the buffers.
  - ⋆ The number of parts in the system is actually *constant* .
  - ⋆ Frein-Commault-Dallery treats the population as *random,* with a specified mean.
- Therefore, we develop an approach that treats the population as constant.

Tolio Line Decomposition



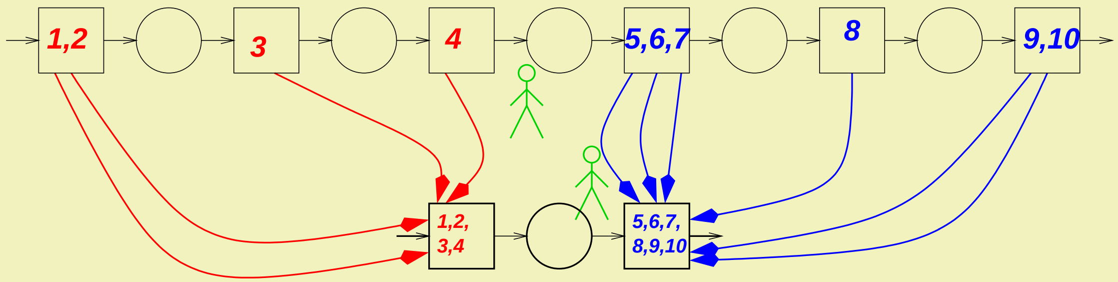
- There is an observer in each buffer who is told that he is actually in the buffer of a two-machine line.

Tolio 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.

Tolio Line Decomposition

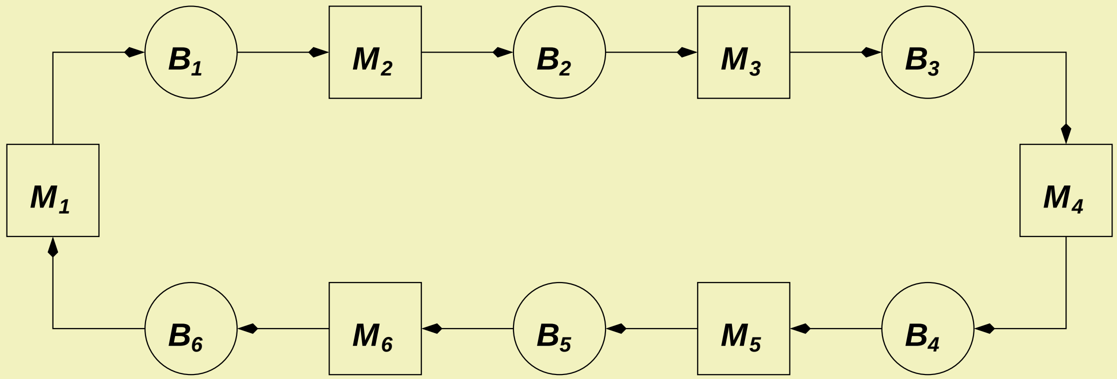


- The downstream failure modes appear to the observer after propagation through *blockage* .
- The upstream failure modes appear to the observer after propagation through *starvation* .
- The two-machine lines are more complex that in earlier decompositions but the decomposition equations are simpler.

Tolio Line Decomposition

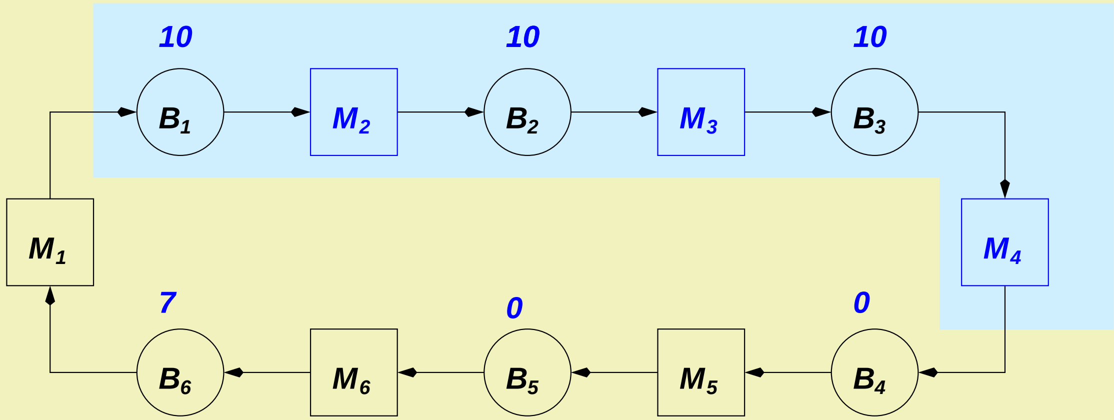
- A set of decomposition equations are formulated.
- They are solved by a Dallery-David-Xie-like algorithm.
- The results are a little more accurate than earlier methods.

