Analysis, Control, and Design of Stochastic Flow Systems with Limited Storage

Stanley B. Gershwin

Department of Mechanical Engineering
Massachusetts Institute of Technology

International Annual Conference of the German OR Society 2012
Leibniz Universität Hannover, Germany

September 4–7, 2012

Outline

Introduction
Real-Time Scheduling

- *Goal of real-time scheduling:* to manage a complex system to respond to random events so its performance is satisfactory
- Factory performance measures include
  - *(maximizing)* production or meeting production targets
  - *(minimizing)* inventory, buffer space, and other costs
  - *(maximizing)* service rate and *(minimizing)* lateness of late parts
- Factory complexity is due to
  - the large number of items
  - the dynamic interactions among items
  - randomness

Introduction

Real-Time Scheduling

- *Goals of the talk:*
  - to describe research
  - to suggest a view of real-time factory scheduling
  - to introduce a set of related scheduling policies
  - to find opportunities for implementation

Introduction

Real-Time Scheduling

Models and modeling

- We use "model" in the traditional engineering/scientific sense of a representation of a real system to be used for predicting something about that system.
- Models may be mathematical, computational, simulation, thought experiments, etc. Here we focus on mathematical and computational models.

Introduction

Real-Time Scheduling

Models and modeling

- Complex real-world problems are solved by
  1. creating a model: a simplified representation of the problem to be solved;
  2. analyzing the model;
  3. adapting the analysis to the real situation.
- Modeling involves ignoring certain features and focusing on others.
- Modeling is an art: *human creativity is the only way to decide what to leave out of the model and what to keep in.*

Introduction

Real-Time Scheduling

Approaches to real-time scheduling:

- Avoid modeling altogether. Invent a reasonable heuristic, and determine its its effectiveness by proving mathematically that it works for simplified models or simulating it on a realistic model of the factory.
- Modeling:
  - If it were possible to analyze it, the real-time scheduling of a complex system would be modeled as a large scale, mixed integer and continuous variable, stochastic dynamic programing problem.
  - But it is not, so we have to simplify.

Introduction

Real-Time Scheduling

Approaches to real-time scheduling: Modeling

- Leave some features out of the formulation and the implementation.
- Leave some features out of the formulation and adapt to them in the implementation. In particular, two main approaches to modeling are:
  - Focus on the large scale and leave out the randomness. Formulate the problem as a large mixed integer linear program. *Adaptation:* solve it, and then solve it again each time an important random event occurs.
  - Focus on the randomness and leave out the large scale. Solve a simple stochastic dynamic programming problem. *Adaptation:* implement the solution in different parts of the system and coordinate them.

Introduction

Real-Time Scheduling

Comments on approaches to real-time scheduling:

- Avoiding modeling is often postponing modeling (since we will have to simulate). Also, it provides no systematic way of inventing new policies for related but different cases.
- Focusing on size in the model and treating randomness in the implementation means representing the system as a large scale deterministic MILP and plugging it into a black-box software package. Responding to random events can create instability. Also this does not help managers and engineers develop intuition on system behavior.
- I prefer the second approach: Focus on the randomness in the model and deal with the large scale in the implementation.

Introduction

Real-Time Scheduling

Advantages of starting with small stochastic models:

- It provides a possible path for scheduling new kinds of systems: modify the model in an appropriate way, and solve that.
- It provides intuition.

Disadvantages of starting with a large deterministic MILP problem:

- It provides very little intuition.
- Re-solving the deterministic problem after random events can lead to instability and unnecessary schedule changes.

