SMA 6304
Factory Planning and Scheduling
Lecture 24: Current Research

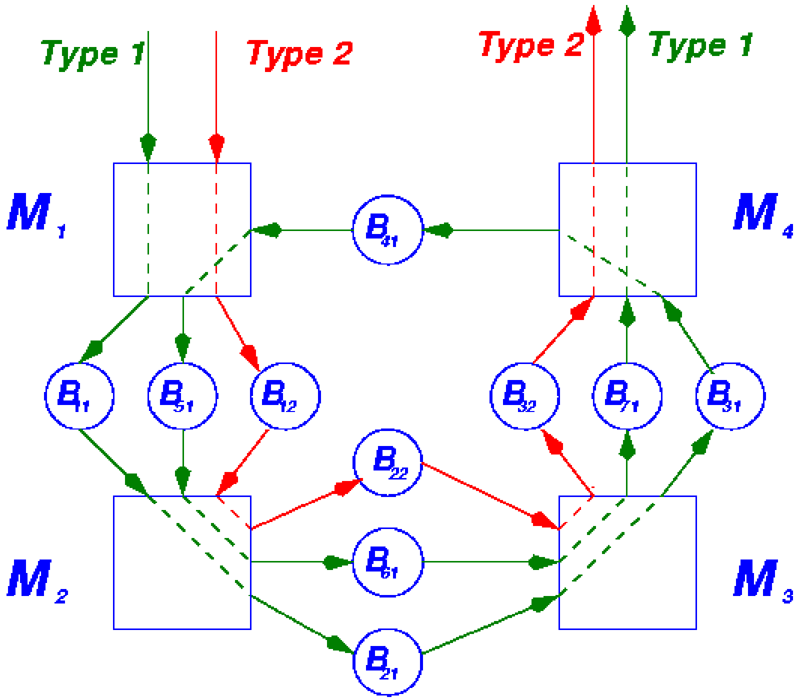
Stanley B. Gershwin

Outline

- Real-time scheduling
- Synthesis
- Loops
- Multiple loops
- Conclusions and future research

Real-time Scheduling

Class of Systems



- Reentrant flow
- Flexible, unreliable machines
- Continuous material
- Homogeneous, finite buffers
- Constant demand rate
- No setups or batches

- $S(s,q)$ is the sth machine that type q parts visit.
- Demand d_q
- Operation time $\tau_{sq} = 1/\mu_{sq}$
- Availability e_i
- ***Feasibility:*** $$\sum_{\{s,q|S(s,q)=i\}} \left(\frac{d_q}{\mu_{sq}}\right) < e_i \text{ for all } i$$

- ***Control:*** $u_{sq}(t)$ is the instantaneous production rate of type q parts at stage s at time t
- Cumulative production $P_{sq}(t) = \int_0^t u_{sq}(\tau) d\tau$
- ***State:***
  - ⋆ $x_{sq}(t) = P_{sq}(t) - d_q t$, surplus
  - ⋆ $\alpha_i(t)$ = 0 or 1: repair state of Machine i

- ***Constraints:***

if $\alpha_i(t) = 0, \qquad u_{sq}(t) = 0;$

if $\alpha_i(t) = 1, \qquad \sum_{\{s,q|S(s,q)=i\}} \left(\frac{u_{sq}(t)}{\mu_{sq}}\right) \leq 1; u_{sq} \geq 0.$

- ***Dynamics:***

$$\frac{dx_{sq}}{dt} = u_{sq} - d_q$$

α : Markov process: exponential up- and down-times

- ***Constraints:***

$$b_{sq} = x_{sq} - x_{s+1,q}$$

$$0 \leq b_{sq} \leq N_{sq}$$

- ***Objective:***

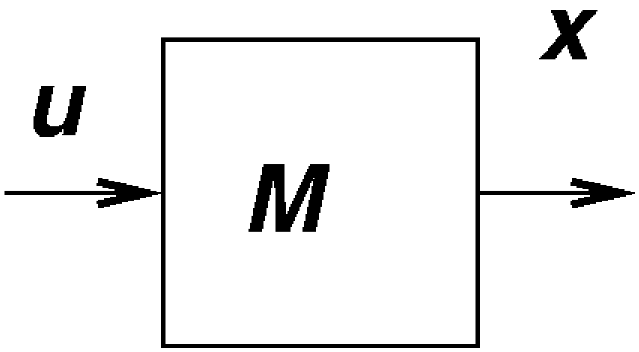
$$J = \min E \int_0^T g(b_{11}(s), b_{12}(s), ..., x_{K(\ell),\ell}(s))ds$$

- Solution is a control law of the form $u(x(t), \alpha(t), t)$.
- Impossible to determine exactly except in special cases.
- Impossible to determine *numerically* except in special cases.
- ***Strategy:*** Investigate special cases and extrapolate.

Real-time Scheduling

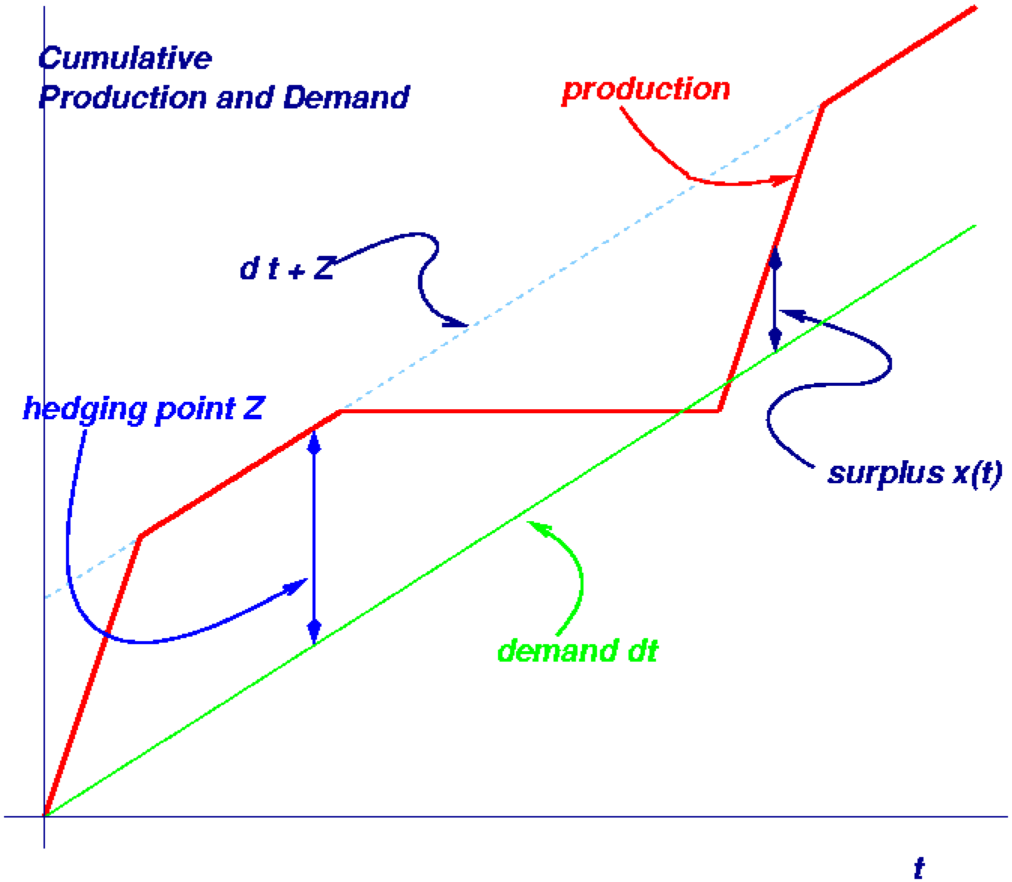
Special case

Bielecki and Kumar, 1988



- Single machine.
- Single part type.
- Constant demand.
- Exponentially distributed time to fail and time to repair.
- *Problem:* choose the production rate at every time instant to minimize inventory and backlog costs.

Special case



Solution:

- if $x(t) > Z$, wait;
- if $x(t) = Z$, operate at demand rate d;
- if $x(t) < Z$, operate at maximum rate μ.