Extension to Loops



- 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.

Extension to Loops

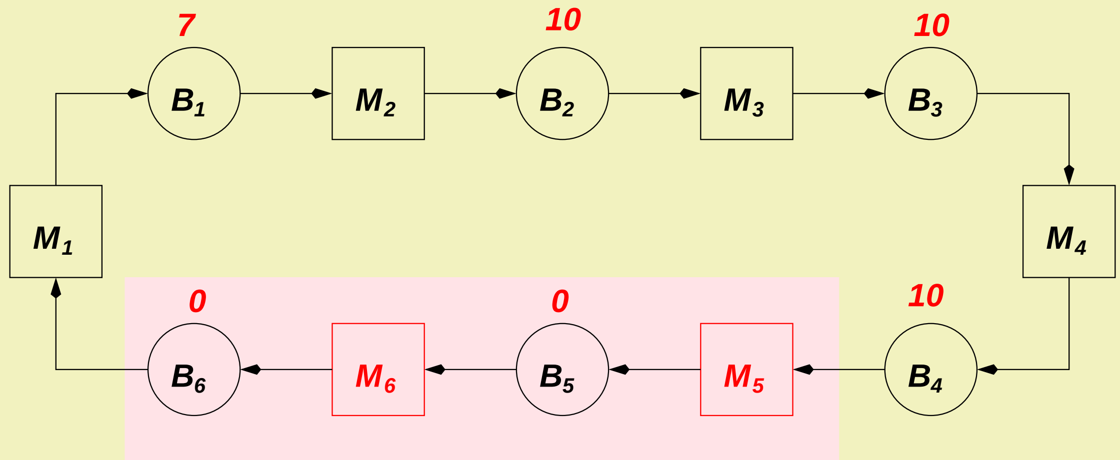


- All buffer sizes are 10.
- Population is 37.
- If M_4 stays down for a long time, it will block M_1.
- Therefore M_4 is in the range of blocking of M_1.
- Similarly, M_2 and M_3 are in the range of blocking of M_1.

Extension to Loops

Ranges

Range of starvation

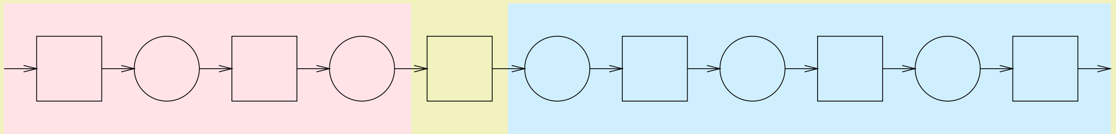


- If M_5 stays down for a long time, it will starve M_1.
- Therefore M_5 is in the range of starvation of M_1.
- Similarly, M_6 is in the range of starvation of M_1.

Extension to Loops

- 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.

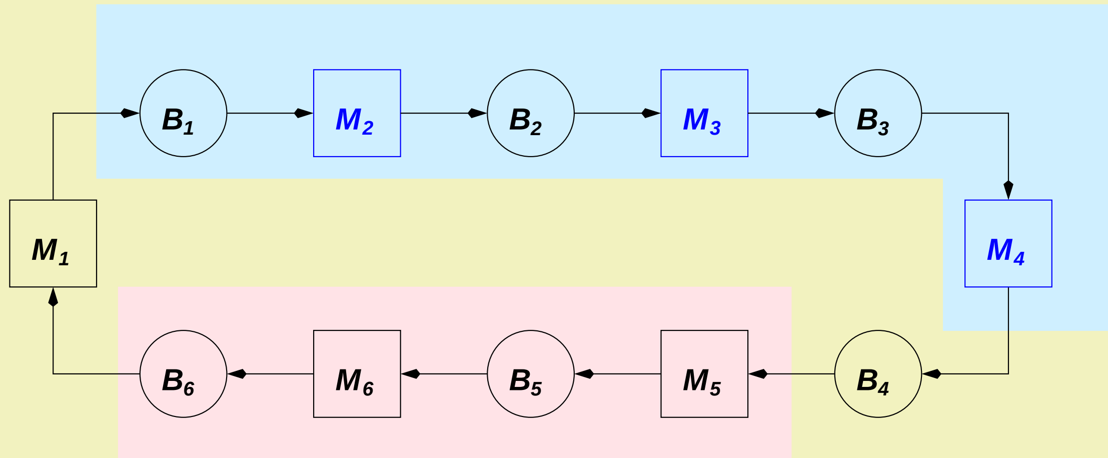
Extension to Loops



- 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.