Introduction

Real-Time Scheduling

Disadvantage of starting with small stochastic models:

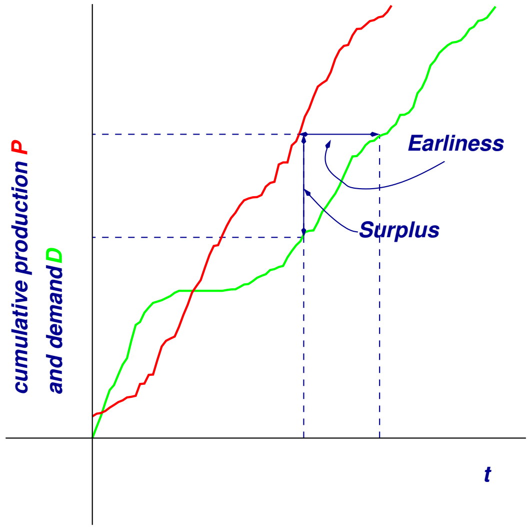
- Developing a model for a new problem is a research project.

Advantage of starting with a large deterministic MILP problem:

- Developing a model for a new problem is relatively easy.

Development of the Control Point Policy

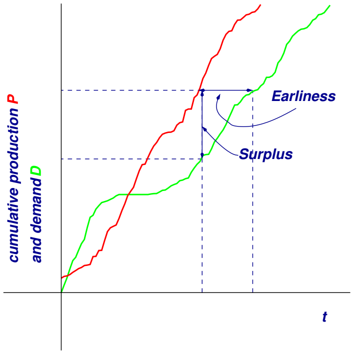
Objective



Keep P as close as possible to D.

Development of the Control Point Policy

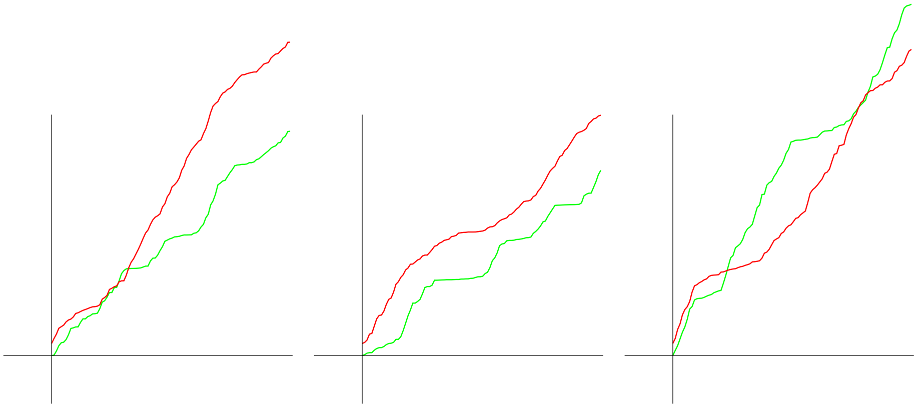
Objective



- Surplus-based scheduling tries to keep the surplus and backlog close to 0;
- Time-based scheduling tries to keep the earliness and lateness close to 0;

Development of the Control Point Policy

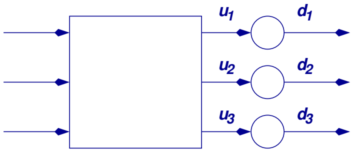
Objective



Multiple part types

Development of the Control Point Policy

Original Formulation



- Single machine
- Multiple part types
- Perfectly flexible: setup (changeover) times are 0.
- Machine is unreliable; exponentially distributed up- and down-times.
- Time and amount of material produced are represented as continuous.

$$u_i = \frac{dP_i}{dt} \qquad d_i = \frac{dD_i}{dt}$$

Development of the Control Point Policy

Dynamic Programming Problem

- Surplus dynamics:
$$\frac{dx_i}{dt} = u_i - d_i$$
- Let $\alpha(t)$ be the state of the machine at time t. $\alpha(t) = 1$ means that the machine is operational; $\alpha(t) = 0$ means that the machine is down.

State variables: the surpluses x_i and the machine state α.

Control variables: the production rates u_i.