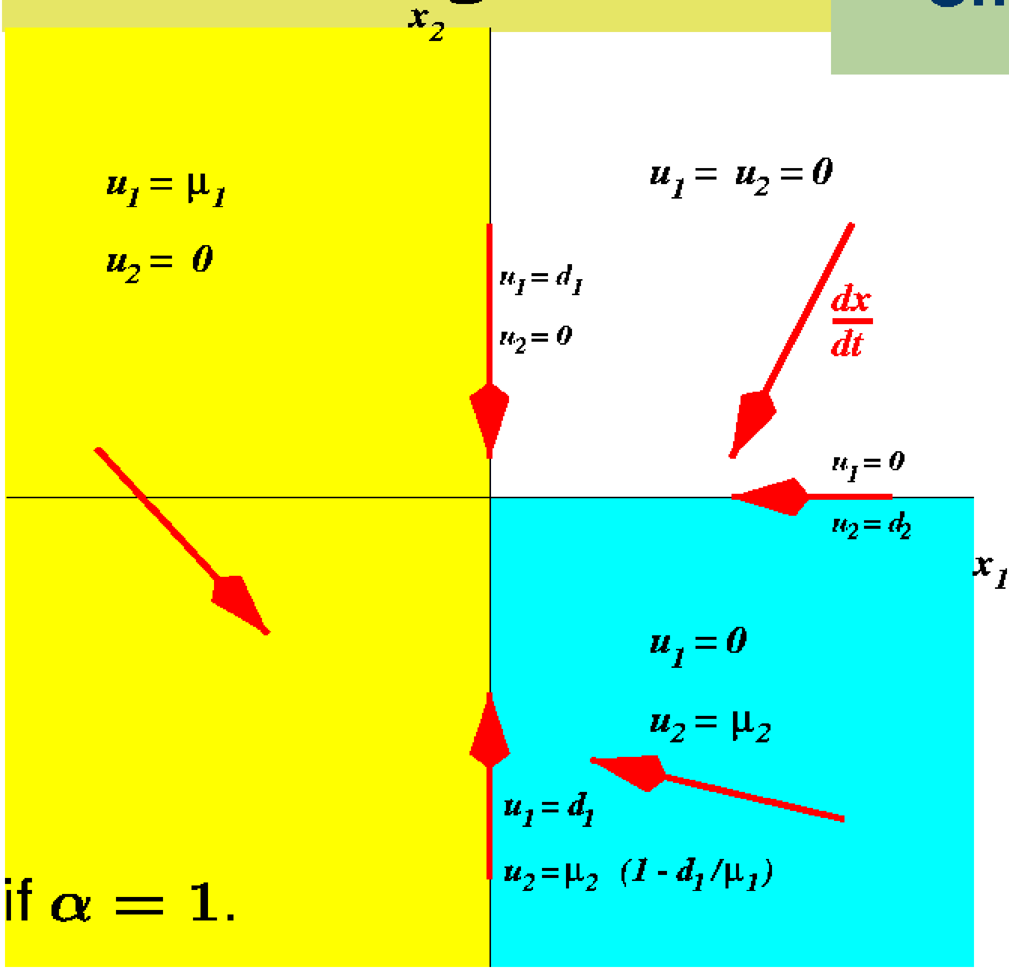
Special case

- The hedging point Z is the single control parameter.
- Z represents a trade-off between costs of inventory and risk of disappointing customers.
- Z is a function of d, μ, r, p, g_+, g_-. A formula exists.
- *There are no complete solutions for any more general case.*

Real-time Scheduling

Other special cases

One machine, two types



- Type 1 has priority because of g.
- Complete solution available only when $Z = 0$.

if $\alpha = 1$.

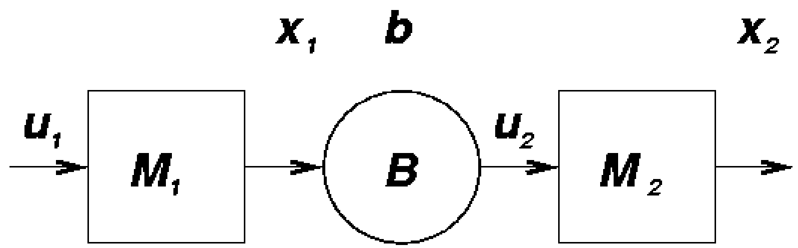
One machine, multiple types

- There is now a surplus vector $\boldsymbol{x}$, and a hedging point vector $\boldsymbol{Z}$. If $\boldsymbol{Z}$=0,
  - ⋆ Rank order the parts
  - ⋆ Drive $\boldsymbol{x_1}$ to $\boldsymbol{Z_1}$.
  - ⋆ Keep $\boldsymbol{x_1} = \boldsymbol{Z_1}$ and drive $\boldsymbol{x_2}$ to $\boldsymbol{Z_2}$.
  - ⋆ Keep $\boldsymbol{x_1} = \boldsymbol{Z_1}$ and $\boldsymbol{x_2} = \boldsymbol{Z_2}$ and drive $\boldsymbol{x_3}$ to $\boldsymbol{Z_3}$.
  - ⋆ Etc.
- If $\boldsymbol{Z} \neq \mathbf{0}$, the optimal solution is not known exactly, but it is known to be more complex.
- *Approximation:* use the same policy.

We now deal with only *single part type systems*, until the very end of the talk.

Other special cases

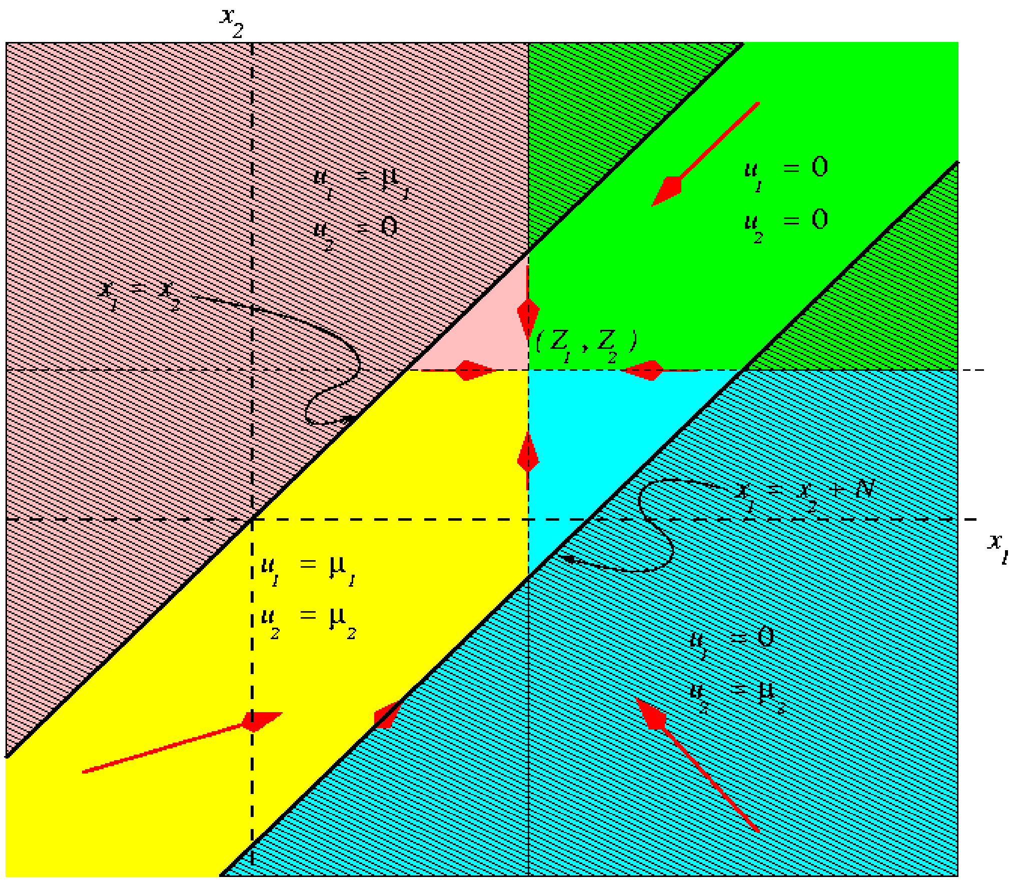
Two machines, one buffer, single type



- Exact optimal solution also impossible to obtain.
- Approximation: *use the one-machine-one-part-type policy at each machine*.

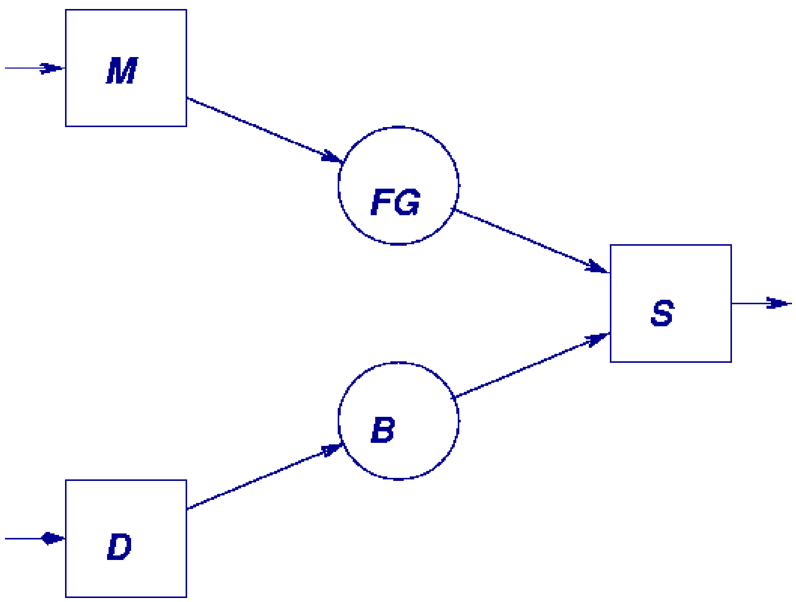
Other special cases

Two machines, one buffer, single type



Synthesis

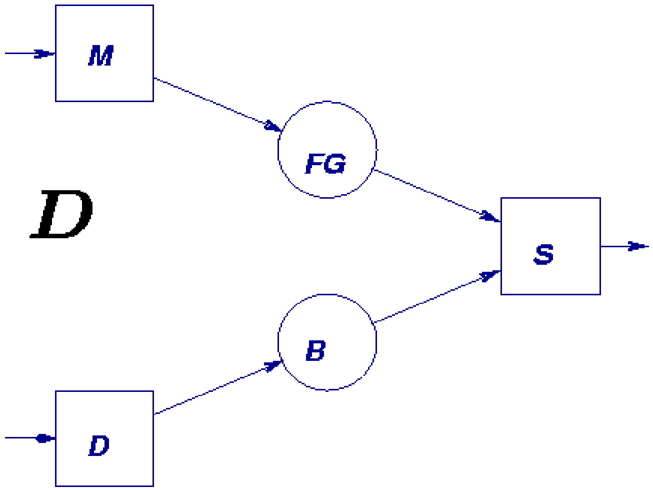
(Gershwin, 2000)



- *Operating Machine* $\boldsymbol{M}$ *according to the hedging point policy is equivalent to operating this assembly system according to a finite buffer policy.*

Synthesis

(Gershwin, 2000)



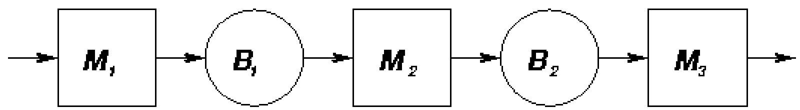
- D is a *demand generator* .
  - ⋆ Whenever a demand arrives, D sends a token to B.
- S is a synchronization machine.
  - ⋆ S is perfectly reliable and infinitely fast.
- FG is a finite finished goods buffer.
- B is an infinite backlog buffer.