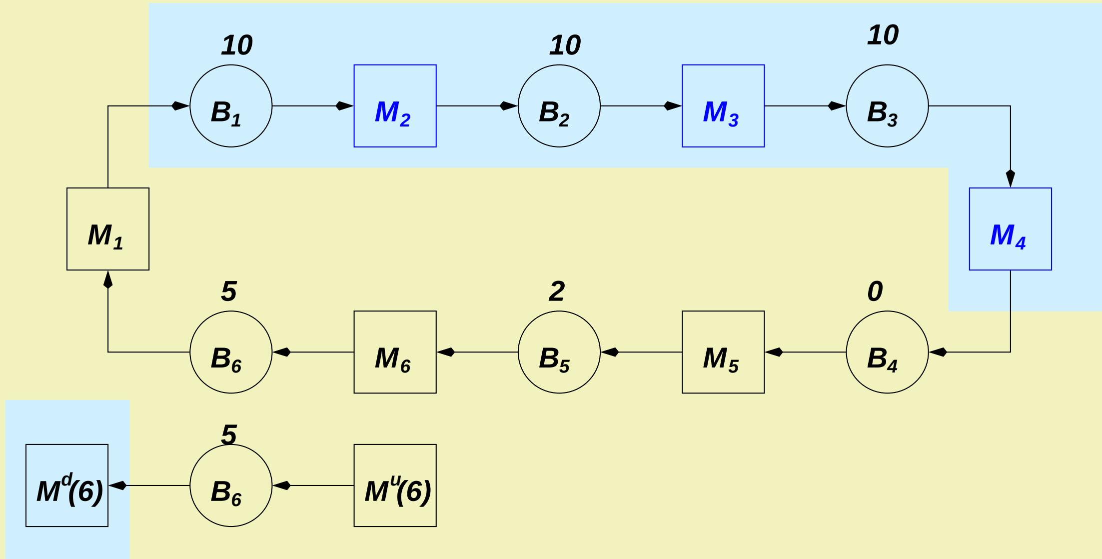
Extension to Loops

Simple decomposition



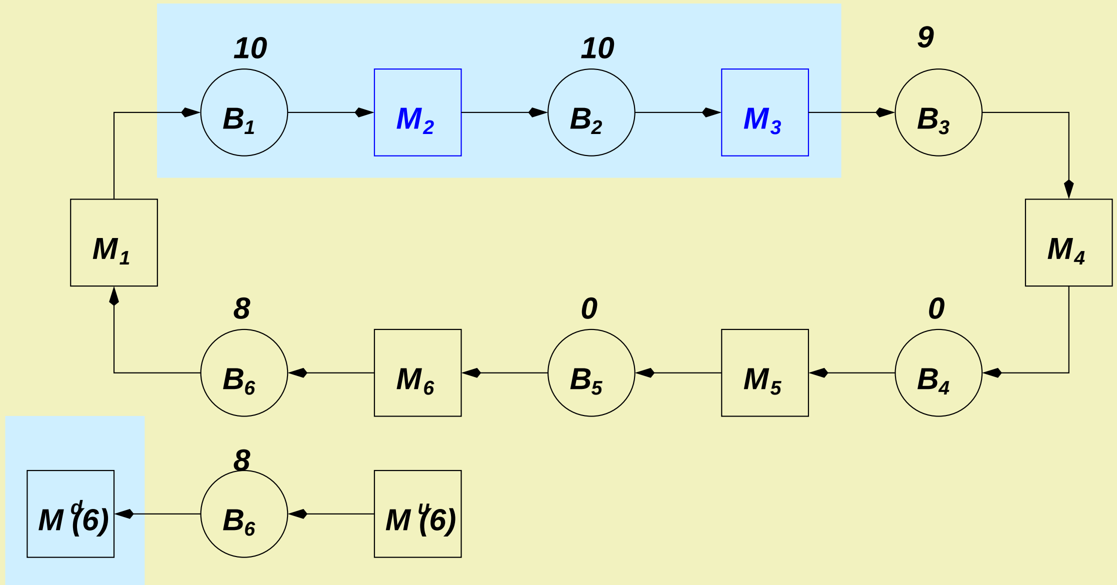
- Use the Tolio decomposition, but adjust the ranges of blocking and starvation accordingly.
- However, this does not take into account the local information that the observer has.

