MIT 2.852
Manufacturing Systems Analysis
Lecture 14-16

Line Optimization

Stanley B. Gershwin

Spring, 2007

Line Design

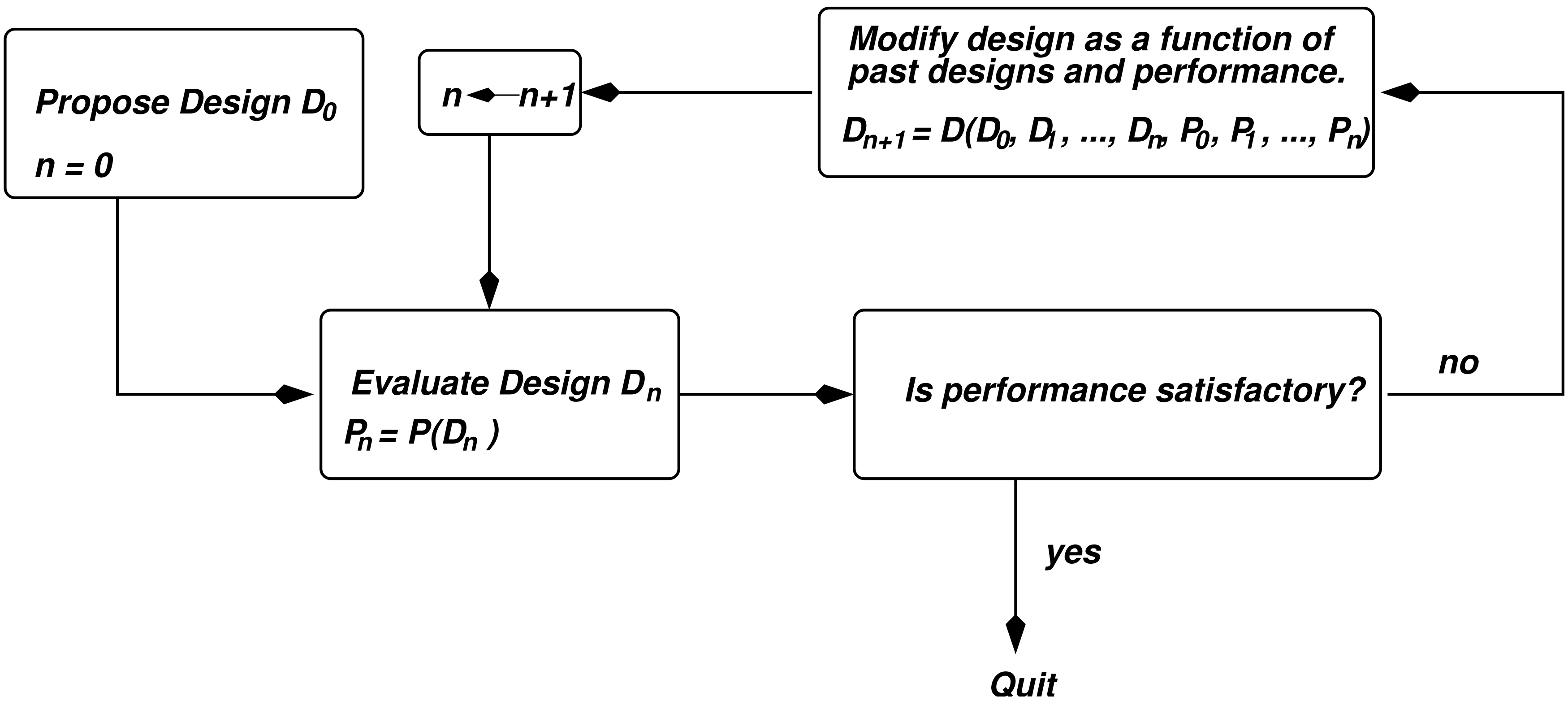
- Given a process, find the best set of machines and buffers on which it can be implemented.
- *Best:* least capital cost; least operating cost; least average inventory; greatest profit, etc.
- Constraints: minimal production rate, maximal stockout probability, maximal floor space, maximal inventory, etc..
- To be practical, computation time must be limited.
- Exact optimality is not necessary, especially since the parameters are not known perfectly.

Optimization

- Optimization may be performed in two ways:
  - ⋆ Analytical solution of optimality conditions; or
  - ⋆ Searching
- For most problems, searching is the only realistic possibility.
- For some problems, optimality cannot be achieved in a reasonable amount of time.

Optimization

Search



Typically, many designs are tested.

Optimization

Issues

- For this to be practical, total computation time must be limited. Therefore, we must control both *computation time per iteration* and *the number of iterations* .
- Computation time per iteration includes evaluation time and the time to determine the next design to be evaluated.
- The technical literature is generally focused on limiting the number of iterations by proposing designs efficiently.
- The number of iterations is also limited by choosing a reasonable termination criterion (ie, required accuracy).
- Reducing computation time per iteration is accomplished by
  - ⋆ using analytical models rather than simulations
  - ⋆ using coarser approximations in early iterations and more accurate evaluations later.

Problem Statement

$\boldsymbol{X}$ is a set of possible choices. $\boldsymbol{J}$ is a scalar function defined on $\boldsymbol{X}$. $\boldsymbol{h}$ and $\boldsymbol{g}$ are vector functions defined on $\boldsymbol{X}$.

Problem: Find $\boldsymbol{x} \in \boldsymbol{X}$ that satisfies

$\boldsymbol{J}(\boldsymbol{x})$ is maximized *(or minimized)* — the *objective*

subject to

$\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$ — *equality constraints*

$\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}$ — *inequality constraints*

Taxonomy

- static/dynamic
- deterministic/stochastic
- X set: continuous/discrete/mixed

(Extensions: multi-criteria optimization*, in which the set of all good compromises between different objectives are sought;* games*, in which there are multiple optimizers, each preferring different xs but none having complete control; etc.)*

Continuous Variables and Objective

$X = R^n$. J is a scalar function defined on R^n. $h(\in R^m)$ and $g(\in R^k)$ are vector functions defined on R^n.

Problem: Find $x \in R^n$ that satisfies

$J(x)$ is maximized *(or minimized)*

subject to

$h(x) = 0$

$g(x) \leq 0$

Continuous Variables and Objective

Unconstrained

One-dimensional search

Find t *such that* $f(t) = 0$.

- This is equivalent to

  Find t *to maximize (or minimize)* $F(t)$

  when $F(t)$ is differentiable, and $f(t) = dF(t)/dt$ is continuous.

- If $f(t)$ is differentiable, maximization or minimization depends on the sign of $d^2F(t)/dt^2$.

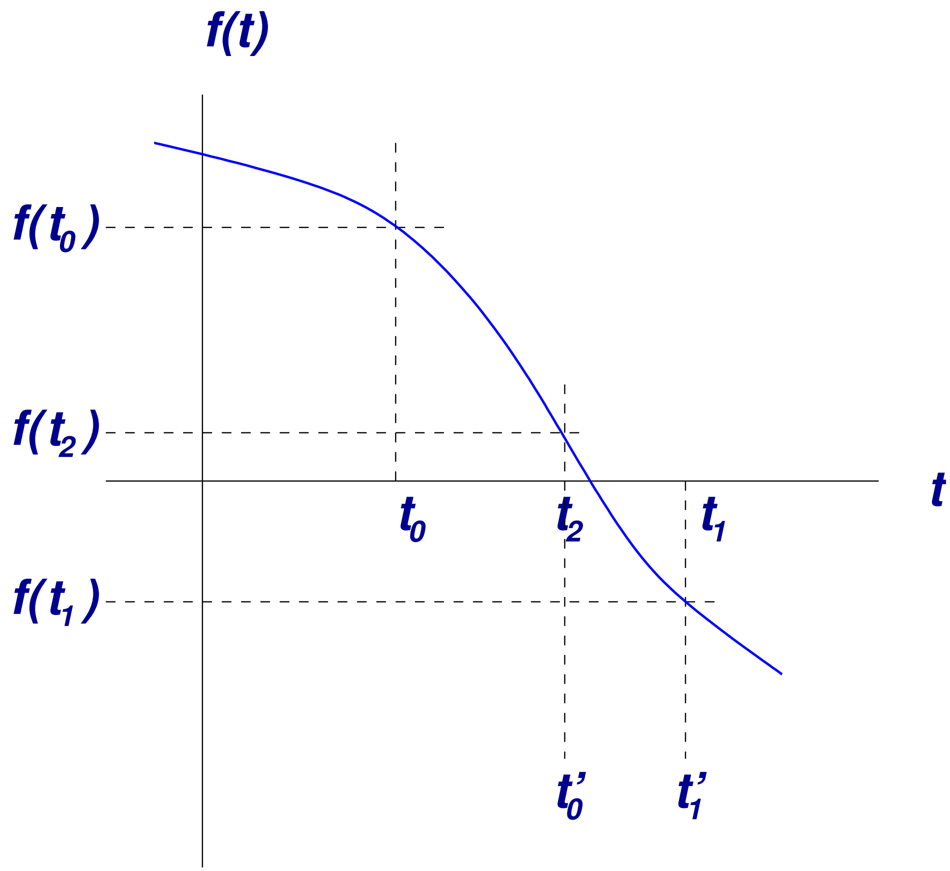
Continuous Variables and Objective

Unconstrained

One-dimensional search

Assume $f(t)$ is decreasing.

- *Binary search:* Guess t_0 and t_1 such that $f(t_0) > 0$ and $f(t_1) < 0$. Let $t_2 = (t_0 + t_1)/2$.
  - ⋆ If $f(t_2) < 0$, then repeat with $t_0' = t_0$ and $t_1' = t_2$.
  - ⋆ If $f(t_2) > 0$, then repeat with $t_0' = t_2$ and $t_1' = t_1$.



Continuous Variables and Objective

Unconstrained

One-dimensional search

Example:

$$f(t) = 4 - t^2$$

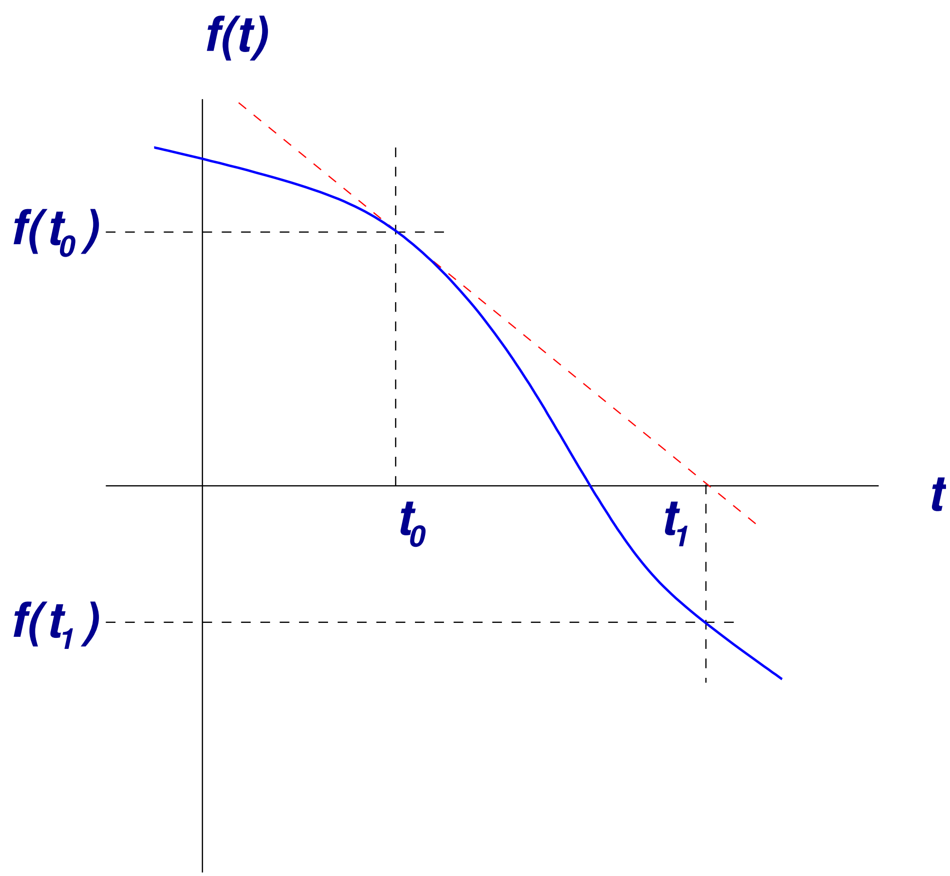
t_0	t_2	t_1
0	1.5	3
1.5	2.25	3
1.5	1.875	2.25
1.875	2.0625	2.25
1.875	1.96875	2.0625
1.96875	2.015625	2.0625
1.96875	1.9921875	2.015625
1.9921875	2.00390625	2.015625
1.9921875	1.998046875	2.00390625
1.998046875	2.0009765625	2.00390625
1.998046875	1.99951171875	2.0009765625
1.99951171875	2.000244140625	2.0009765625
1.99951171875	1.9998779296875	2.000244140625
1.9998779296875	2.00006103515625	2.000244140625
1.9998779296875	1.99996948242188	2.00006103515625
1.99996948242188	2.00001525878906	2.00006103515625
1.99996948242188	1.99999237060547	2.00001525878906
1.99999237060547	2.00000381469727	2.00001525878906
1.99999237060547	1.99999809265137	2.00000381469727
1.99999809265137	2.00000095367432	2.00000381469727

Continuous Variables and Objective

Unconstrained

One-dimensional search

- *Newton search, exact tangent:*
  - ⋆ Guess t_0. Calculate $df(t_0)/dt$.
  - ⋆ Choose t_1 so that $f(t_0) + (t_1 - t_0)\frac{df(t_0)}{dt} = 0$.
  - ⋆ Repeat with $t_0' = t_1$ until $|f(t_0')|$ is small enough.