Real-time Scheduling

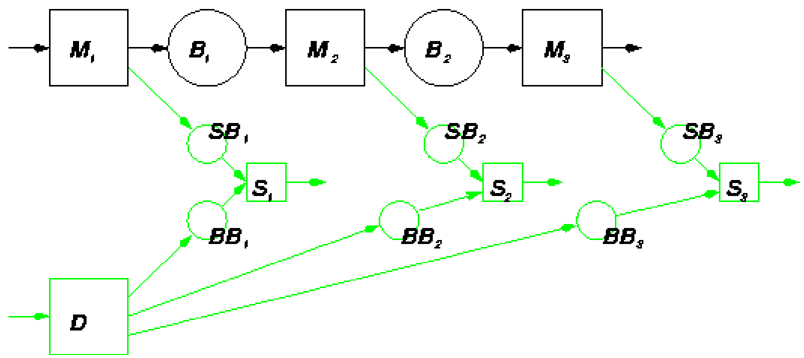
Proposed control policy

(Gershwin, 2000)

To control



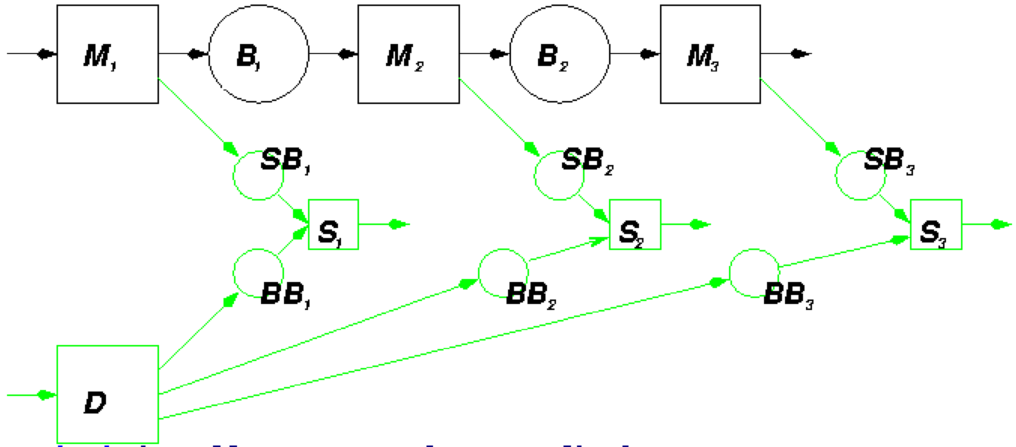
add an information flow system:



Real-time Scheduling

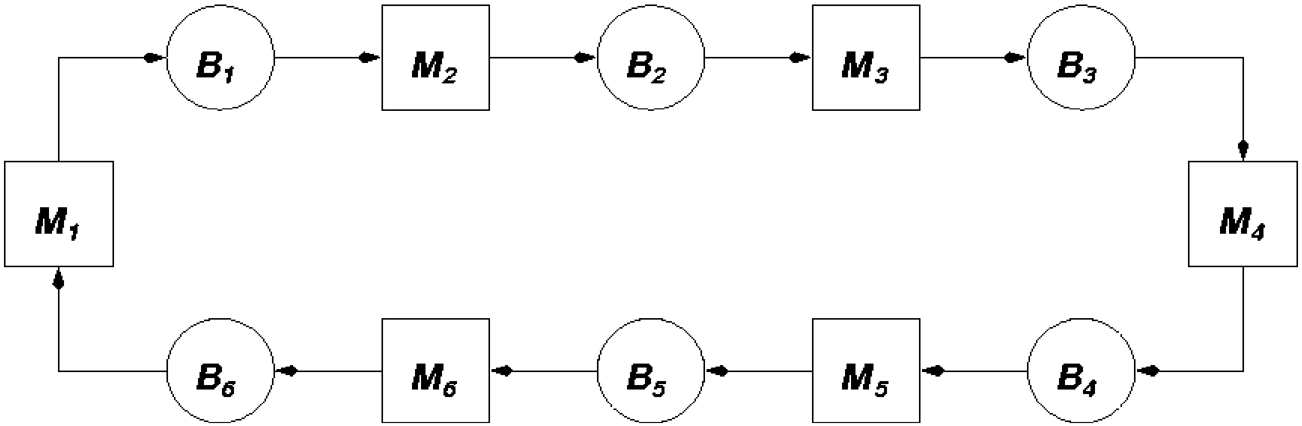
Proposed control policy

(Gershwin, 2000)



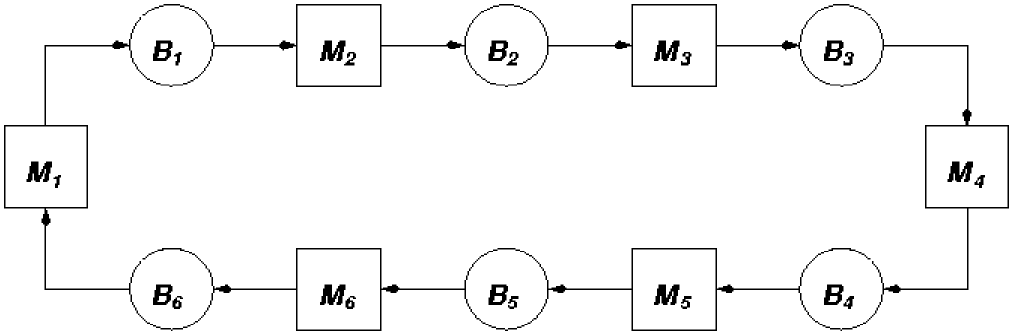
- B_i are *material* buffers and are finite.
- SB_i are *surplus* buffers and are finite.
- BB_i are *backlog* buffers and are infinite.
- The sizes of B_i and SB_i are control parameters.
- *Problem:* predicting the performance of this system.

Problem Statement

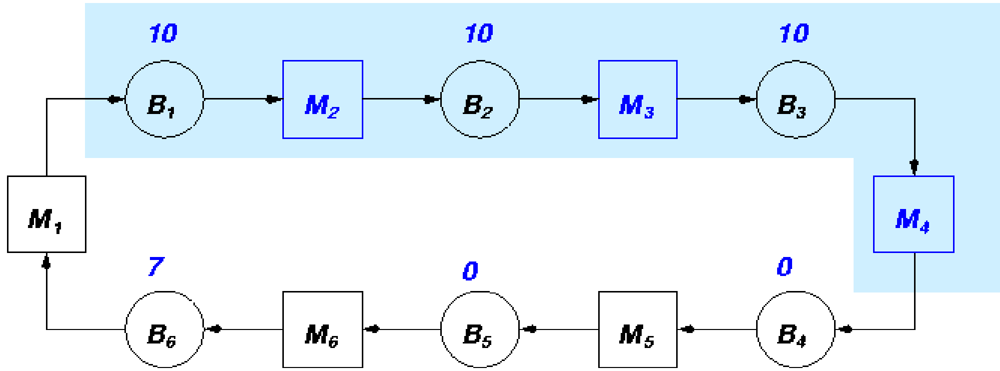


- Finite buffers $(0 \leq n_i(t) \leq N_i)$.
- Closed loop – fixed population $(\sum_i n_i(t) = N)$.
- Buzacott model (deterministic processing time; geometric up and down times). Repair probability = r_i failure probability = p_i.
- *Goal:* calculate production rate and inventory distribution.

- *Desire:* a decomposition method similar to that for the line, which can be extended to multiple-loop systems.
- *Difficulty:* the correlation among buffer levels.
  - ⋆ A method exists (Frein, Commault, and Dallery, 1996) but is only accurate for large 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.