Thresholds



- 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.

Thresholds



- 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.*

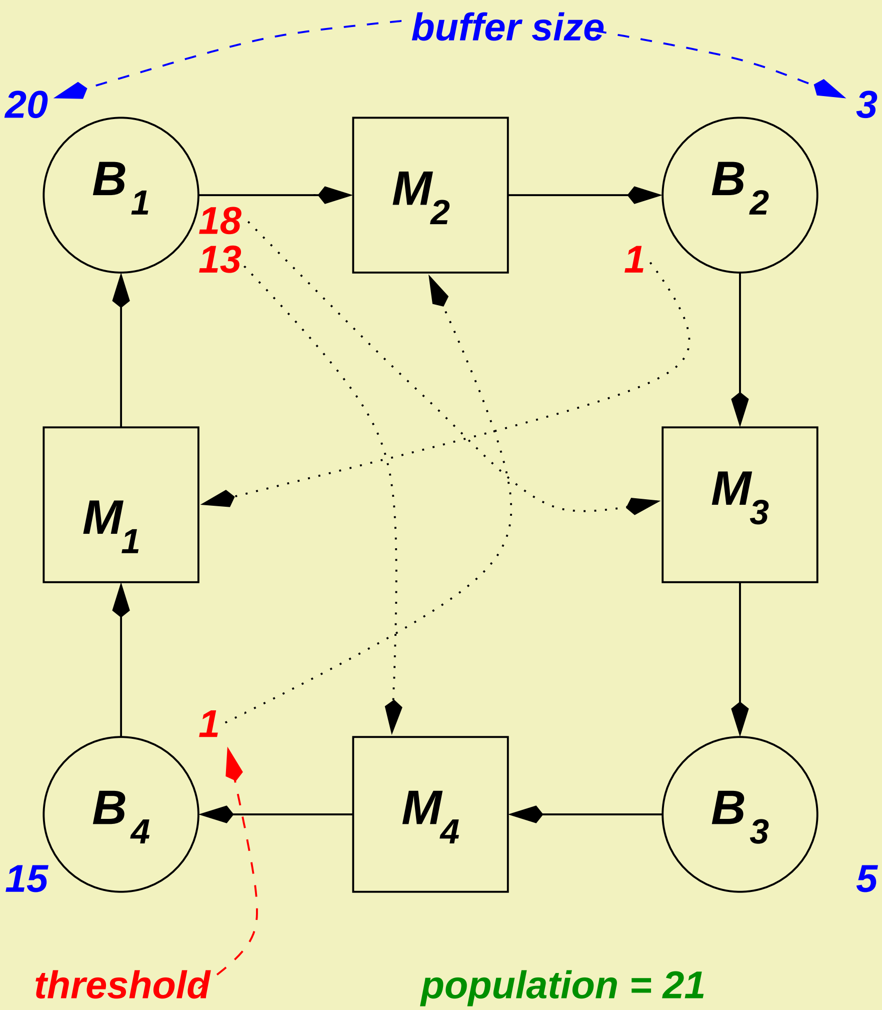
Thresholds

- 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.
- We analyzed three-machine loops, to keep the complexity limited, and found very good agreement with simulation.