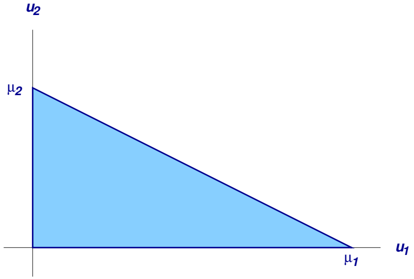
Development of the Control Point Policy

Dynamic Programming Problem

Production rate constraints: $u_i \geq 0$; $\sum_i \tau_i u_i \leq \alpha$ where τ_i is the time to make 1 unit of type i.

For two part types:

When $\alpha = 1$ this is ...



where $\mu_i = 1/\tau_i$.

When $\alpha = 0, u = 0$.

Development of the Control Point Policy

Dynamic Programming Problem

Let $g(x)$ be a measure of how far the surplus is from 0.

$$g(0) = 0$$

$$\lim_{\|x\| \to \infty} g(x) = \infty$$

$g(x)$ can assign different costs to different part types. Inventory or backlog of some parts may be more expensive than that of others.

Development of the Control Point Policy

Dynamic Programming Problem

Formulation:

$$\min_{u(t), t_0 \leq t \leq T} J(x(t_0), \alpha(t_0), t_0) = E\left\{ \int_{t_0}^{T} g(x(s))ds \mid x(t_0), \alpha(t_0) \right\}$$

subject to
$$\frac{dx}{dt} = u - d$$

$$\mathbf{p}\left[\alpha(t+\delta t) = 1 \mid \alpha(t) = 0\right] = r\delta t; \quad \mathbf{p}\left[\alpha(t+\delta t) = 0 \mid \alpha(t) = 1\right] = p\delta t$$

$$u_i \geq 0 \qquad \sum_i \tau_i u_i \leq \alpha$$

$x(t_0), \alpha(t_0)$ specified

Development of the Control Point Policy

Dynamic Programming Problem

Solution: The optimal production rates $u(x, \alpha, t)$ are given by a *feedback policy.* They are determined by:

$$\min_{u(t)} \frac{\partial J}{\partial x} u$$

such that

$$\sum_i \tau_i u_i \leq \alpha$$

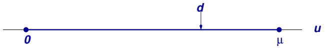
$$u_i \geq 0$$

Development of the Control Point Policy

Hedging Point Policy

Single part type case:

Constraint set: $0 \leq u \leq \mu$



Optimal policy for constant d: For some Z, u satisfies

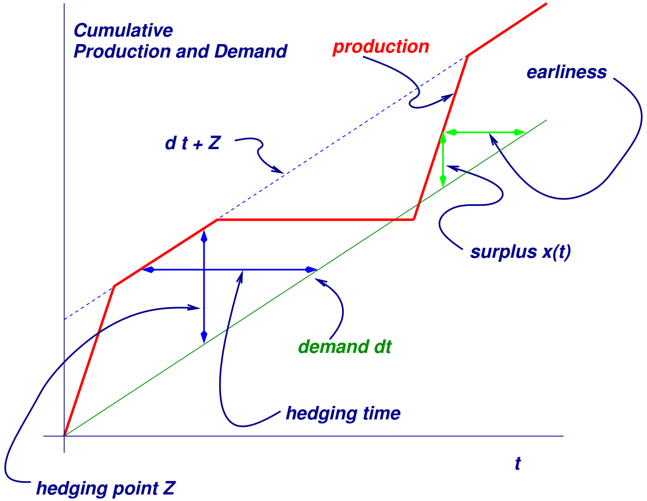
$$\left.\begin{aligned} u &= 0 \text{ if } x > Z \\ u &= d \text{ if } x = Z \\ u &= \mu \text{ if } x < Z \end{aligned}\right\} \quad \text{when } \alpha = 1.$$

$u = 0$ when $\alpha = 0$.