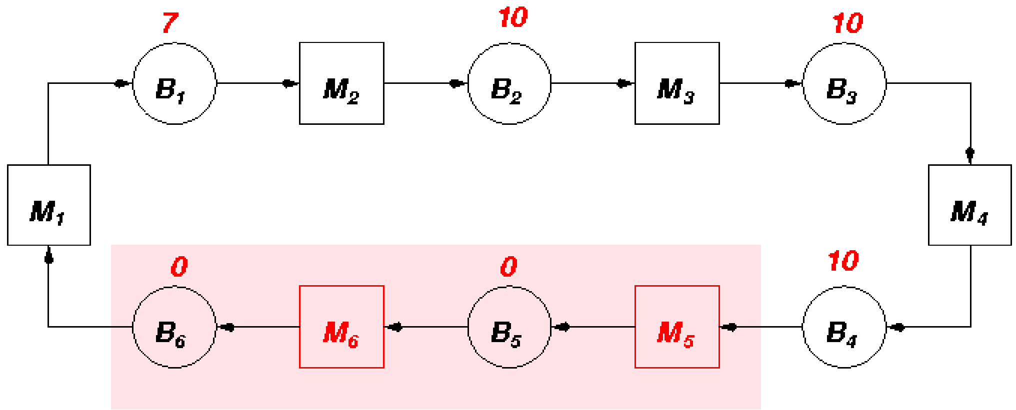


- All buffer sizes are 10.
- Population is 37.
- If M_4 stays down for a long time, it will block M_1.
- Therefore M_4 is in the range of blocking of M_1.
- Similarly, M_2 and M_3 are in the range of blocking of M_1.

Loops

Ranges

Range of starvation

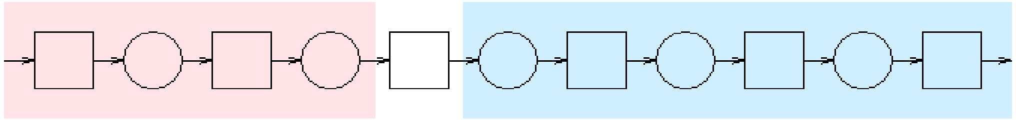


- If M_5 stays down for a long time, it will starve M_1.
- Therefore M_5 is in the range of starvation of M_1.
- Similarly, M_6 is in the range of starvation of M_1.

Loops

Ranges

Line

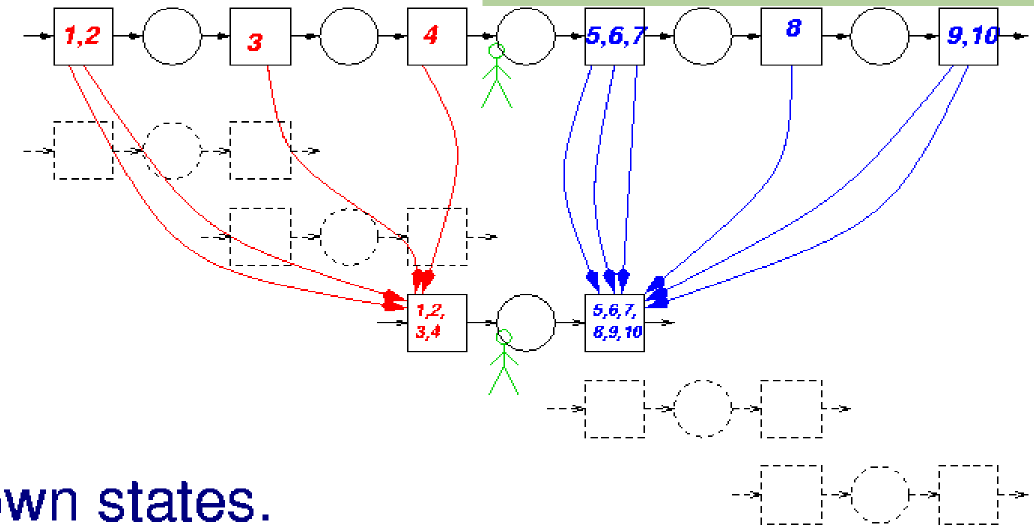


- The range of blocking of a machine in a line is the entire downstream part of the line.
- The range of starvation of a machine in a line is the entire upstream part of the line.

Loops

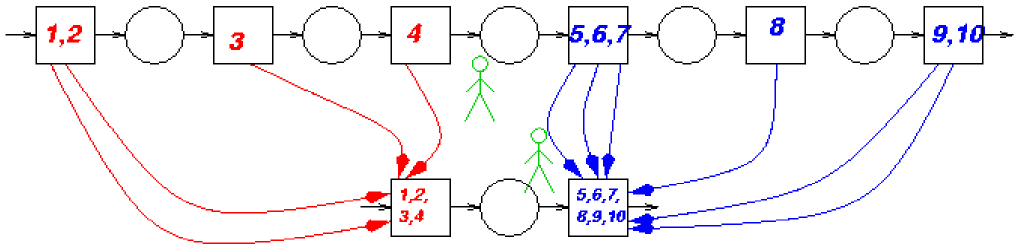
Tolio line decomposition

Tolio and Matta, 1998



- Multiple down states.
- Needed because original decomposition does not carry enough information.
- *However,* more than two parameters per machine are needed.

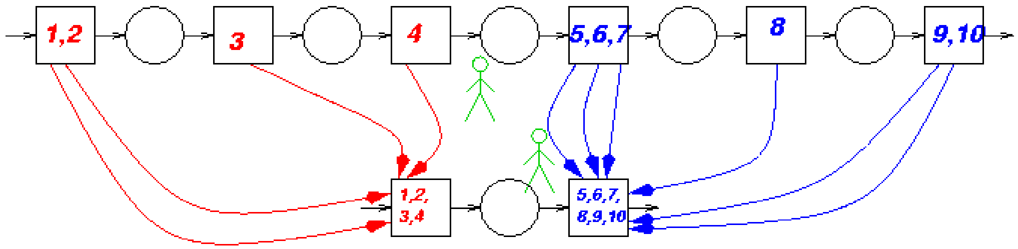
Loops



- 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.

Loops

Tolio line decomposition

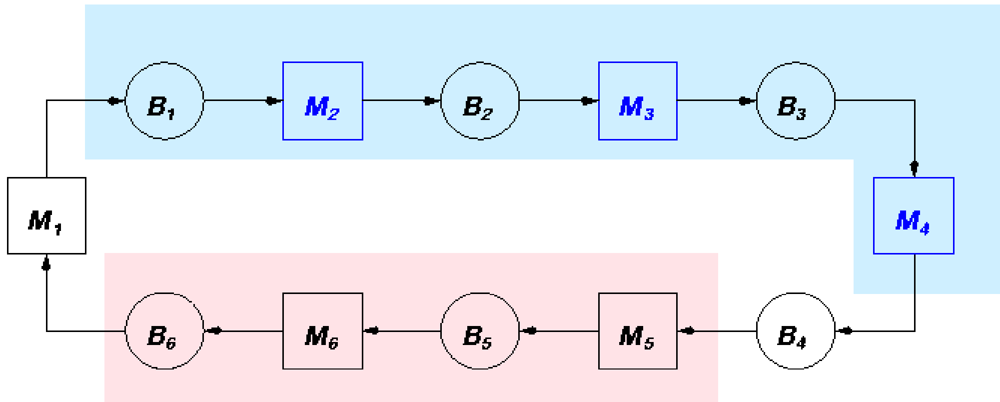


- The downstream failure modes appear to the observer after propagation through *blockage* .
- The upstream failure modes appear to the observer after propagation through *starvation* .
- The two-machine lines are more complex that in earlier decompositions but the decomposition equations are simpler.

Loop decomposition

- A set of decomposition equations are formulated.
- They are solved by an algorithm similar to the that of the original.
- The results are a little more accurate than earlier methods for lines.
- This decomposition can be extended to loops because we can include range information.

Loop decomposition



- Use the Tolio decomposition, but adjust the ranges of blocking and starvation accordingly.

- 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%.

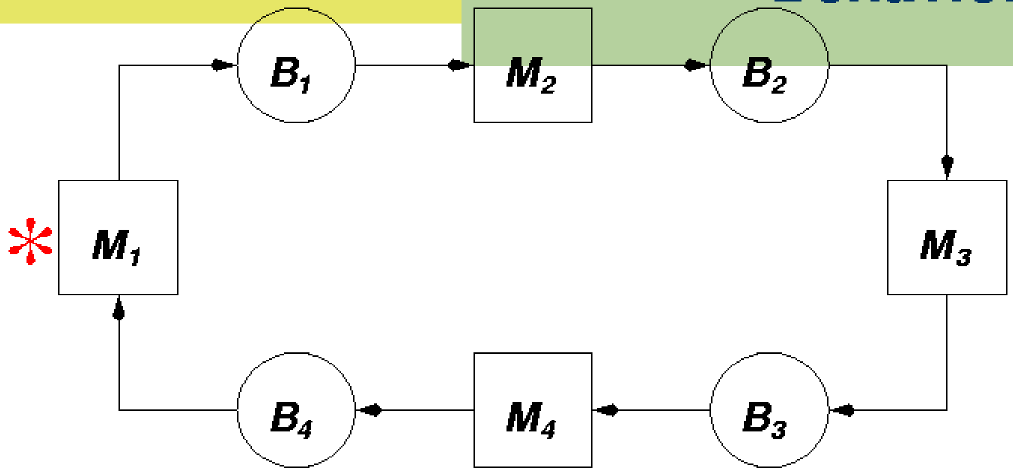
Other algorithm attributes

- Convergence reliability: almost always.
- Speed: execution time increases rapidly with loop size.
- Maximum size system: growing each time we implement.