Transformation

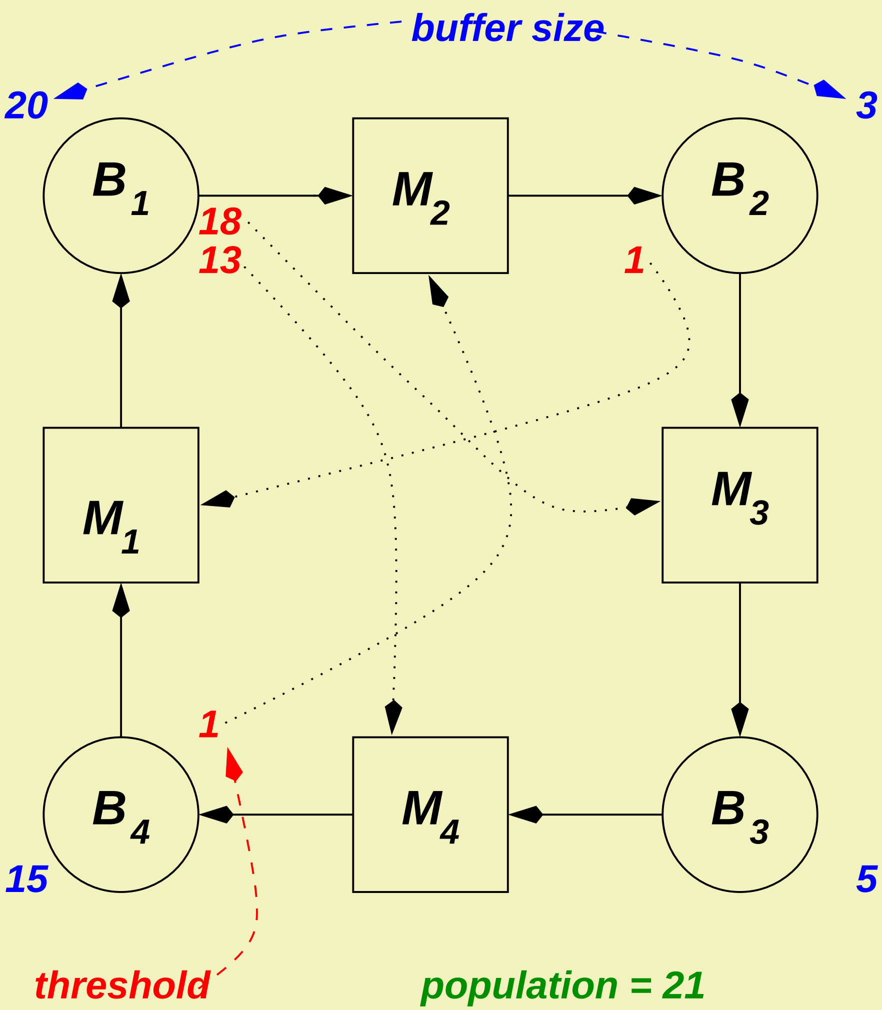
- *Purpose:* to avoid the complexities caused by thresholds.
- *Idea:* Wherever there is a threshold in a buffer, break up the buffer into smaller buffers separated by perfectly reliable machines.

Transformation



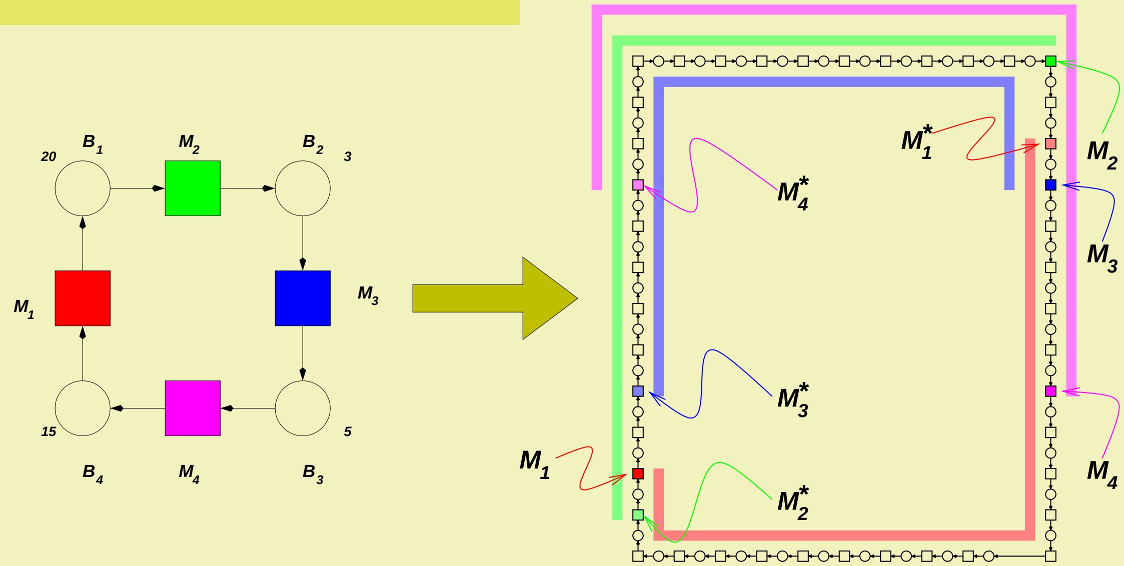
- When M_1 fails for a long time, B_4 and B_3 fill up, and there is one part in B_2. Therefore there is a threshold of 1 in B_2.
- When M_2 fails for a long time, B_1 fills up, and there is one part in B_4. Therefore there is a threshold of 1 in B_4.
- When M_3 fails for a long time, B_2 fills up, and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