Continuous Variables and Objective

Unconstrained

One-dimensional search

Example:

$$f(t) = 4 - t^2$$

t_0
3
2.166666666666667
2.006410256410260
2.000010240026 21
2.000000000002621
2

Continuous Variables and Objective

Unconstrained

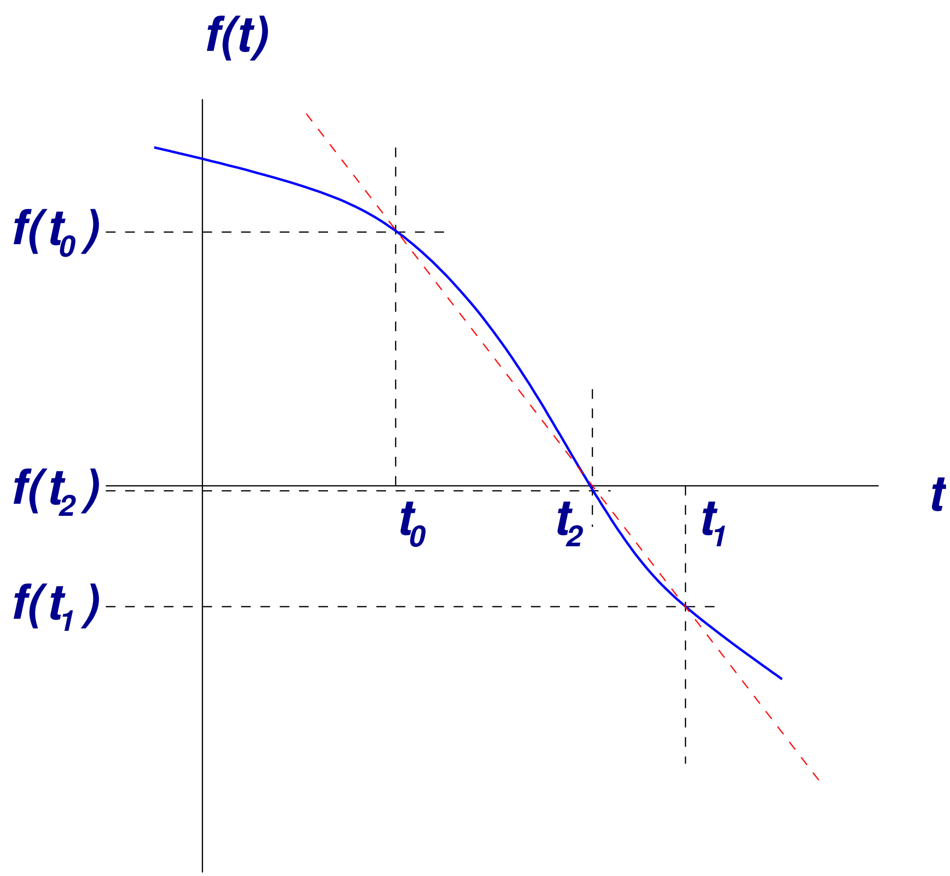
One-dimensional search

- *Newton search, approximate tangent:*

  ⋆ Guess t_0 and t_1. Calculate approximate slope $s = \frac{f(t_1) - f(t_0)}{t_1 - t_0}$.

  ⋆ Choose t_2 so that $f(t_0) + (t_2 - t_0)s = 0$.

  ⋆ Repeat with $t_0' = t_1$ and $t_1' = t_2$ until $|f(t_0')|$ is small enough.



Continuous Variables and Objective

Unconstrained

One-dimensional search

Example:

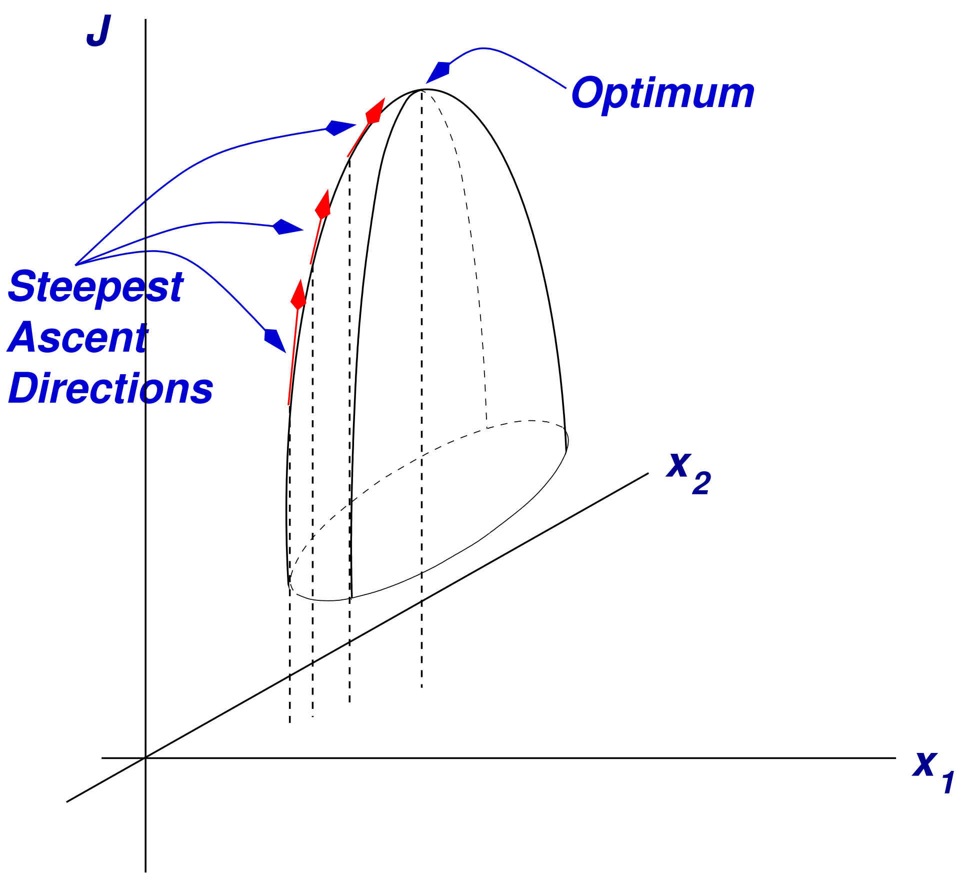
$$f(t) = 4 - t^2$$

t_0
0
3
1.33333333333333
1.84615384615385
2.03225806451613
1.99872040946897
1.99998976002621
2.0000000032768
1.99999999999999
2

Continuous Variables and Objective

Unconstrained

Multi-dimensional search



Optimum often found by *steepest ascent* or *hill-climbing* methods.

Continuous Variables and Objective

Unconstrained

Gradient search*

To maximize $J(x)$, where x is a vector (and J is a scalar function that has *nice* properties):

0. Set $n = 0$. Guess x_0.

1. Evaluate $\frac{\partial J}{\partial x}(x_n)$.

2. Let t be a scalar. Define $J_n(t) = J\left\{x_n + t\frac{\partial J}{\partial x}(x_n)\right\}$

   Find (by *one-dimensional search*) $t_n^\star$, the value of t that maximizes $J_n(t)$.

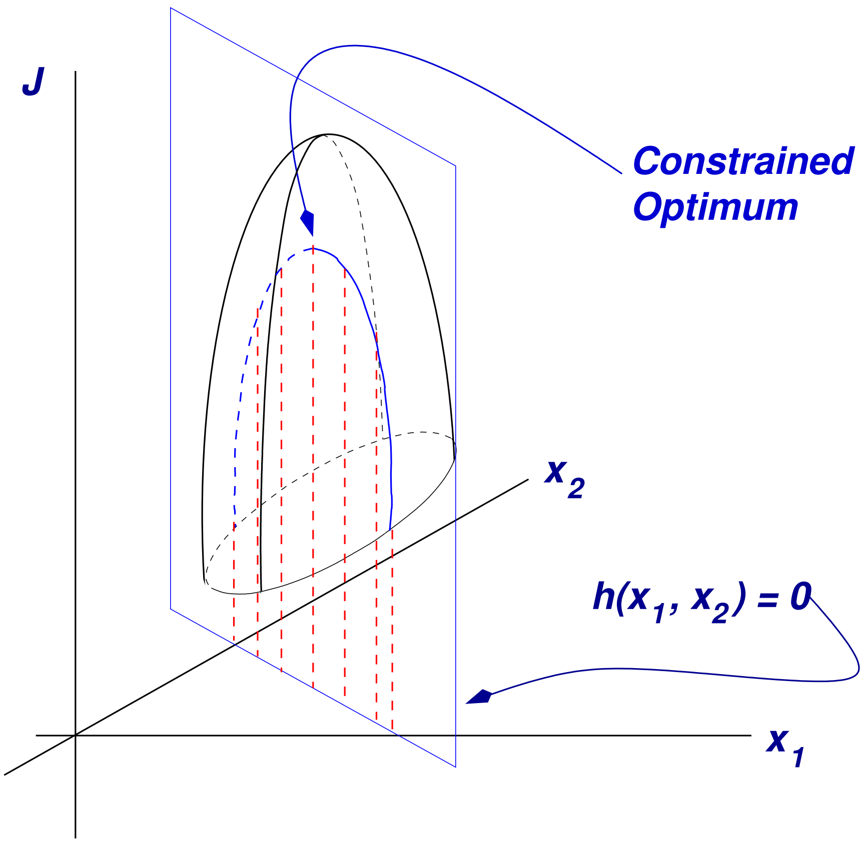
3. Set $x_{n+1} = x_n + t_n^\star \frac{\partial J}{\partial x}(x_n)$.

4. Set $n \leftarrow n + 1$. Go to Step 1.

* also called *steepest ascent* or *steepest descent*.

Continuous Variables and Objective

Constrained



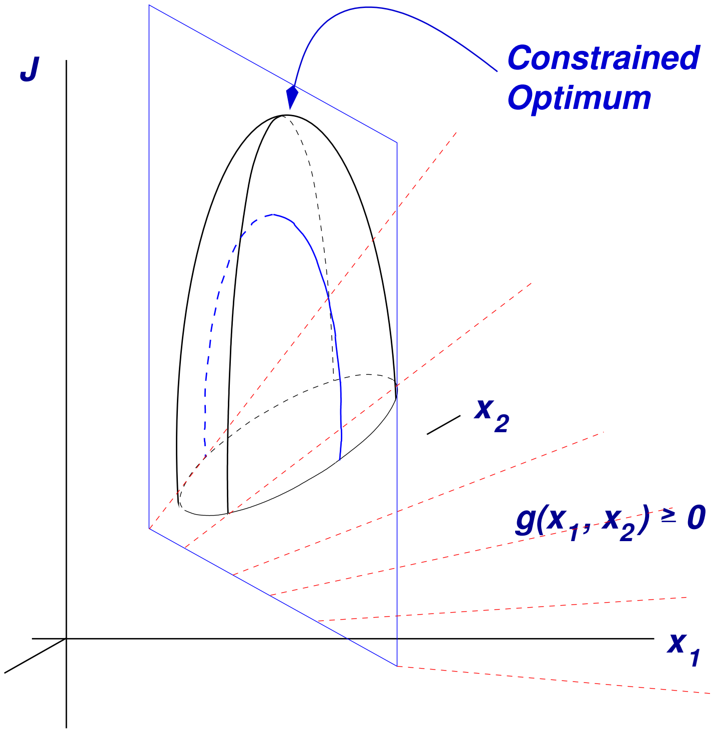
Equality constrained: solution is *on* the constraint surface.