Z is the *hedging point.* The policy is the *hedging point policy.*

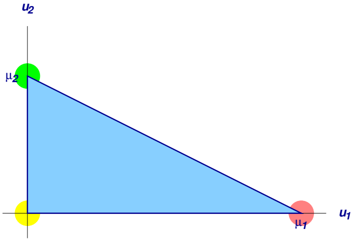
Development of the Control Point Policy

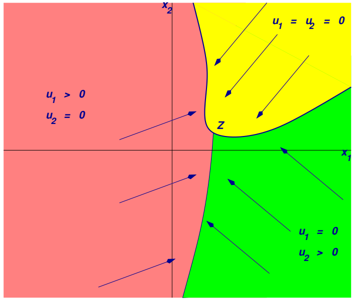
Hedging Point Policy



Development of the Control Point Policy
Hedging Point Policy

Two part type case: For two part types, the surplus space is divided into three regions because the production rate constraints form a triangle:

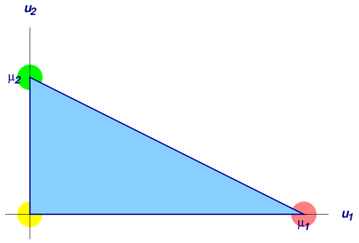


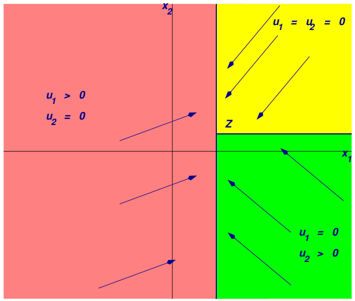


Development of the Control Point Policy

Priority Hedging Point Policy

Approximate Solution: The curved boundaries are replaced by straight lines parallel to the axes:





Priority: Here, when both part types are below their hedging points, make Type 1. This is because of the cost function $g(x_1, x_2)$.

Development of the Control Point Policy

Priority Hedging Point Policy

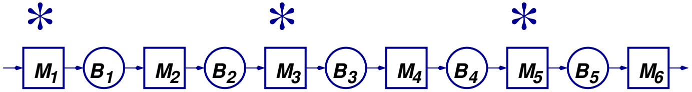
Approximate Solution: This approximation leads to the *Priority Hedging Point (PHP)* policy.

0. Before executing the policy, rank order the products.
1. Produce the highest ranking product, until its surplus reaches its hedging point. (The others fall behind.)
2. Keep the highest ranking product at its hedging point. Devote all remaining capacity to the second highest ranking product, until it reaches its hedging point. (The others fall further behind.)
3. Keep the two highest ranking products at their hedging points. Devote all remaining capacity to the third highest ranking product, until it reaches its hedging point. (The others fall still further behind.)
4. etc.

This statement can be simplified to: *Produce the highest ranking part whose surplus is below its hedging point* .

Development of the Control Point Policy

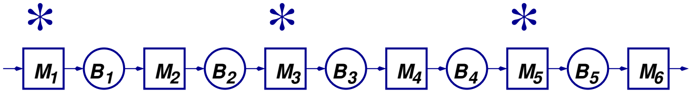
The Control Point Policy is the extension of the (Priority) Hedging Point Policy to systems with multiple stages.



It may not be necessary to control every stage. Here, we control machines marked with $*$.

Development of the Control Point Policy

Surplus-based implementation: When the machine at control point j is available, consider parts whose surplus $x_i^j < Z_i^j$. Operate on the *highest ranking* part of that set.

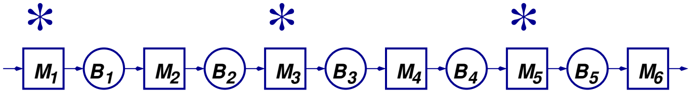


Other machines may be scheduled in any way that does not cause unnecessary idleness (for example, FIFO).

Problem: how to determine optimal hedging points Z_i^j and optimal buffer sizes.

Development of the Control Point Policy

Time-based implementation: When the machine at control point j is available, consider parts whose *earliness* is less than their hedging times. Operate on the *highest ranking* part of that set.



Other machines may be scheduled in any way that does not cause unnecessary idleness (for example, FIFO).