Transformation



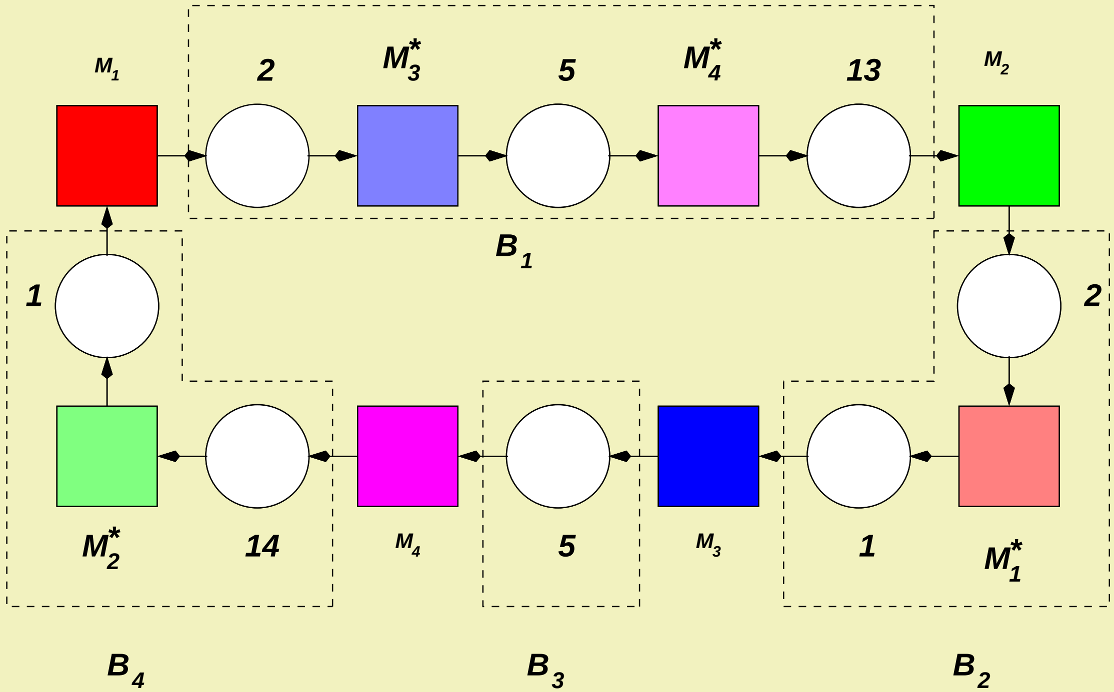
- When M_4 fails for a long time, B_3 and B_2 fill up, and there are 13 parts in B_1. Therefore there is a threshold of 13 in B_1.
- *Note:* B_1 has two thresholds and B_3 has none.
- *Note:* The number of thresholds equals the number of machines.

Transformation



- 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 the count ends at.

Transformation



- 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.