Problems are much easier when constraint is linear, ie, when the surface is a plane.

- In that case, replace $\partial J / \partial x$ by its projection onto the constraint plane.
- *But first: find an initial feasible guess.*

Continuous Variables and Objective

Constrained



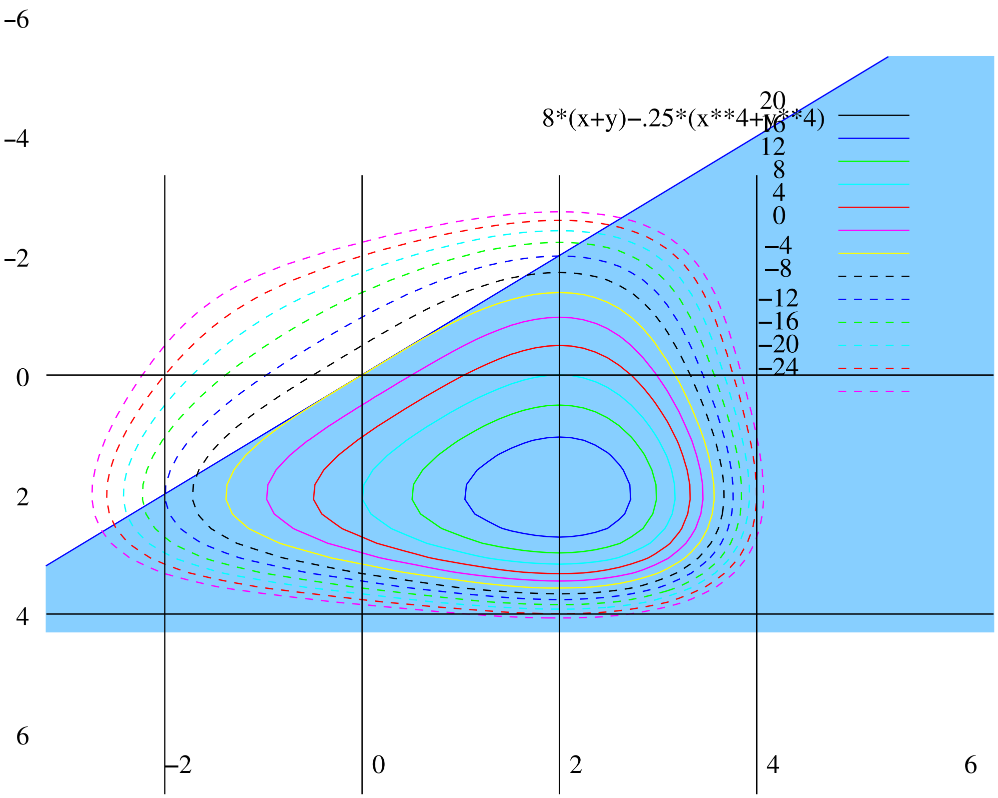
Inequality constrained: solution is required to be on *one side of* the plane.

Inequality constraints that are satisfied with equality are called *effective* or *active* constraints.

If we knew which constraints would be effective, the problem would reduce to an equality-constrained optimization.

Continuous Variables and Objective

Constrained



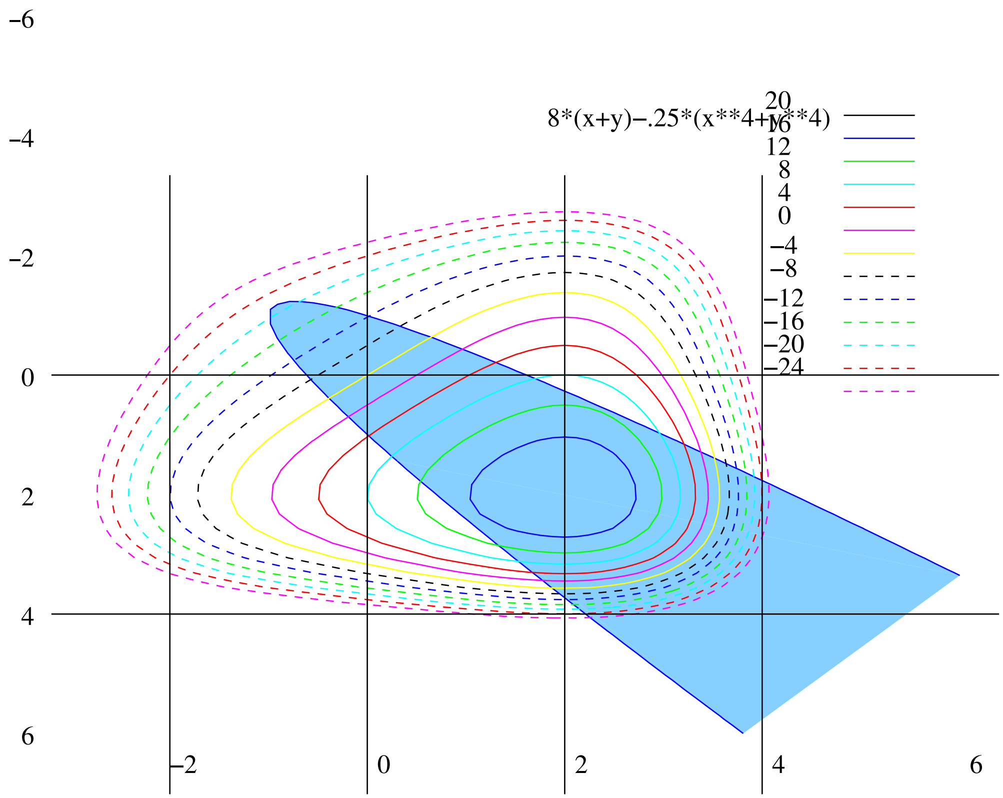
Minimize $8(x+y) - (x^4 + y^4)/4$

subject to $x + y \geq 0$

Solving a linearly-constrained problem is relatively easy. If the solution is not in the interior, search within the boundary plane.

Continuous Variables and Objective

Constrained



Minimize $8(x + y) - (x^4 + y^4)/4$

subject to $x - (x - y)^2 + 1 \geq 0$

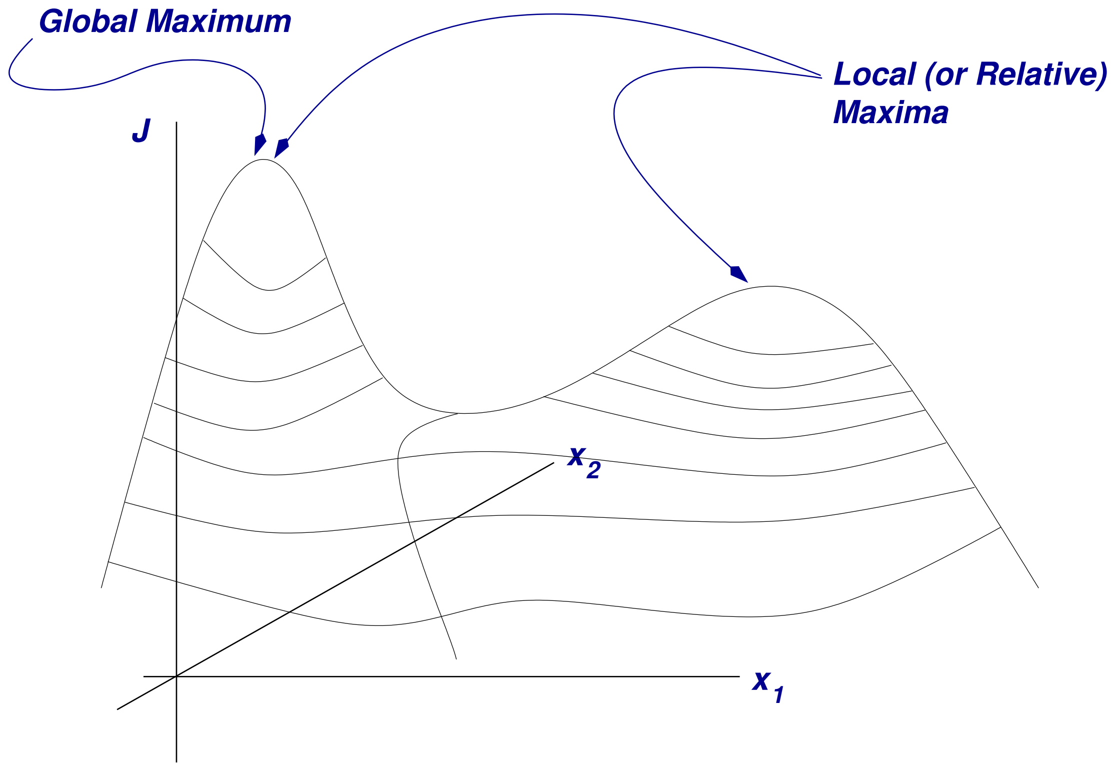
Solving a nonlinearly-constrained problem is not so easy.

Searching within the boundary is numerically difficult.

Optimization problems with continuous variables, objective, and constraints are called *nonlinear programming* problems, especially when at least one of J, h, g are not linear.

When all of J, h, g are linear, the problem is a *linear programming* problem.

Multiple Optima



Danger: a search might find a local, rather than the global, optimum.

Primals and Duals

Consider the two problems:

$$\min f(x) \qquad\qquad \max j(x)$$

$$\text{subject to } j(x) \geq J \qquad\qquad \text{subject to } f(x) \leq F$$

$f(x)$, F, $j(x)$, and J are scalars. We will call these problems *duals* of one another. (However, this is not the conventional use of the term.) Under certain conditions when the last inequalities are effective, the same x satisfies both problems.

We will call one the *primal problem* and the other the *dual problem*.

Primals and Duals

Generalization:

$$\min f(x)$$

subject to $h(x) = 0$

$$g(x) \leq 0$$

$$j(x) \geq J$$

$$\max j(x)$$

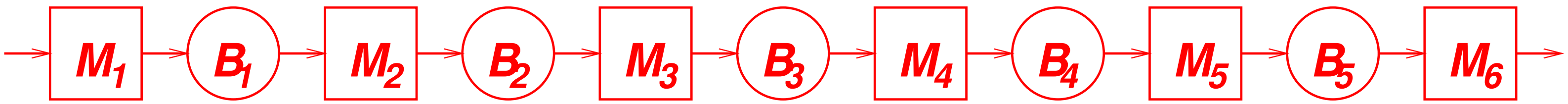
subject to $h(x) = 0$

$$g(x) \leq 0$$

$$f(x) \leq F$$

Buffer Space Allocation

Problem statement



Problem: Design the buffer space for a line. The machines have already been selected. Minimize the total buffer space needed to achieve a target production rate.

Other problems: minimize total average inventory; maximize profit (revenue - inventory cost - buffer space cost); choose machines as well as buffer sizes; etc.

Buffer Space Allocation

Problem statement

Assume a deterministic processing time line with k machines with r_i and p_i known for all $i = 1, ..., k$. Assume minimum buffer size N^{MIN}. Assume a target production rate P^*. Then the function $P(N_1, ..., N_{k-1})$ is known — it can be evaluated using the decomposition method. The problem is:

Primal problem:

$$\text{Minimize} \quad \sum_{i=1}^{k-1} N_i$$

$$\text{subject to} \quad P(N_1, ..., N_{k-1}) \geq P^*$$

$$N_i \geq N^{\text{MIN}}, i = 1, ..., k-1.$$

In the following, we treat the N_is like a set of continuous variables.