Problem: how to determine optimal hedging times and optimal buffer sizes.

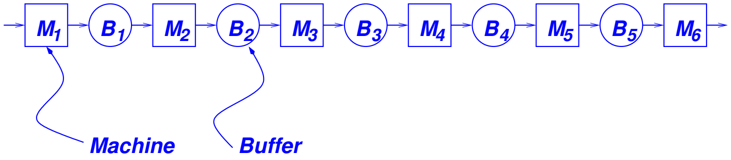
Development of the Control Point Policy

Problem: how to determine optimal hedging points Z_i^j or hedging times and optimal buffer sizes.

We can experiment with simulation, but an analytical method may be faster. This is discussed below.

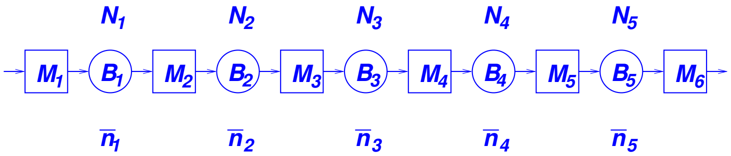
Another problem: how to determine which machines should be control points. This is *not* discussed below.

Evaluation of Finite-Buffer Systems



- Randomness due to unreliable machines
- Finite buffers
- Propagation of failures through starvation and blockage

Evaluation of Finite-Buffer Systems



Performance measures:

- Production rate P.
- Average in-process inventory $\bar{n}_i$

Evaluation of Finite-Buffer Systems

Methodology

Models:

- Lines are modeled as Markov processes.
- Material and time can be discrete or continuous.
- Machines fail and are repaired at random times.
- *Basic* models have geometric or exponential distributions of up- and down-time.
- Notation:
  - p=1/MTTF is failure probability or failure rate.
  - r=1/MTTR is repair probability or repair rate.
- *Basic* models have a single part type.
- More general models are now available.

Evaluation of Finite-Buffer Systems

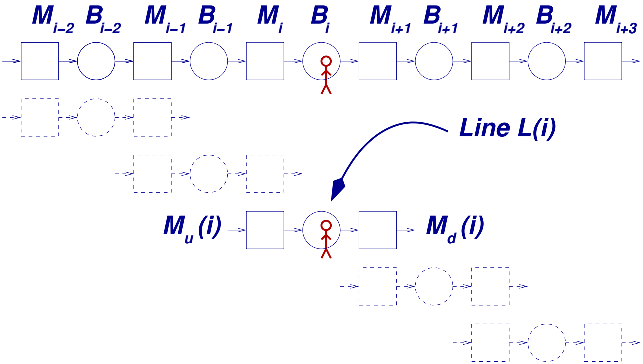
Methodology

Performance evaluation:

- Two-machine lines are evaluated analytically, exactly. We obtain the steady-state probability distribution and use it to determine the average production rate and buffer level.
- *Short lines can be evaluated numerically, exactly.*
- Long lines are evaluated analytically, approximately.

Evaluation of Finite-Buffer Systems

Methodology



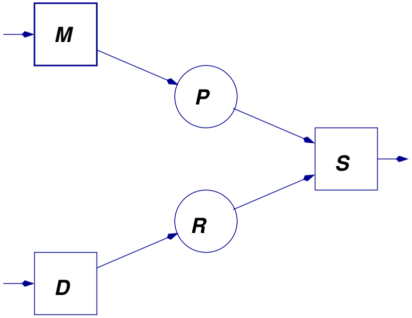
- Long lines are approximately evaluated by *decomposition.*
- Extension to loop-free assembly/disassembly systems is easy.

Synthesis

Consider the three-machine assembly system:

Assume

- M is the same as the single machine controlled by the hedging point policy.
- D generates parts at a constant rate d.
- S is perfectly reliable and infinitely fast.
- P *(Production)* is a buffer of size Z.
- R *(Requirements)* is an infinite buffer.