Loop decomposition

Numerical results

Behavior

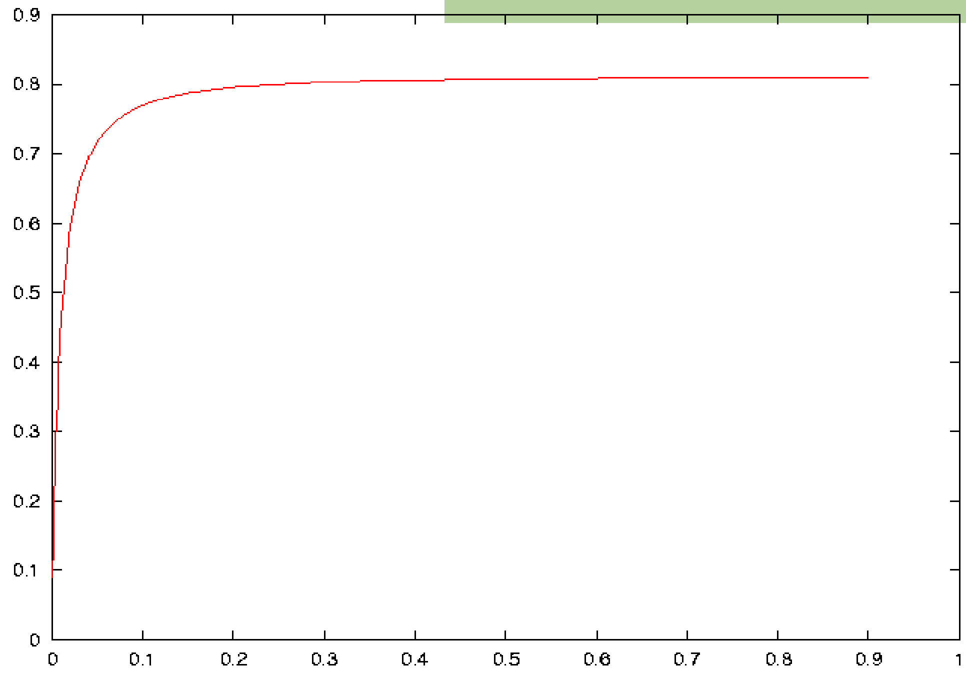


- All buffer sizes 10. Population 15. Identical machines except for M_1.
- Observe average buffer levels and production rate as a function of r_1.

Loop decomposition

Numerical results

Behavior

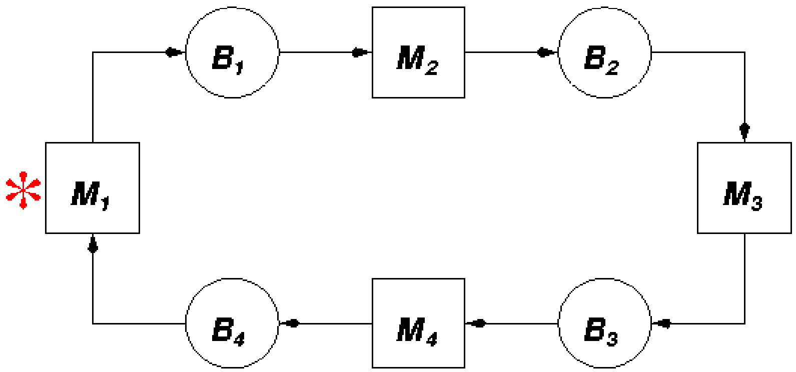


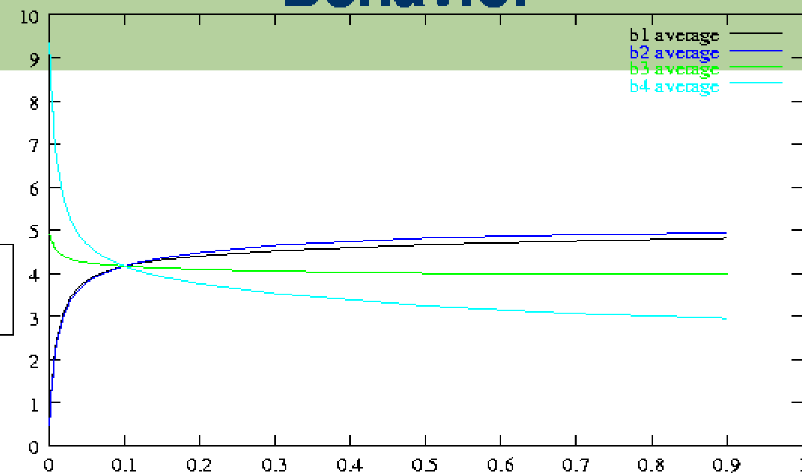
- Usual saturating graph.

Loop decomposition

Numerical results

Behavior



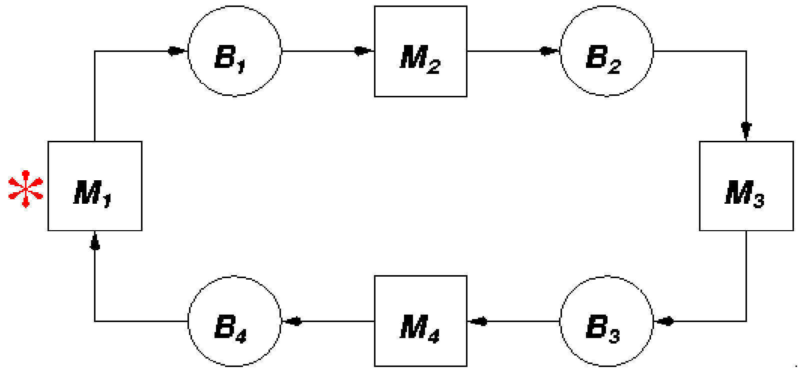


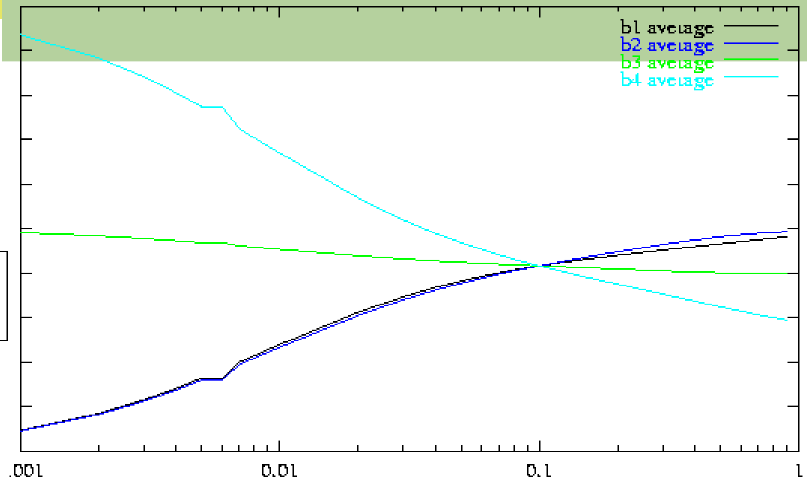
- When r_1 is small, M_1 is a bottleneck, so B_4 holds 10 parts, B_3 holds 5 parts, and the others are empty.
- As r_1 increases, material is more evenly distributed. When $r_1 = 0.1$, the network is totally symmetrical.

Loop decomposition

Numerical results

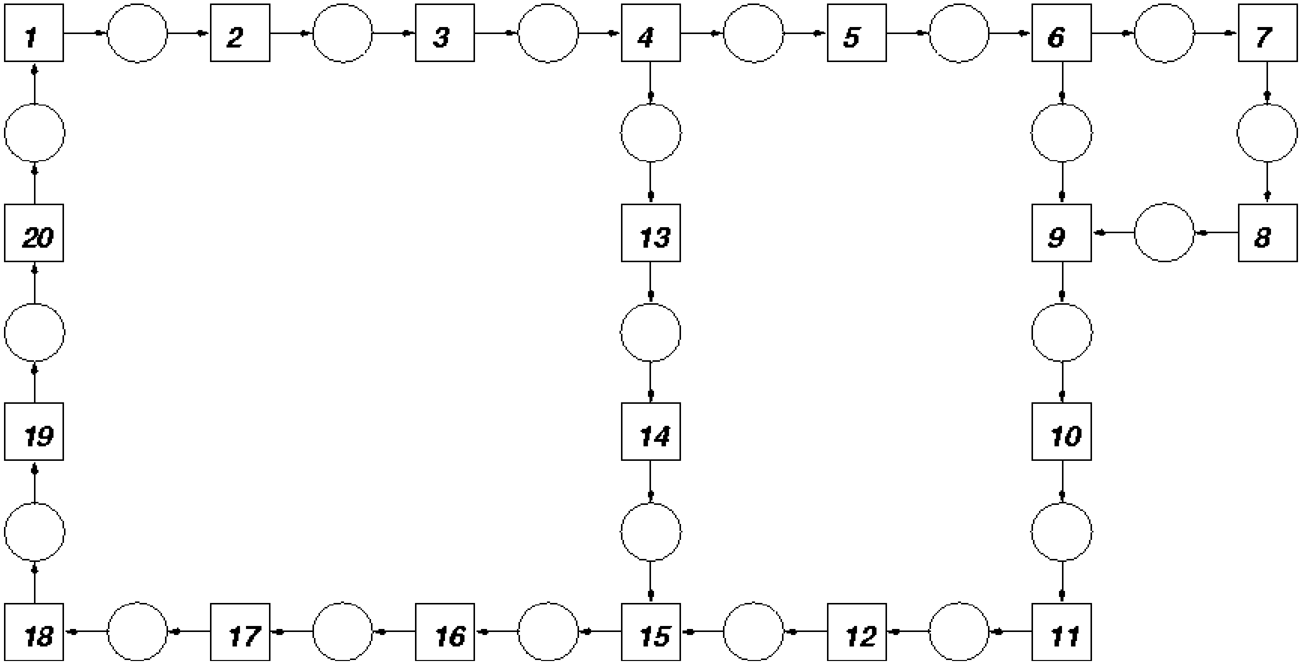
Behavior





- 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.