Properties of $P(N_1, ..., N_{k-1})$

$$P(\infty, ..., \infty) = \min_{i=1,...,k} e_i$$

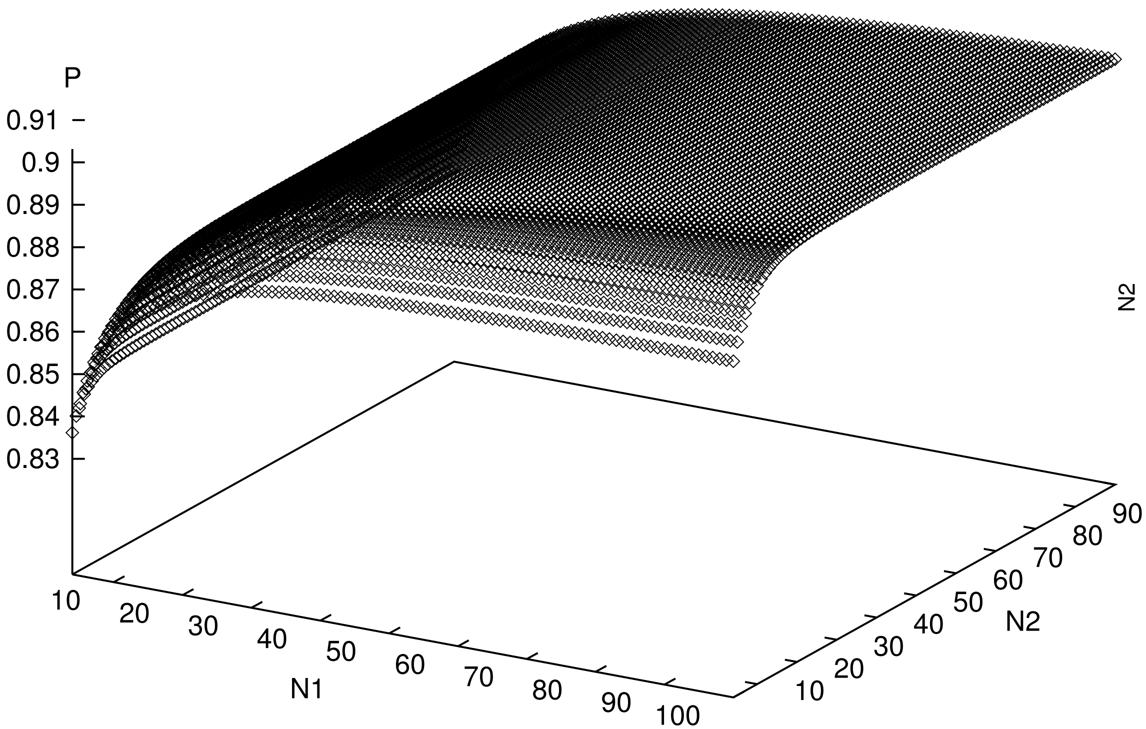
$$P(N^{\text{MIN}}, ..., N^{\text{MIN}}) \approx \frac{1}{1 + \sum_{i}^{k-1} \frac{p_i}{r_i}} << P(\infty, ..., \infty)$$

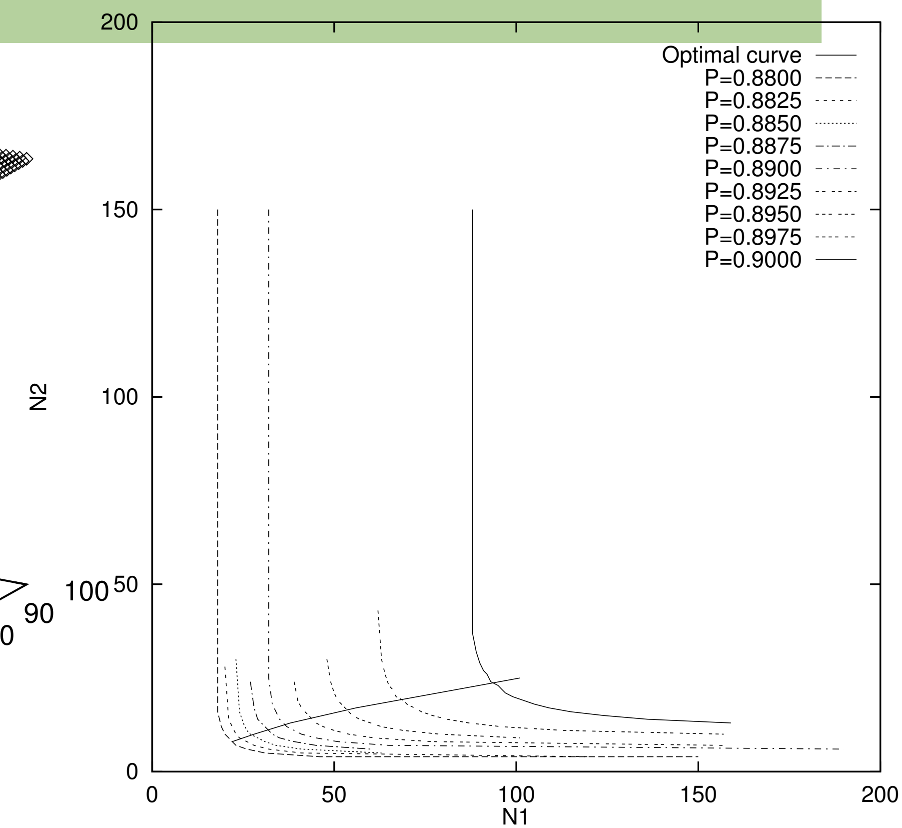
- *Continuity:* A small change in any N_i creates a small change in P.
- *Monotonicity:* The production rate increases monotonically in each N_i .
- *Concavity:* The production rate appears to be a concave function of the vector $(N_1, ..., N_{k-1})$.

Buffer Space Allocation

Properties of $P(N_1, ..., N_{k-1})$

Example — 3-machine line





$r_1 = .35$	$r_2 = .15$	$r_3 = .4$
$p_1 = .037$	$p_2 = .015$	$p_3 = .02$
$e_1 = .904$	$e_2 = .909$	$e_3 = .952$

Buffer Space Allocation

Solution

Primal problem

Minimize $$\sum_{i=1}^{k-1} N_i$$

subject to $$P(N_1, ..., N_{k-1}) \geq P^*$$

$$N_i \geq N^{\text{MIN}}, i = 1, ..., k-1.$$

Difficulty: If all the buffers are larger than N^{MIN}, the solution will satisfy $P(N_1, ..., N_{k-1}) = P^*$. *(Why?)* But $P(N_1, ..., N_{k-1})$ is nonlinear and cannot be expressed in closed form. Therefore any solution method will have to search within this constraint and all steps will be small and there will be many iterations.

It would be desirable to transform this problem into one with linear constraints.

Buffer Space Allocation

Solution

Dual problem

Maximize $P(N_1, ..., N_{k-1})$

subject to $$\sum_{i=1}^{k-1} N_i \leq N^{\text{TOTAL}}$$ *specified*

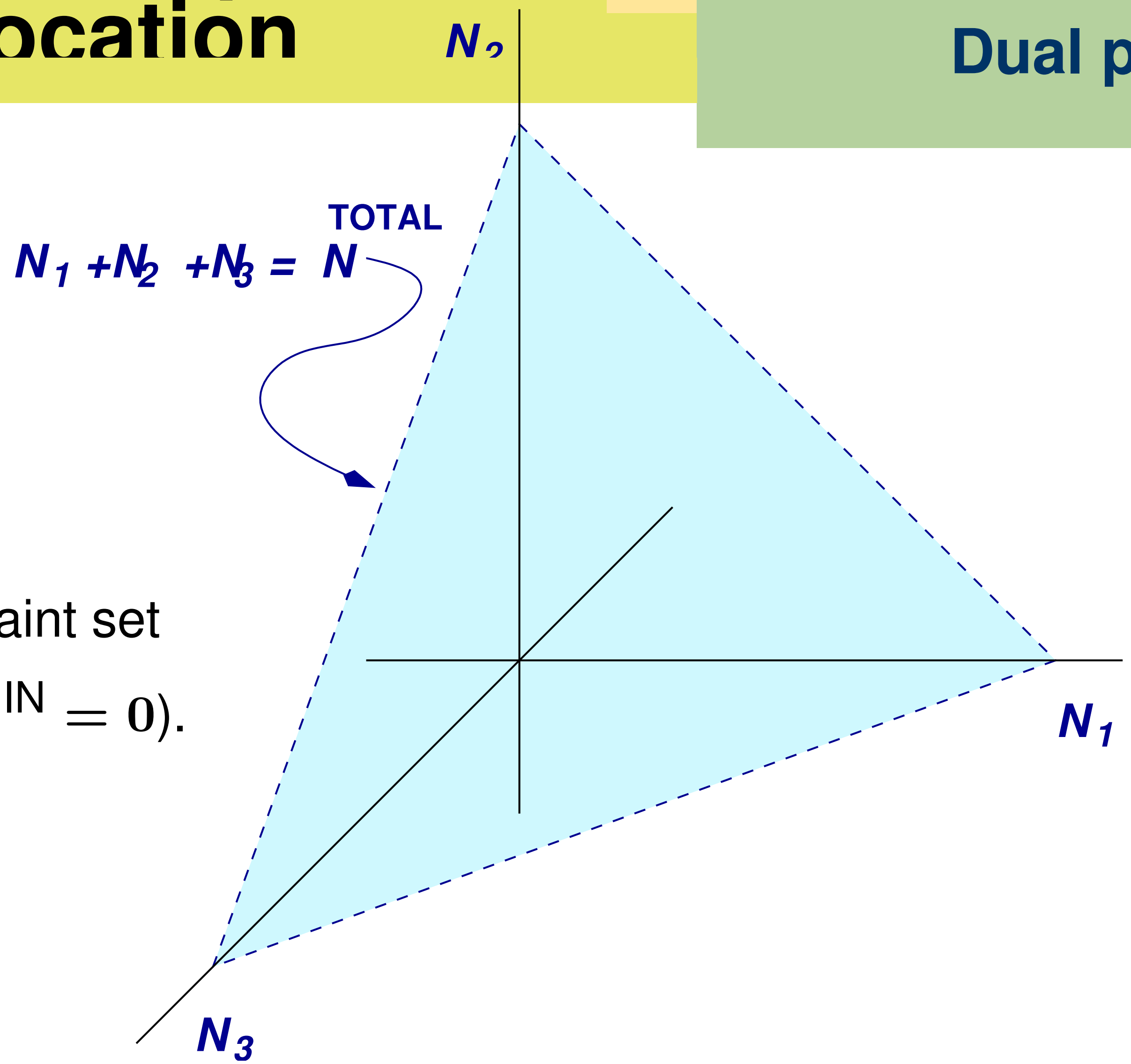
$$N_i \geq N^{\text{MIN}}, i = 1, ..., k-1.$$

All the constraints are linear. The solution will satisfy the N^{TOTAL} constraint with equality (assuming the problem is feasible). *(1. Why? 2. When would the problem be infeasible?)* This problem is consequently relatively easy to solve.

Buffer Space Allocation

Solution

Dual problem



Constraint set

(if $N^{\text{MIN}} = 0$).

Buffer Space Allocation

Solution

Primal strategy

Solution of the primal problem:

1. Guess N^{TOTAL}.

2. Solve the dual problem. Evaluate $P = P_{\text{MAX}}(N^{\text{TOTAL}})$.

3. Use a one-dimensional search method to find N^{TOTAL} such that $P_{\text{MAX}}(N^{\text{TOTAL}}) = P^*$.

Buffer Space Allocation

Solution

Dual Algorithm

Maximize $P(N_1, ..., N_{k-1})$

subject to $\sum_{i=1}^{k-1} N_i = N^{\text{TOTAL}}$ *specified*

$N_i \geq N^{\text{MIN}}, i = 1, ..., k-1.$

- Start with an initial guess $(N_1, ..., N_{k-1})$ that satisfies $\sum_{i=1}^{k-1} N_i = N^{\text{TOTAL}}$.
- Calculate the gradient vector $(g_1, ..., g_{k-1})$:

$$g_i = \frac{P(N_1, \ldots, N_i + \delta N, \ldots, N_{k-1}) - P(N_1, \ldots, N_i, \ldots, N_{k-1})}{\delta N}$$

- Calculate the *projected* gradient vector $(\hat{g}_1, ..., \hat{g}_{k-1})$:

$$\hat{g}_i = g_i - \bar{g} \quad \text{where} \quad \bar{g} = \frac{1}{k-1} \sum_{i=1}^{k-1} g_i$$

Buffer Space Allocation

Solution

Dual Algorithm

- The projected gradient $\hat{g}$ satisfies

$$\sum_{i=1}^{k-1} \hat{g}_i = \sum_{i=1}^{k-1} (g_i - \bar{g}) = \sum_{i=1}^{k-1} g_i - (k-1)\bar{g} = 0$$

- Therefore, if A is a scalar, then

$$\sum_{i=1}^{k-1} (N_i + A\hat{g}_i) = \sum_{i=1}^{k-1} N_i + \sum_{i=1}^{k-1} A\hat{g}_i = \sum_{i=1}^{k-1} N_i$$

so if $(N_1, ..., N_{k-1})$ satisfies $\sum_{i=1}^{k-1} N_i = N^{\text{TOTAL}}$ and $N_i' = N_i + A\hat{g}_i$ for any scalar A, then $(N_1', ..., N_{k-1}')$ satisfies $\sum_{i=1}^{k-1} N_i' = N^{\text{TOTAL}}$.