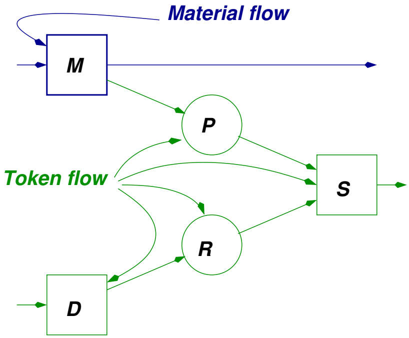
R and P cannot both have non-zero contents because of S. Define x such that x^+ is the contents of P and x^- is the contents of R. *Then (x, α) of this system is isomorphic to (x, α) of the single machine controlled by the hedging point policy.*

Synthesis

A Token-Based Control System

We change the picture slightly. Now, we can view D, R, S, and P to be a *control system* for M.

- When M performs an operation, it sends a part downstream and a token to P.
- When a demand arrives, D sends a token to R.
- When P and R are both non-empty, S removes the same amount from both so that only one is non-empty.
- M cannot operate if it is down, if the part is blocked downstream, or if P is full.

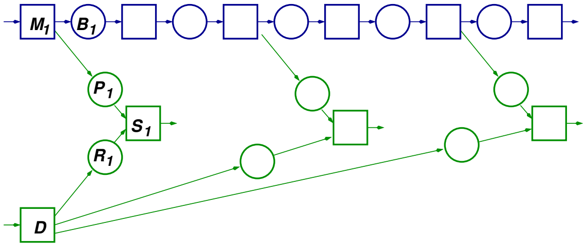


This system keeps M producing parts at an average rate d, and prevents the production of parts from getting too far ahead of demand.

Synthesis

A Token-Based Control System

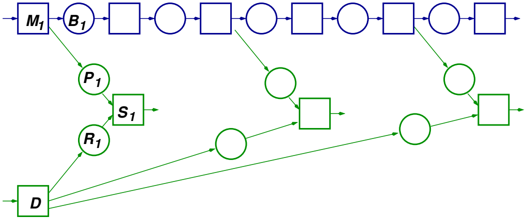
Now, we install the control system at all the control points. This system keeps all the controlled machines producing parts at an average rate d, it prevents the production of parts from getting too far ahead of demand, and the finite material buffers prevent each machine from getting too far ahead of its neighbors.



Synthesis

A Token-Based Control System

Token-based implementation: When a demand arrives, D sends a token simultaneously into each of the requirements buffers. When a controlled machine does an operation, it sends a part into the downstream material buffer and it sends a token into its production buffer.

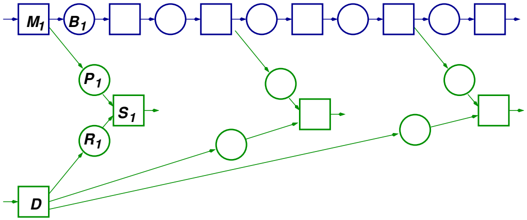


The machine can only operate if its downstream material buffer is not full *and* its production token buffer is not full. Tokens are removed from the production buffers by the synchronization machines when there are token in the corresponding requirements buffer.

Synthesis

A Token-Based Control System

We can evaluate this network because there has been recent work in performance analysis of multi-loop networks. Optimization should not be too difficult.



Optimizing the material and production token buffers can be used to get optimal buffer sizes, hedging points, and hedging times for the surplus- and time-based versions of the policy.

Conclusions

- Real-time factory scheduling can be treated as a control problem.
- This can reduce instability and provide better performance,
- The design of the scheduler can be integrated with the design of buffers.
- Three views of real-time scheduling: surplus-, time-, and token-based.

Future Research

- Model, analyze, and optimize multi-part type systems.
- Optimize the choice of control points.
- Get experience with simulation.
- Extend in other directions: reentrant flow systems, systems with alternate routes or rework and rejection, etc.
- Integrate with system design, especially machine choice.
- Implement in the real world.