Multiple-loop systems



- Ranges of blocking and starvation are more complex, but possible to determine.
- Algorithm is an extension of single-loop algorithm.

Multiple-loop systems

Important features

- One *invariant* for each independent loop.
- Ranges of starvation and blockage need not be contiguous.

Multiple-loop systems

Range of Blocking

Current approach

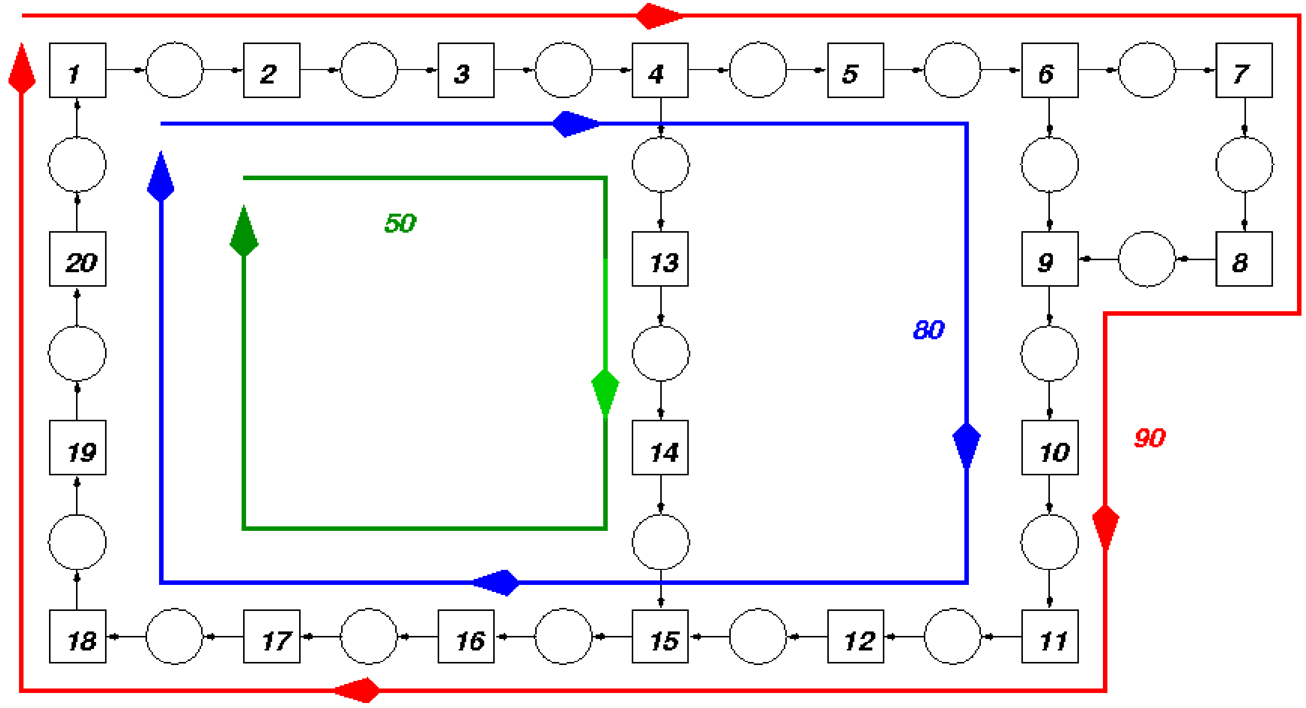
- To calculate the *range* of blocking for each machine M_i ...
  - ⋆ *(the set of machines* M_j *that could block* M_i *if* M_j *were down for a very long time)*
- we first determine the *domain* of blocking for each M_j
  - ⋆ *(the set of machines that* M_j *could block if it were down for a very long time)*
- and then transpose the table.

Similarly for the range of starvation.

Multiple-loop systems

Range of Blocking

Example

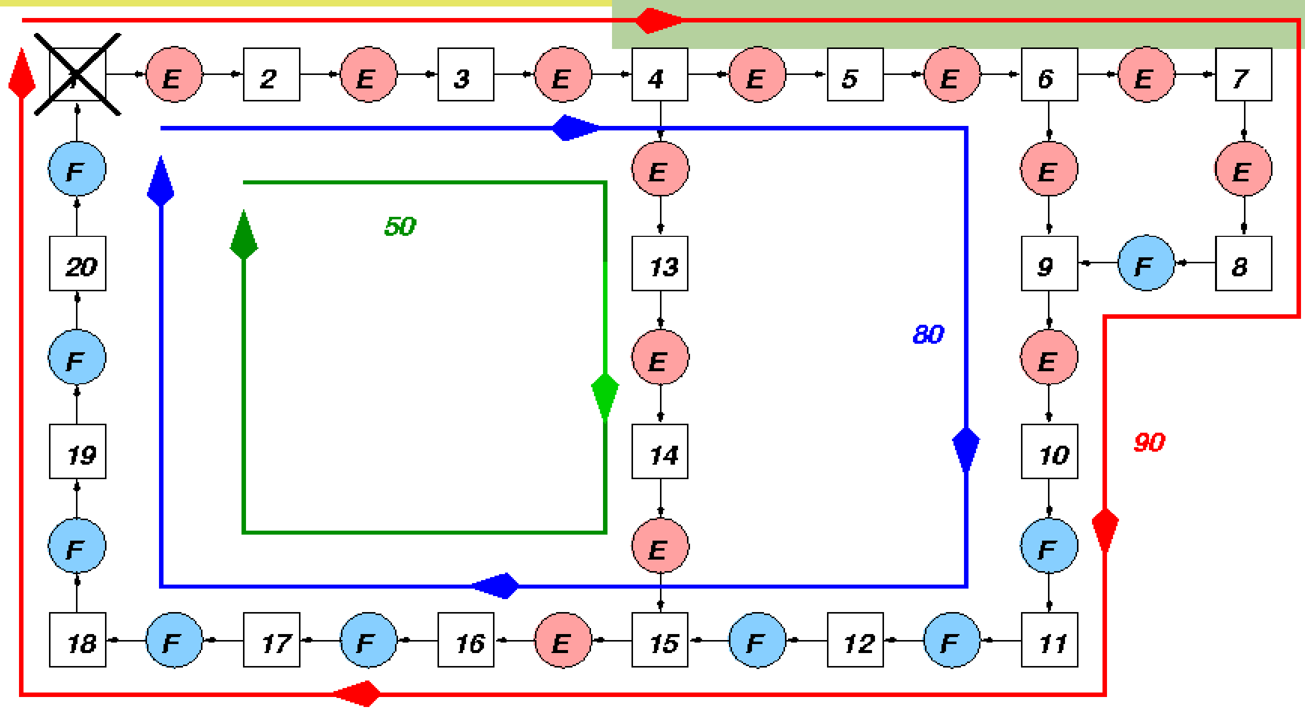


- All buffer sizes = 10.