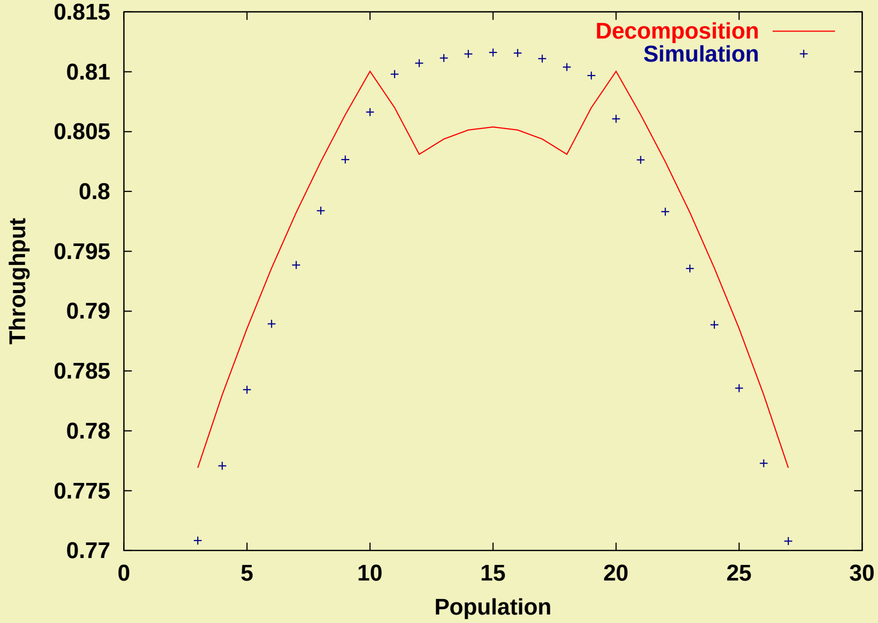
Numerical Results

- 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

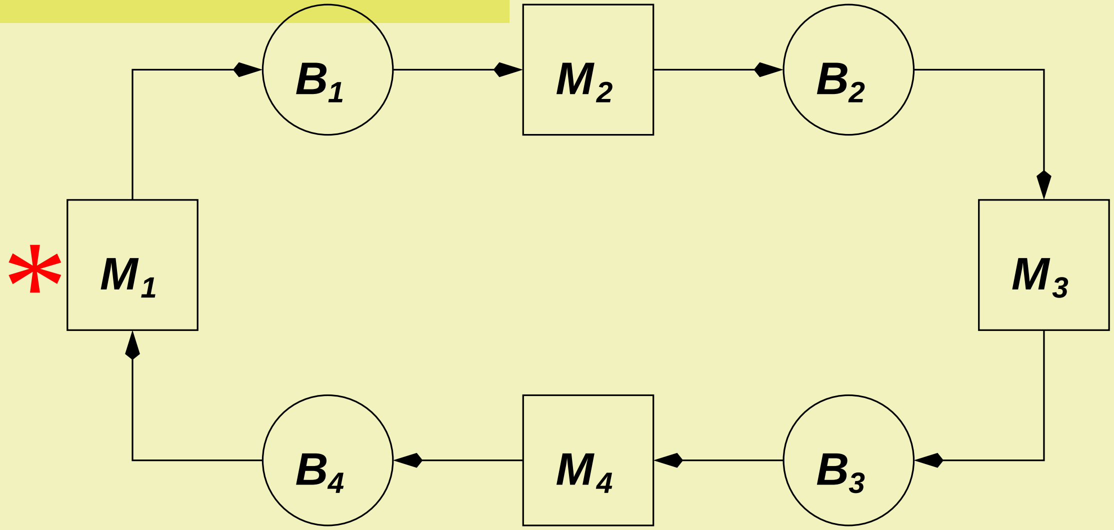
- Convergence reliability: almost always.
- Speed: execution time increases rapidly with loop size.
- Maximum size system: 18 machines. Memory requirements grow rapidly also.

Numerical Results



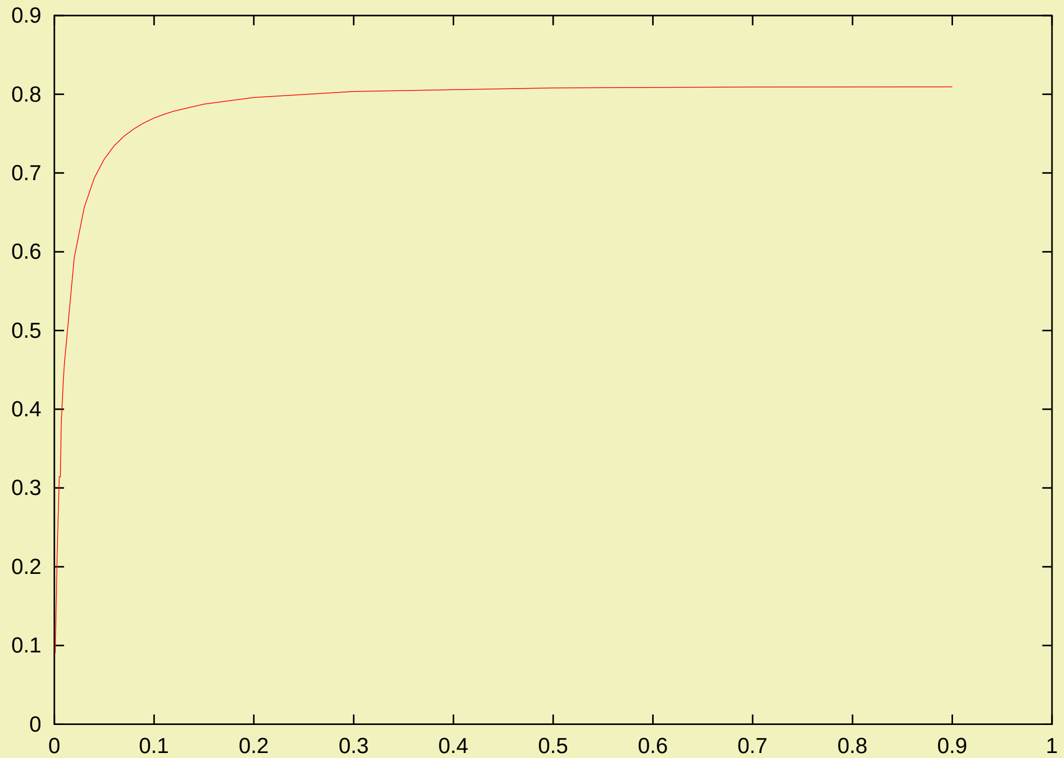
- Error is very small, but there are apparent discontinuities.
- This is because we cannot deal with buffers of size 1, and because we do not need to introduce reliable machines in cases where there would be no thresholds.