- That is, if N is on the constraint, then $N + A\hat{g}$ is also on the constraint (as long as all elements $\geq N^{\text{MIN}}$).

Buffer Space Allocation

Solution

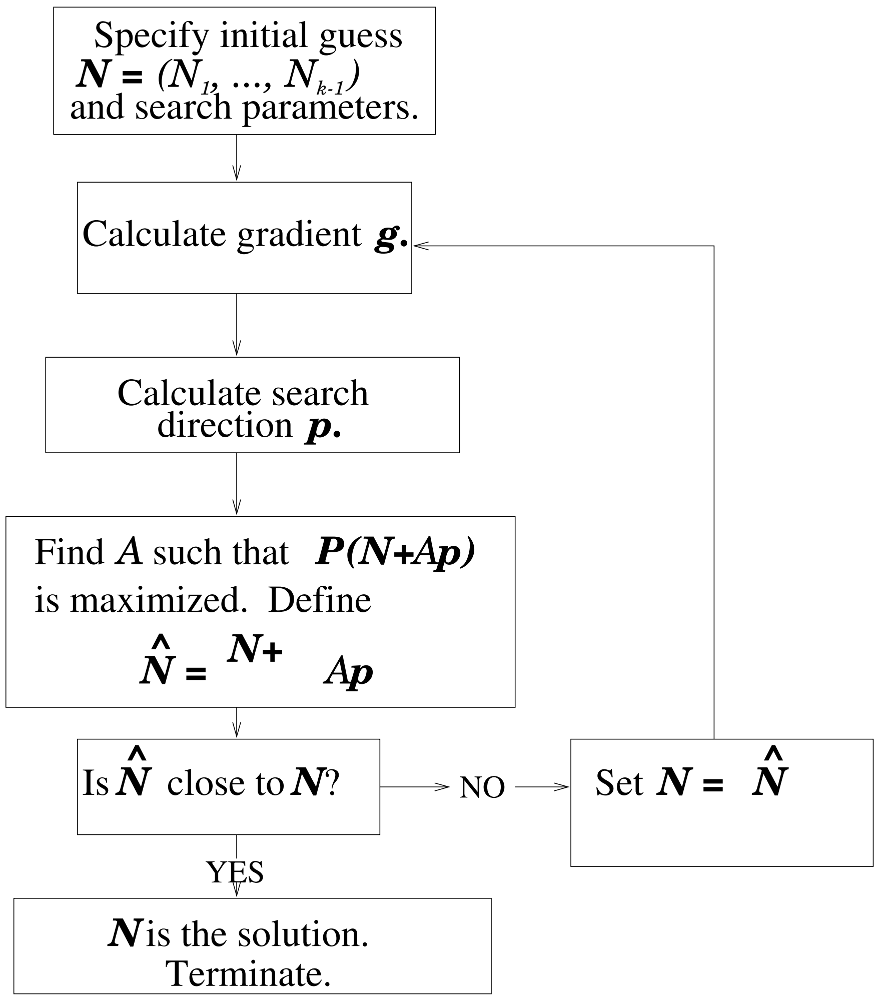
Dual Algorithm

- The gradient $\boldsymbol{g}$ is the direction of greatest increase of $\boldsymbol{P}$.
- The projected gradient $\hat{\boldsymbol{g}}$ is the direction of greatest increase of $\boldsymbol{P}$ *within the constraint plane* .
- Therefore, once we have a point $\boldsymbol{N}$ on the constraint plane, the best improvement is to move in the direction of $\hat{\boldsymbol{g}}$; that is, $\boldsymbol{N} + \boldsymbol{A}\hat{\boldsymbol{g}}$.
- To find the best possible improvement, we find $\boldsymbol{A}^*$, the value of $\boldsymbol{A}$ that maximizes $\boldsymbol{P}(\boldsymbol{N} + \boldsymbol{A}\hat{\boldsymbol{g}})$. $\boldsymbol{A}$ is a scalar, so this is a *one-dimensional search.*
- $\boldsymbol{N} + \boldsymbol{A}^*\hat{\boldsymbol{g}}$ is the next guess for $\boldsymbol{N}$, and the process repeats.

Buffer Space Allocation

Solution

Dual Algorithm

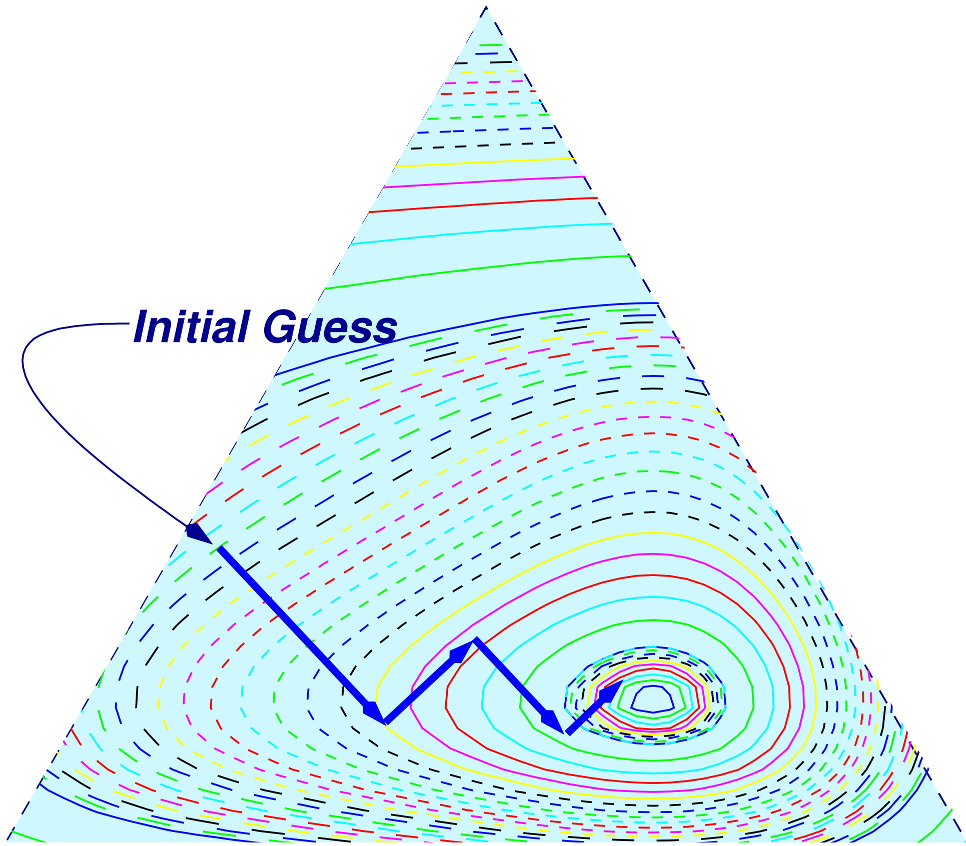


(Here, $p = \hat{g}$.)

Buffer Space Allocation

Solution

Dual Algorithm

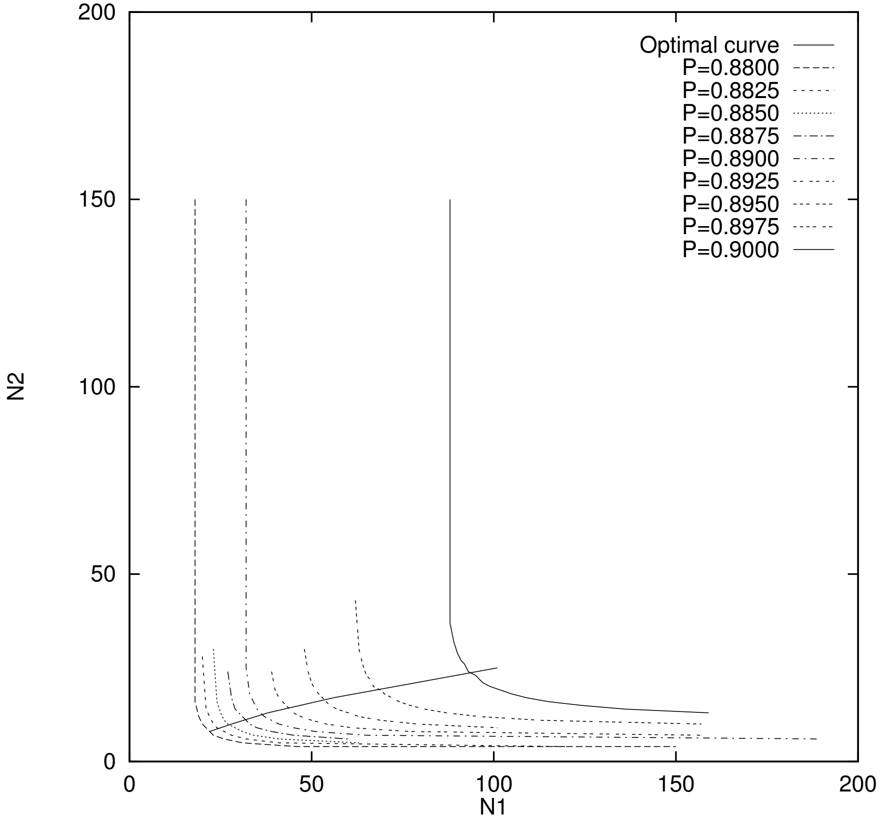


Buffer Space Allocation

Solution

Primal algorithm

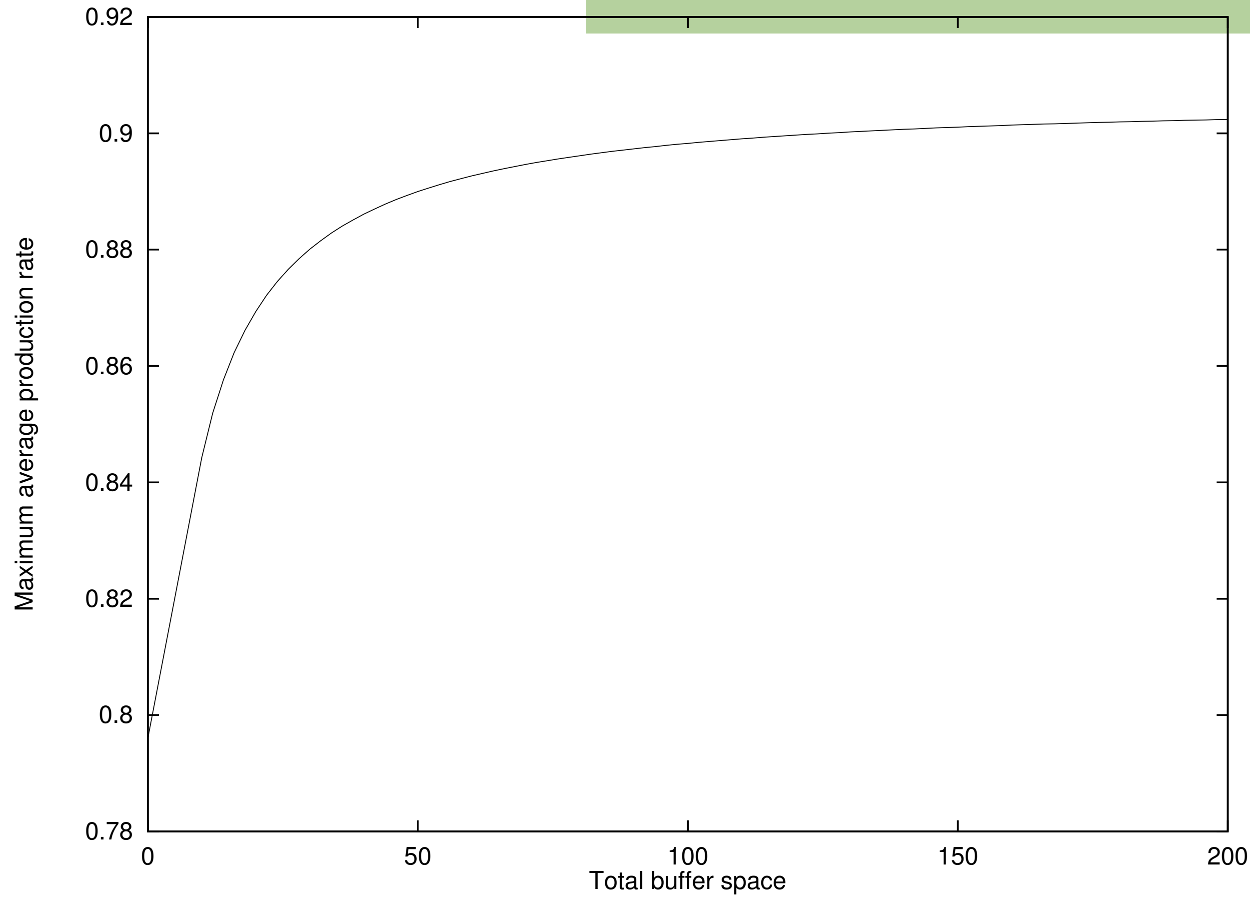
- We can solve the dual problem for any N^{TOTAL}.
- We can calculate $N_1(N^{\text{TOTAL}})$, $N_2(N^{\text{TOTAL}})$, ..., $N_{k-1}(N^{\text{TOTAL}})$, $P_{\text{MAX}}(N^{\text{TOTAL}})$.