Multiple-loop systems

Range of Blocking

Example

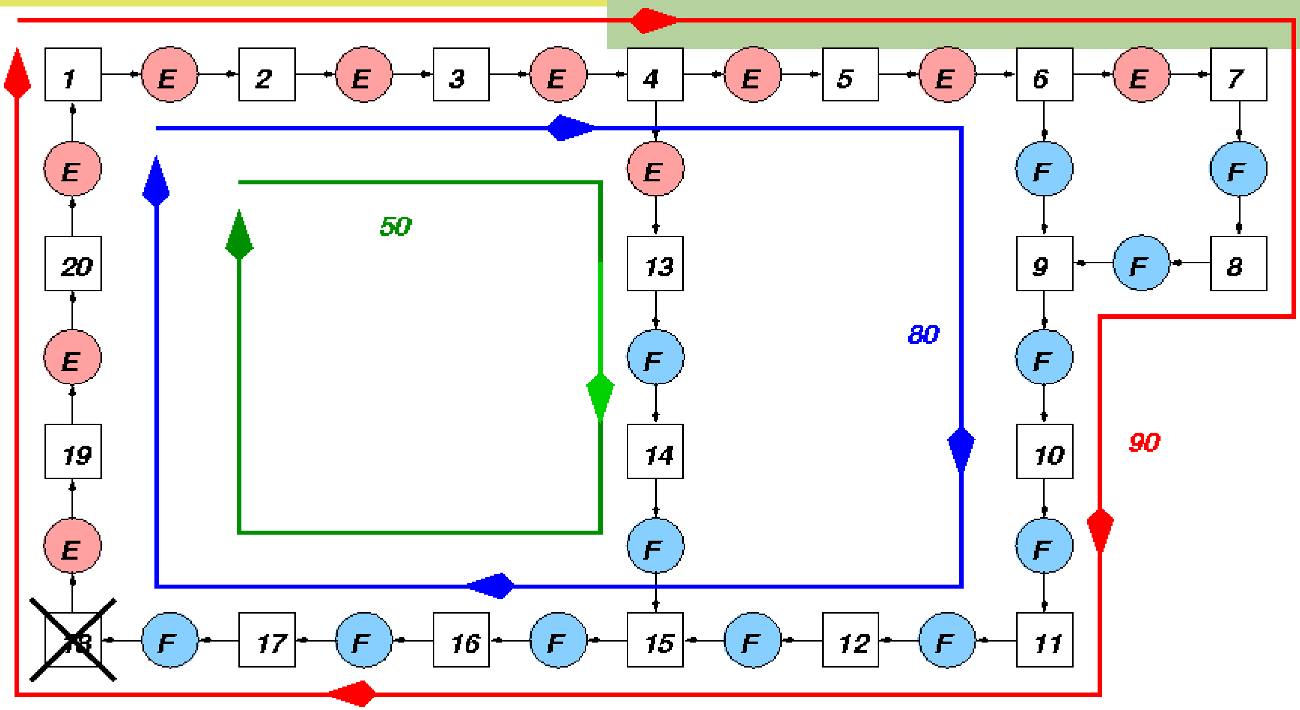


- Machine 1 fails for a long time.
- Buffers in the *domains* of blocking and starvation indicated.

Multiple-loop systems

Range of Blocking

Example

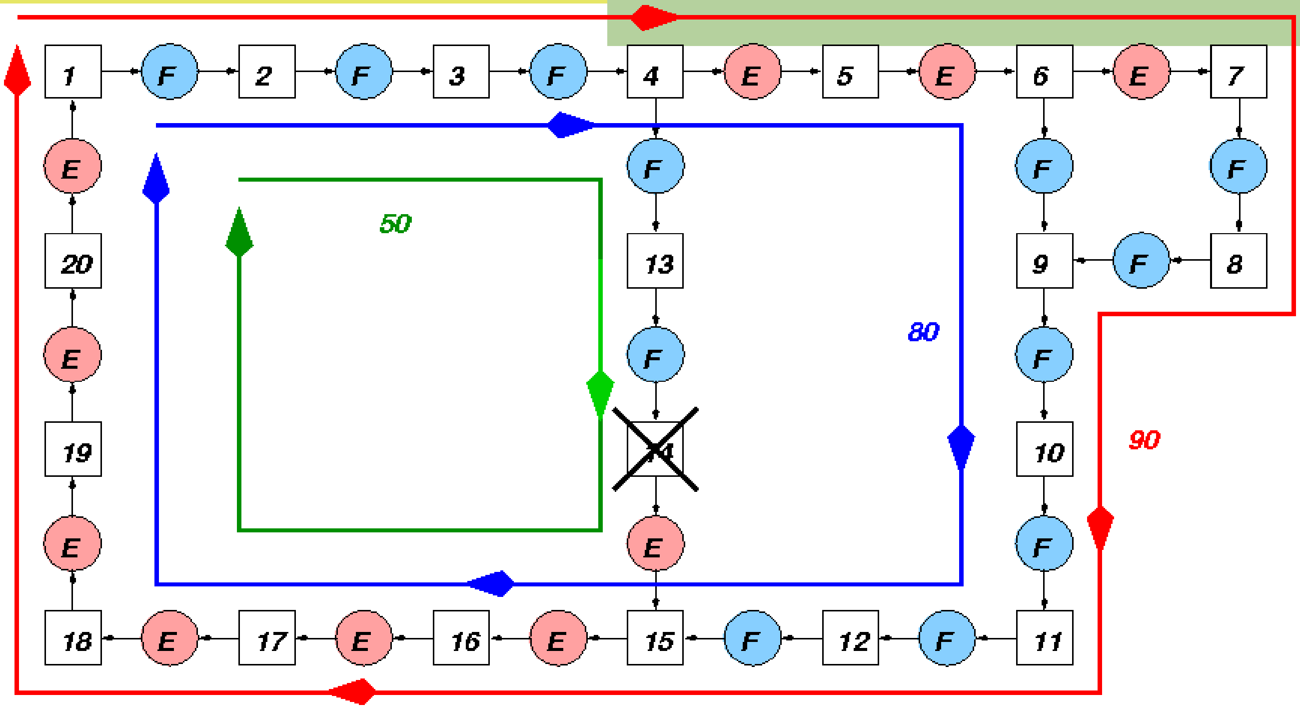


- Machine 18 fails for a long time.
- Buffers in the *domains* of blocking and starvation indicated.

Multiple-loop systems

Range of Blocking

Example



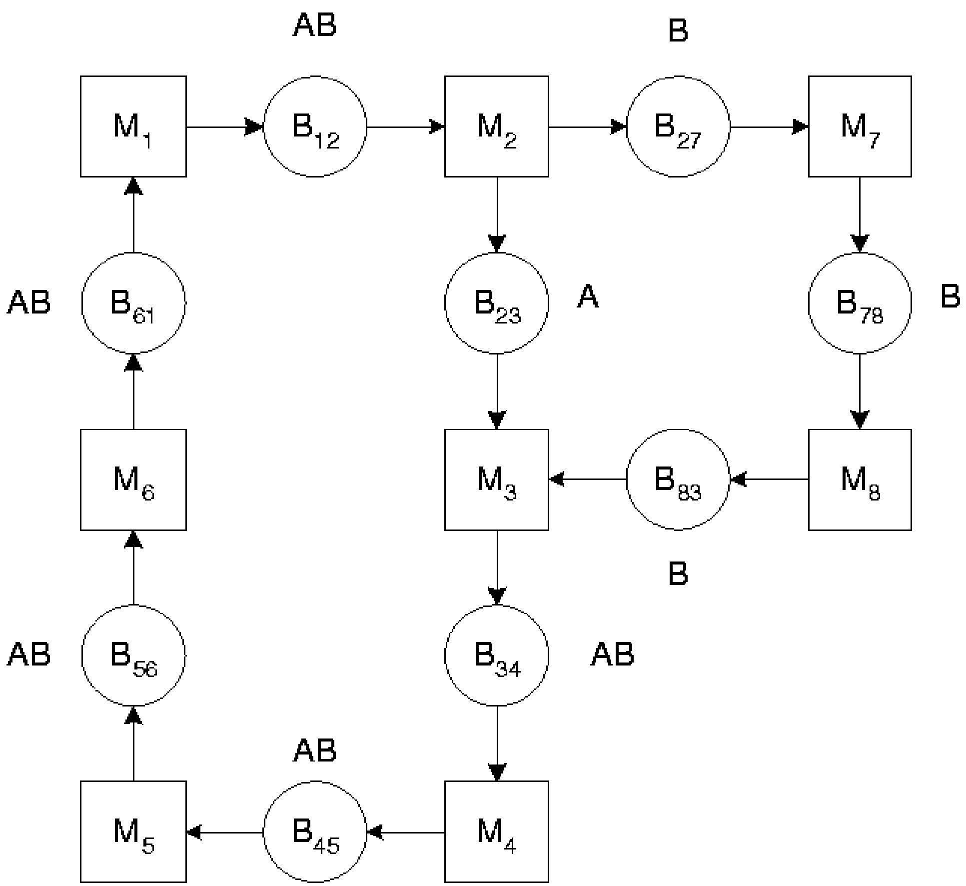
- Machine 14 fails for a long time.
- Buffers in the *domains* of blocking and starvation indicated.

Levantesi PhD thesis, 2001

- Decomposition equations are identical to those of Tolio and Matta (1998).
- Algorithm:
  - ⋆ Phase 1: Determine ranges of blocking and starvation.
  - ⋆ Phase 2: Tolio and Matta DDX-type algorithm