Numerical Results



- 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.

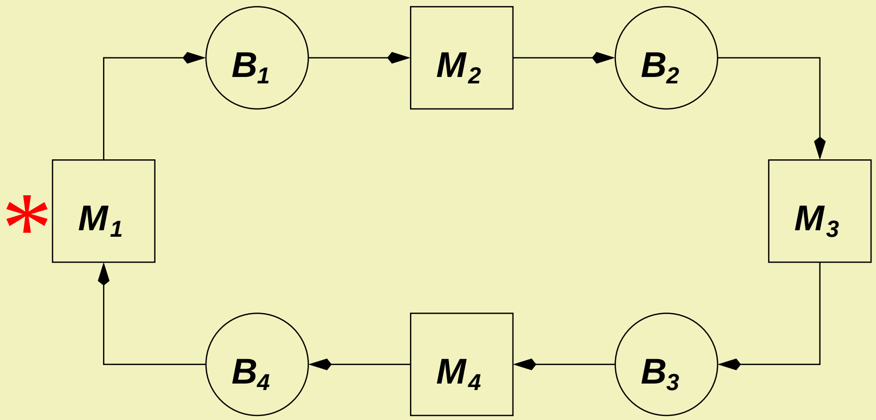
Numerical Results

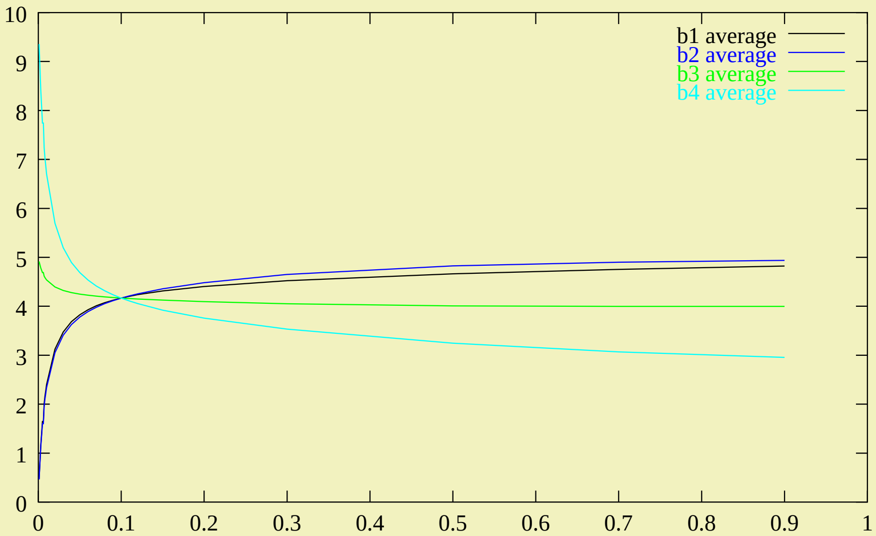


- Usual saturating graph.

Numerical Results

Behavior





- 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.

Conclusions

- Method is accurate.
- Further research:
  - ⋆ Extend to other models, especially continuous material.
  - ⋆ Make algorithm faster by eliminating unnecessary modes from the two-machine lines.
  - ⋆ Extend to more complex systems.