Buffer Space Allocation

Solution

Primal algorithm



$P_{\text{MAX}}(N^{\text{TOTAL}})$ as a function of N^{TOTAL}.

Buffer Space Allocation

Solution

Primal algorithm

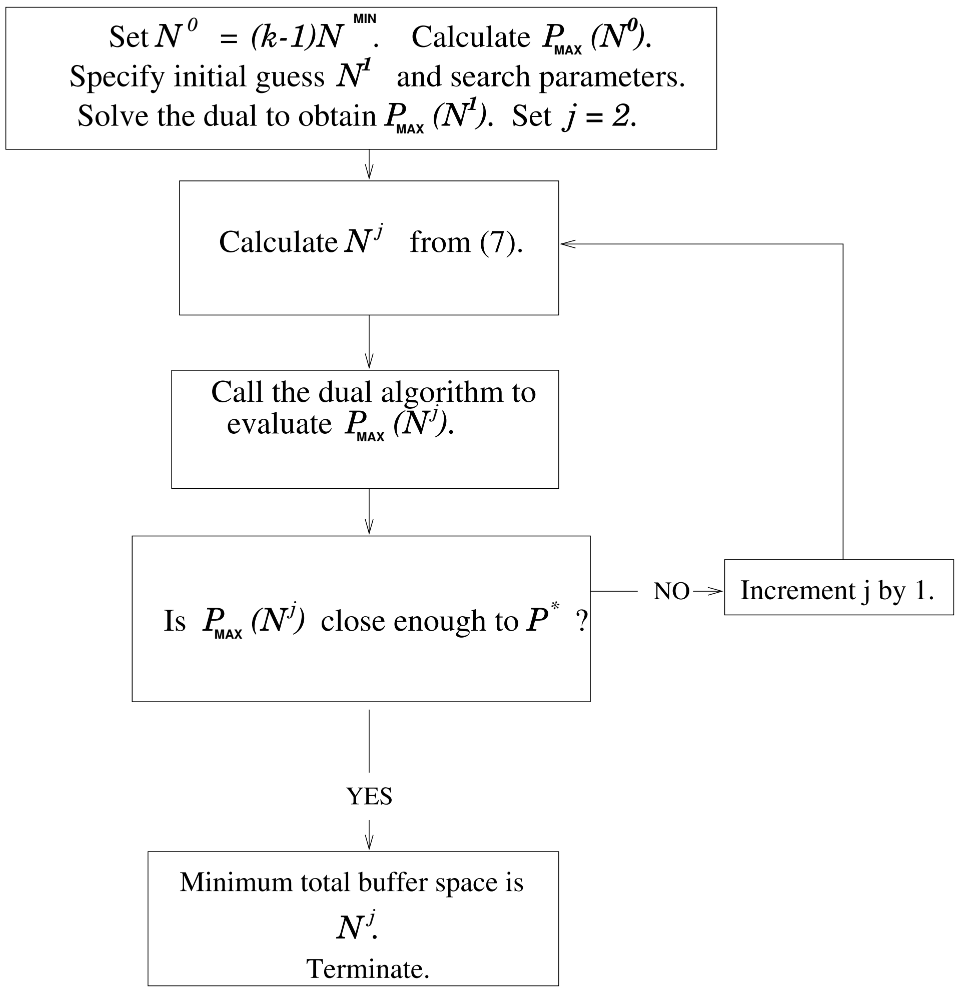
Then, we can find, by 1-dimensional search, N^{TOTAL} such that

$$P_{\text{MAX}}(N^{\text{TOTAL}}) = P^*.$$

Buffer Space Allocation

Solution

Primal algorithm



(Here, (7) refers to a one-dimensional search.)

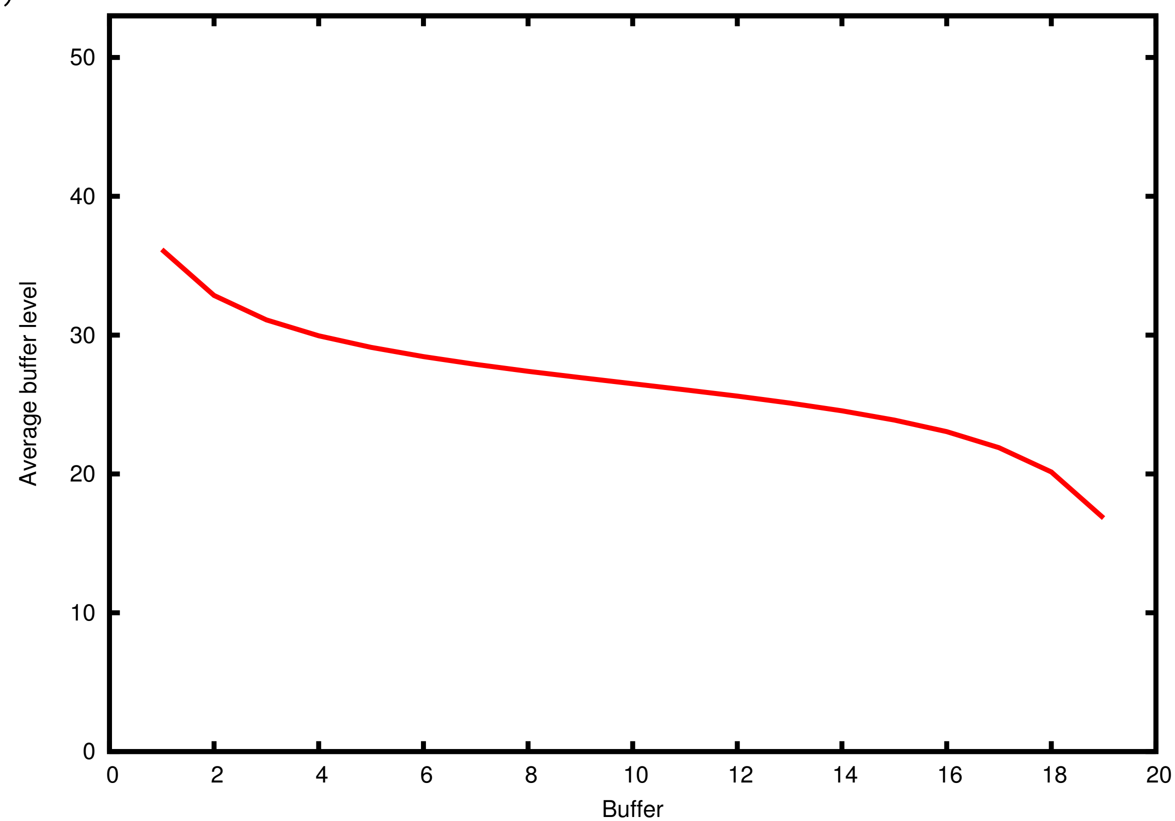
Buffer Space Allocation

Example

The "Bowl Phenomena"

- Problem: how to allocate space in a line with identical machines.
- Case: 20-machine continuous material line, $r_i = .0952$, $p_i = .005$, and $\mu_i = 1$, $i = 1, ..., 20$.

First, we show the average WIP distribution if all buffers are the same size: $N_i = 53$, $i = 1, ..., 19$

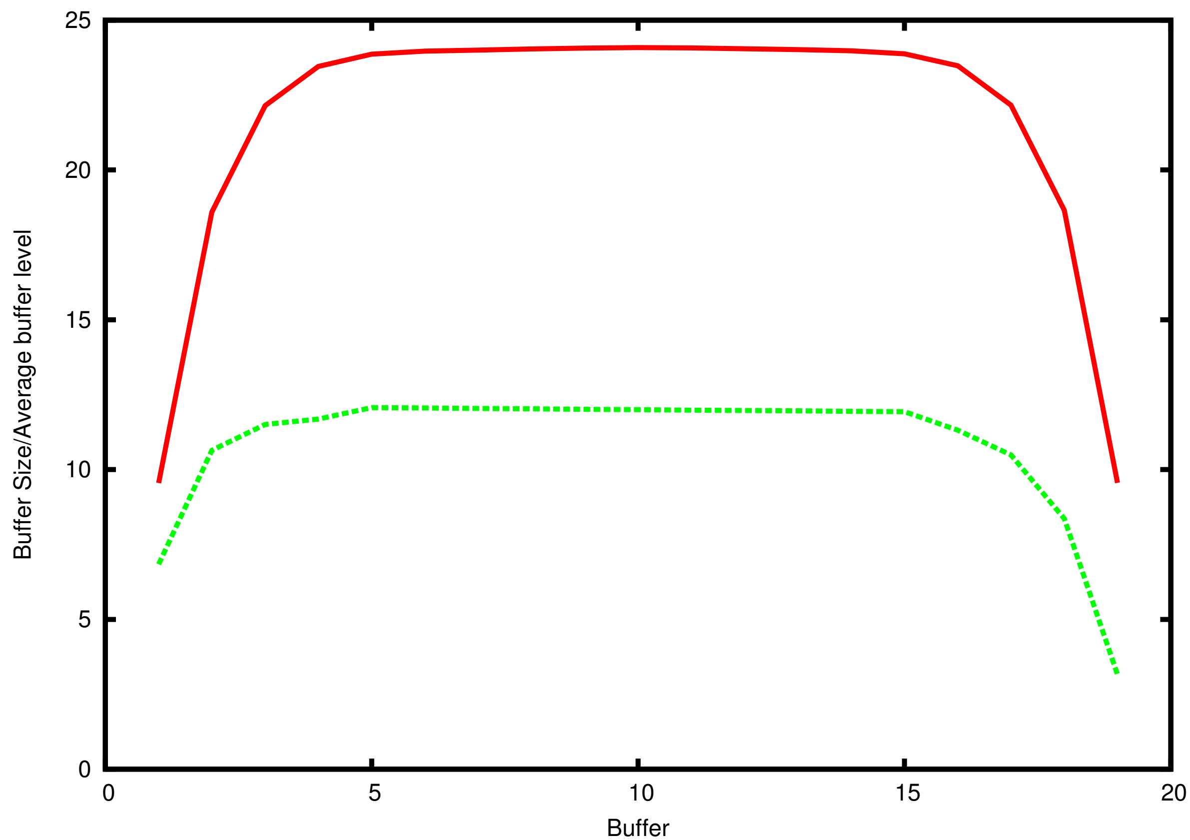


Buffer Space Allocation

Example

The “Bowl Phenomena”

- This shows the optimal distribution of buffer *space* and the resulting distribution of *average inventory* .