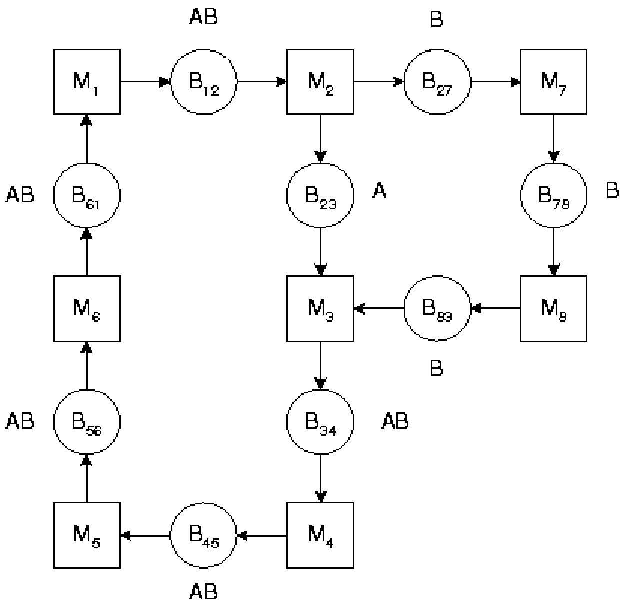
Multiple-loop systems

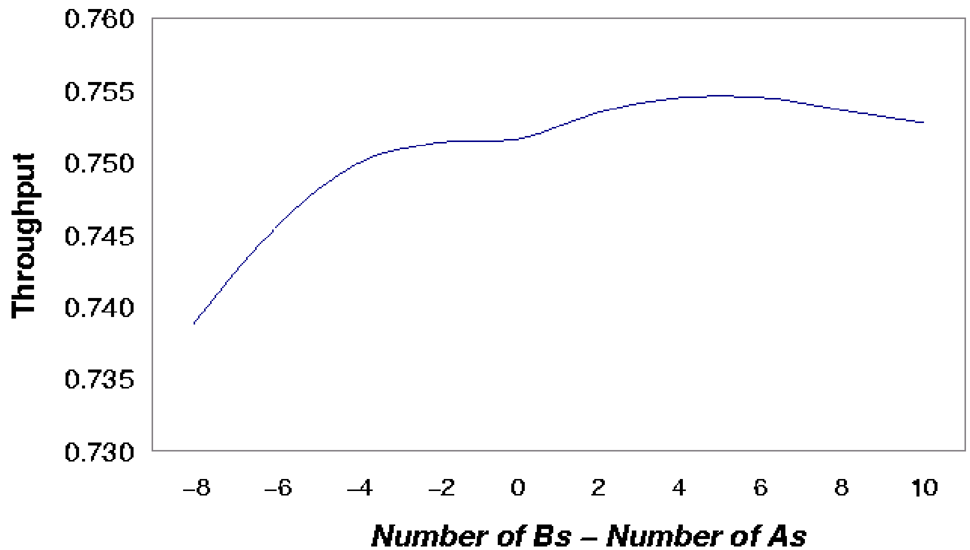
Example



Multiple-loop systems

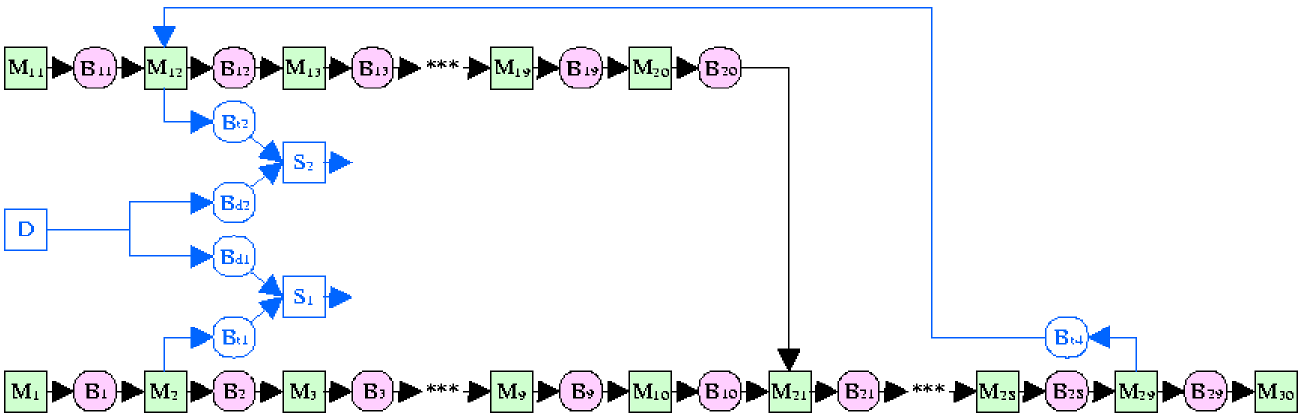
Example





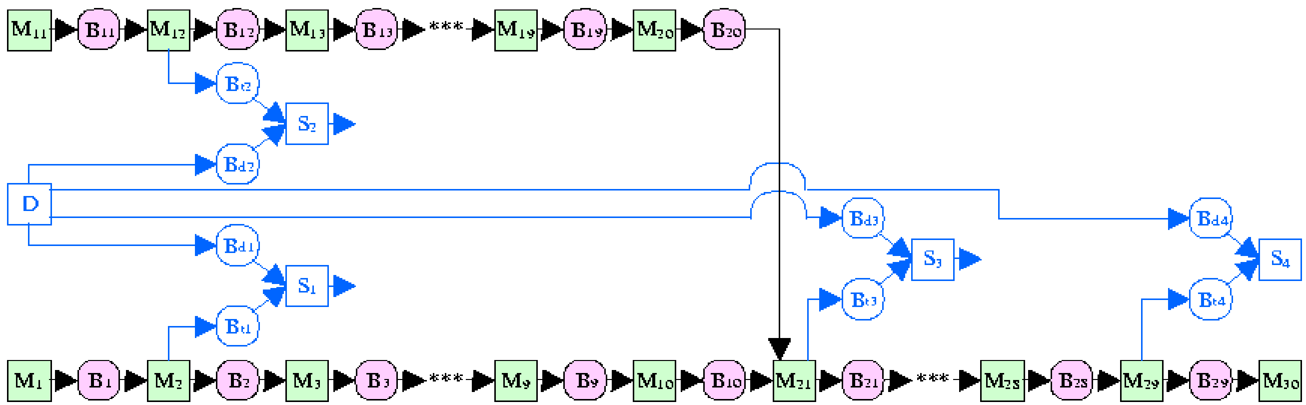
Multiple-loop systems

Current experiments



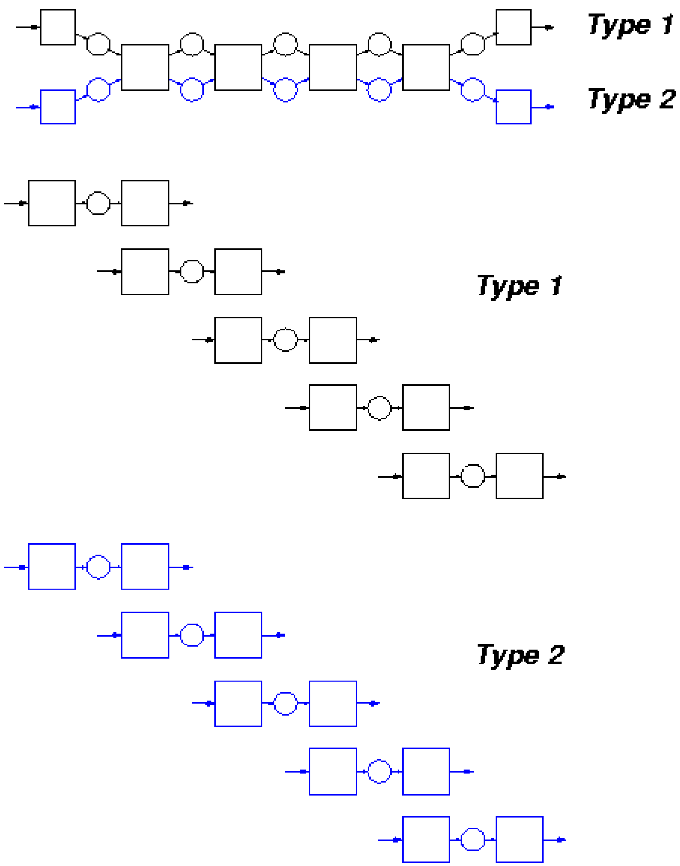
Multiple-loop systems

Current experiments

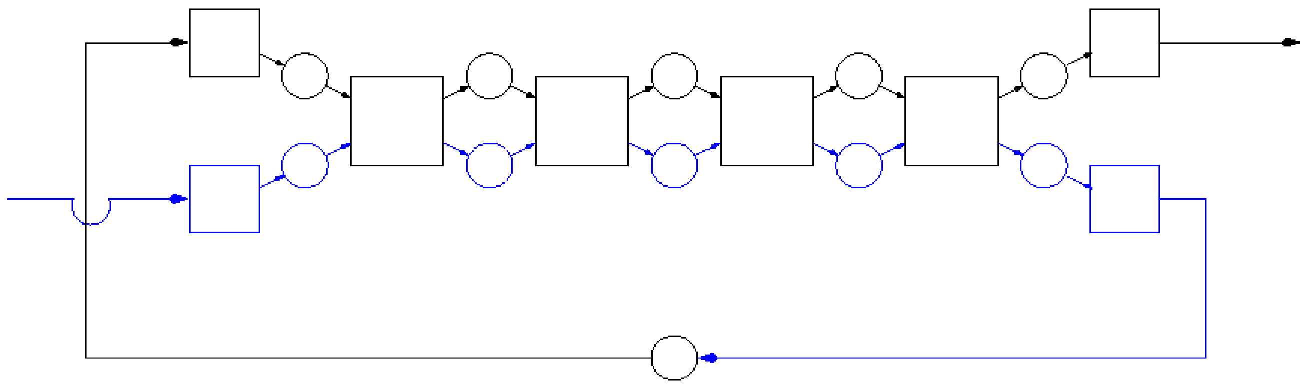


- Algorithm has worked successfully for this system.

Multiple-Part-Type Decomposition



- We have made some progress with two-part-type systems with strict priority, controlled by the finite buffer policy.



We plan to study reentrant systems as multiple-part-type systems where when one part leaves, it retruns as another part type.

Conclusions and Future Research

Promising approach to production system design and production system control policy design.

- Comparison with simulation
- Numerical experimentation
- Optimization of buffer sizes
- Optimization of control structure
- Multiple part types — with token-based control policies
- Real world implementation experiments