Buffer Space Allocation

Example

The "Bowl Phenomena"

Observations:

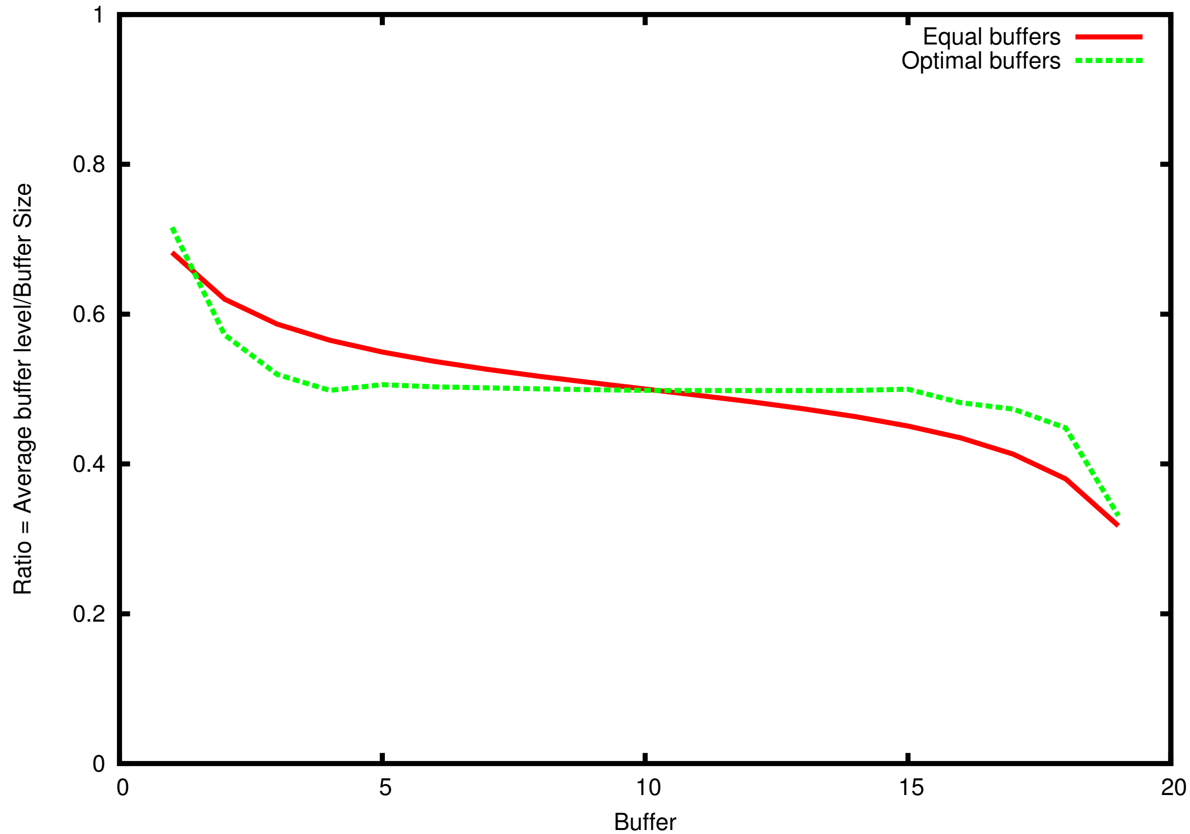
- The optimal distribution of buffer space does not look like the distribution of inventory in the line with equal buffers. *Why not? Explain the shape of the optimal distribution.*
- The distribution of average inventory is not symmetric.

Buffer Space Allocation

Example

The “Bowl Phenomena”

- This shows the ratios of *average inventory* to buffer size with equal buffers and with optimal buffers.



Buffer Space Allocation

Example

- Design the buffers for a 20-machine production line.
- The machines have been selected, and the only decision remaining is the amount of space to allocate for in-process inventory.
- *The goal is to determine the smallest amount of in-process inventory space so that the line meets a production rate target.*

Buffer Space Allocation

Example

- The common operation time is one operation per minute.
- The target production rate is .88 parts per minute.

Buffer Space Allocation

Example

- *Case 1* MTTF= 200 minutes and MTTR = 10.5 minutes for all machines ($\boldsymbol{P = .95}$ parts per minute).

Buffer Space Allocation

Example

- *Case 1* MTTF= 200 minutes and MTTR = 10.5 minutes for all machines ($\boldsymbol{P = .95}$ parts per minute).
- *Case 2* Like Case 1 except Machine 5. For Machine 5, MTTF = 100 and MTTR = 10.5 minutes ($\boldsymbol{P = .905}$ parts per minute).

Buffer Space Allocation

Example

- *Case 1* MTTF= 200 minutes and MTTR = 10.5 minutes for all machines ($\boldsymbol{P = .95}$ parts per minute).
- *Case 2* Like Case 1 except Machine 5. For Machine 5, MTTF = 100 and MTTR = 10.5 minutes ($\boldsymbol{P = .905}$ parts per minute).
- *Case 3* Like Case 1 except Machine 5. For Machine 5, MTTF = 200 and MTTR = 21 minutes ($\boldsymbol{P = .905}$ parts per minute).

Buffer Space Allocation

Example

Are buffers really needed?

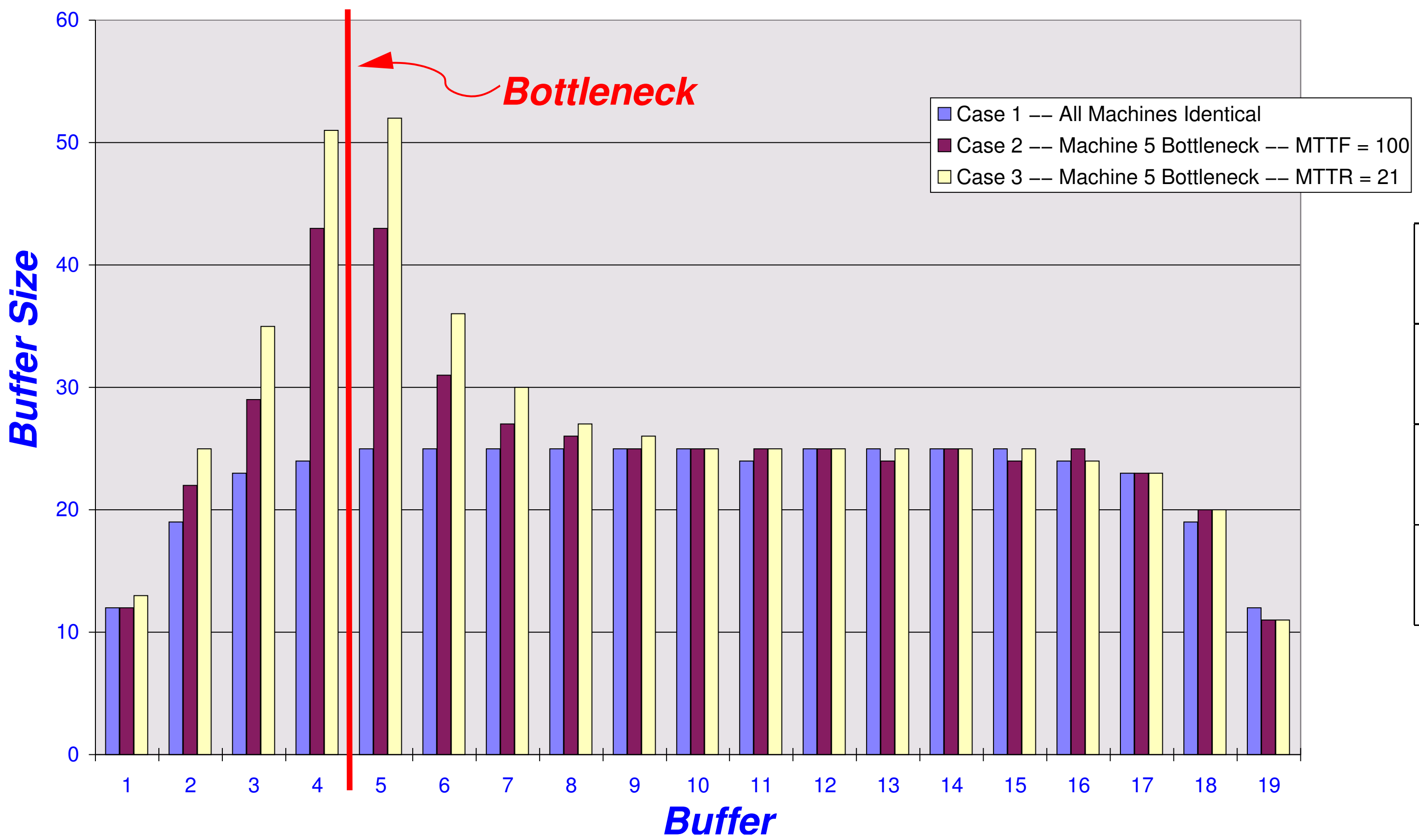
Line	Production rate with no buffers, parts per minute
Case 1	.487
Case 2	.475
Case 3	.475

Yes. *How were these numbers calculated?*

Buffer Space Allocation

Example

Solution

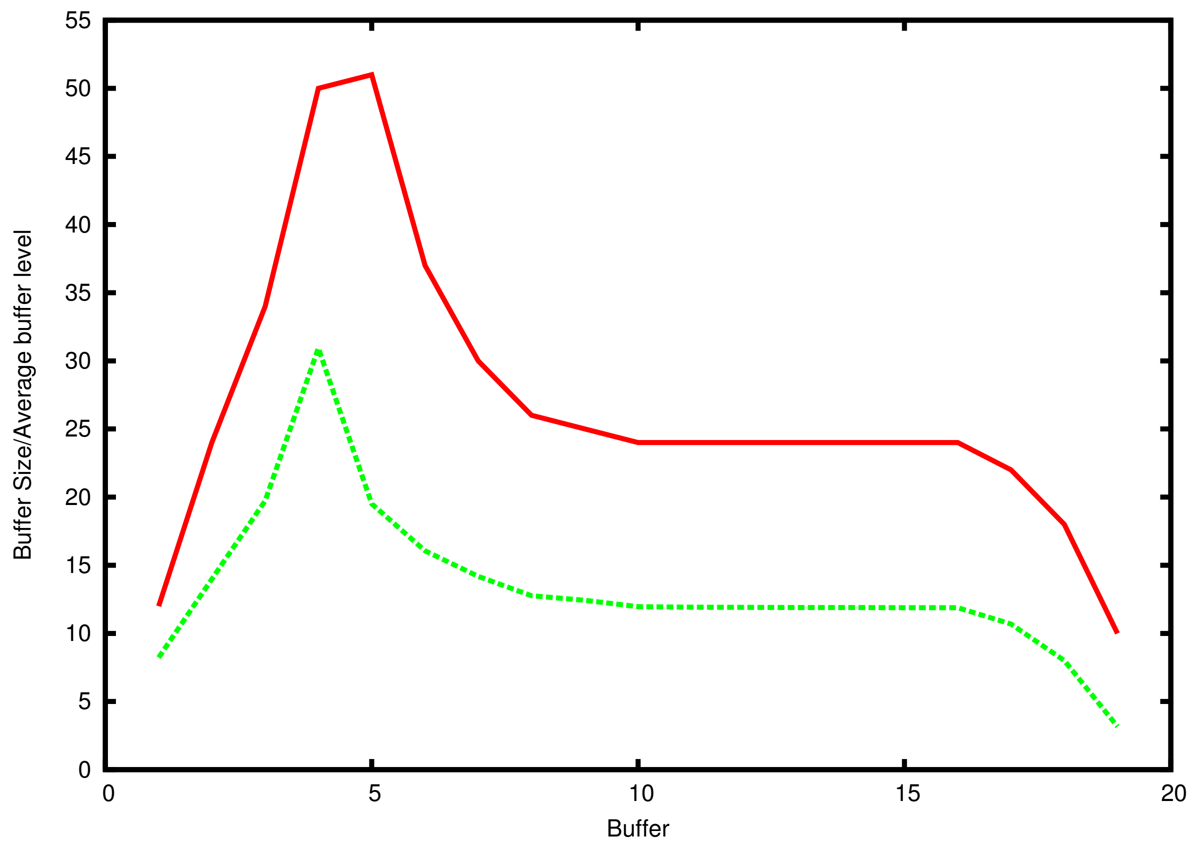


Line	Space
Case 1	430
Case 2	485
Case 3	523

Buffer Space Allocation

Example

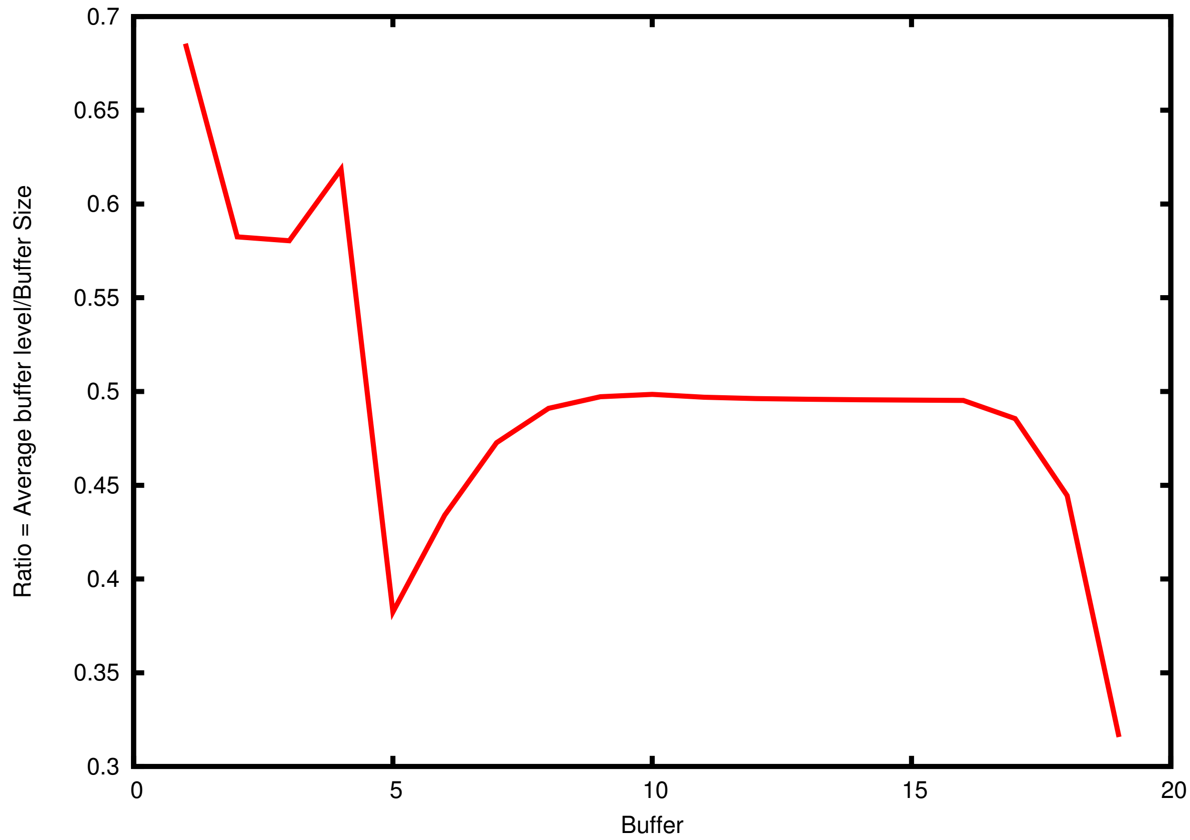
- This shows the optimal distribution of buffer *space* and the resulting distribution of *average inventory* for Case 3.



Buffer Space Allocation

Example

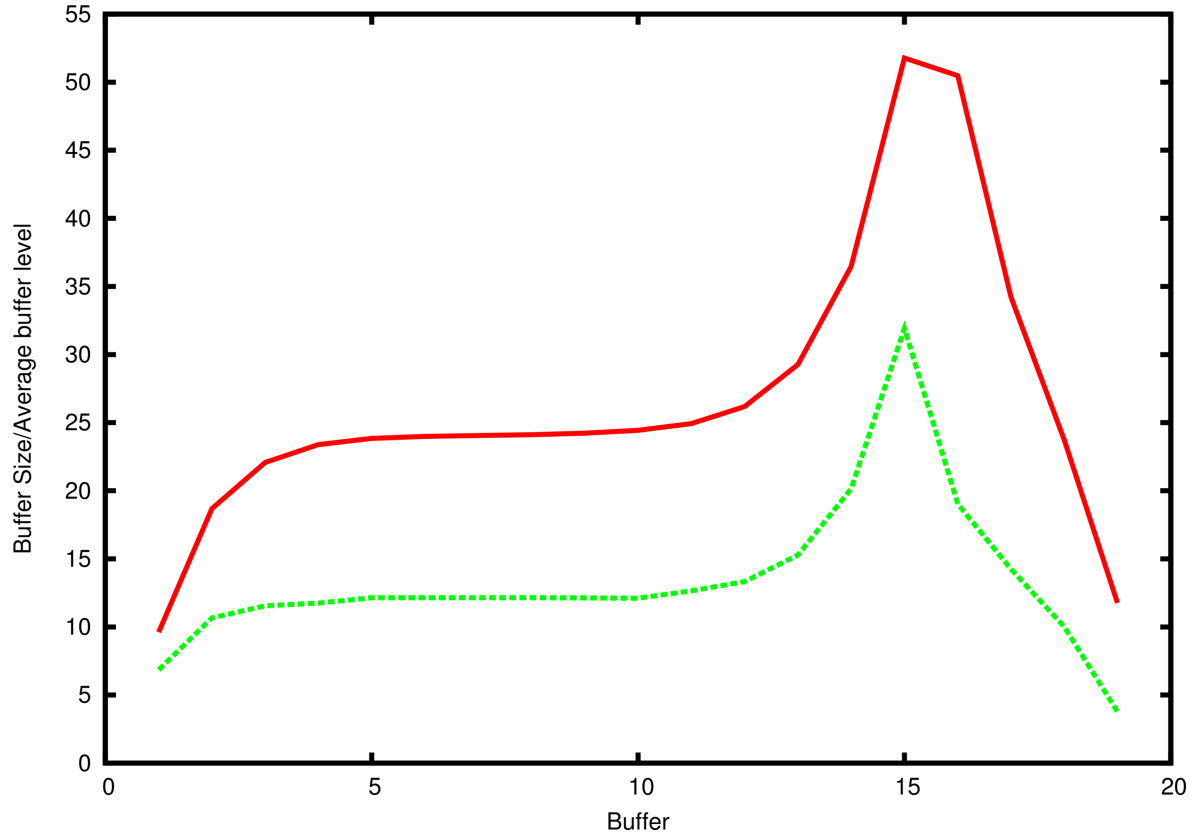
- This shows the ratio of *average inventory* to buffer size with optimal buffers for Case 3.



Buffer Space Allocation

Example

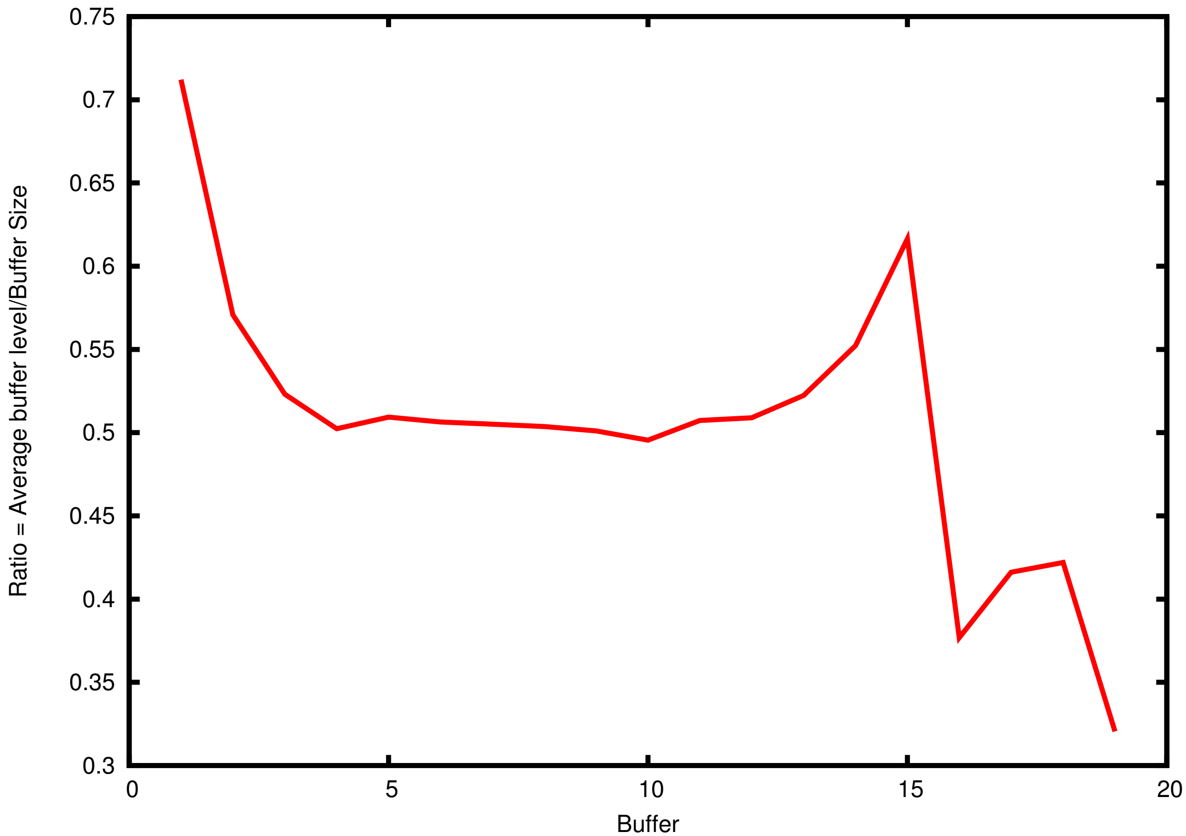
- *Case 4:* Same as Case 3 except bottleneck is at Machine 15.
- This shows the optimal distribution of buffer *space* and the resulting distribution of *average inventory* for Case 4.



Buffer Space Allocation

Example

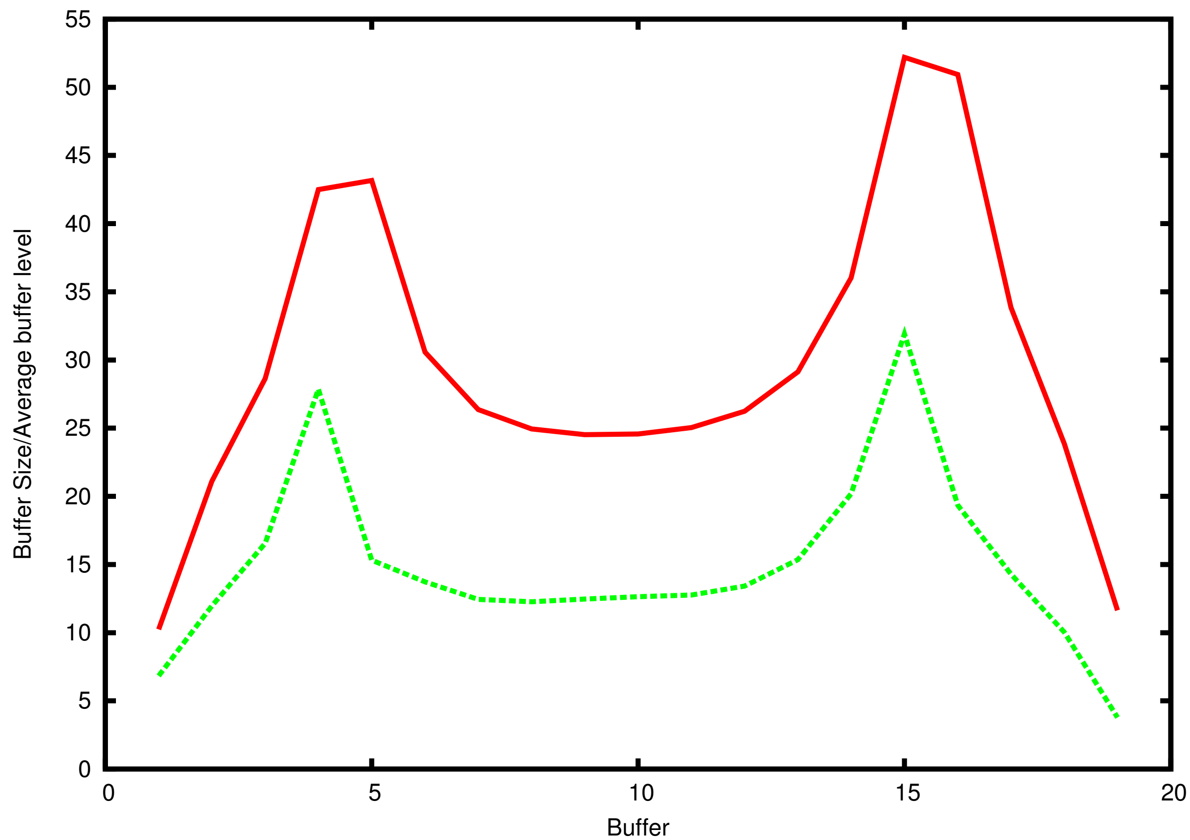
- This shows the ratio of *average inventory* to buffer size with optimal buffers for Case 4.



Buffer Space Allocation

Example

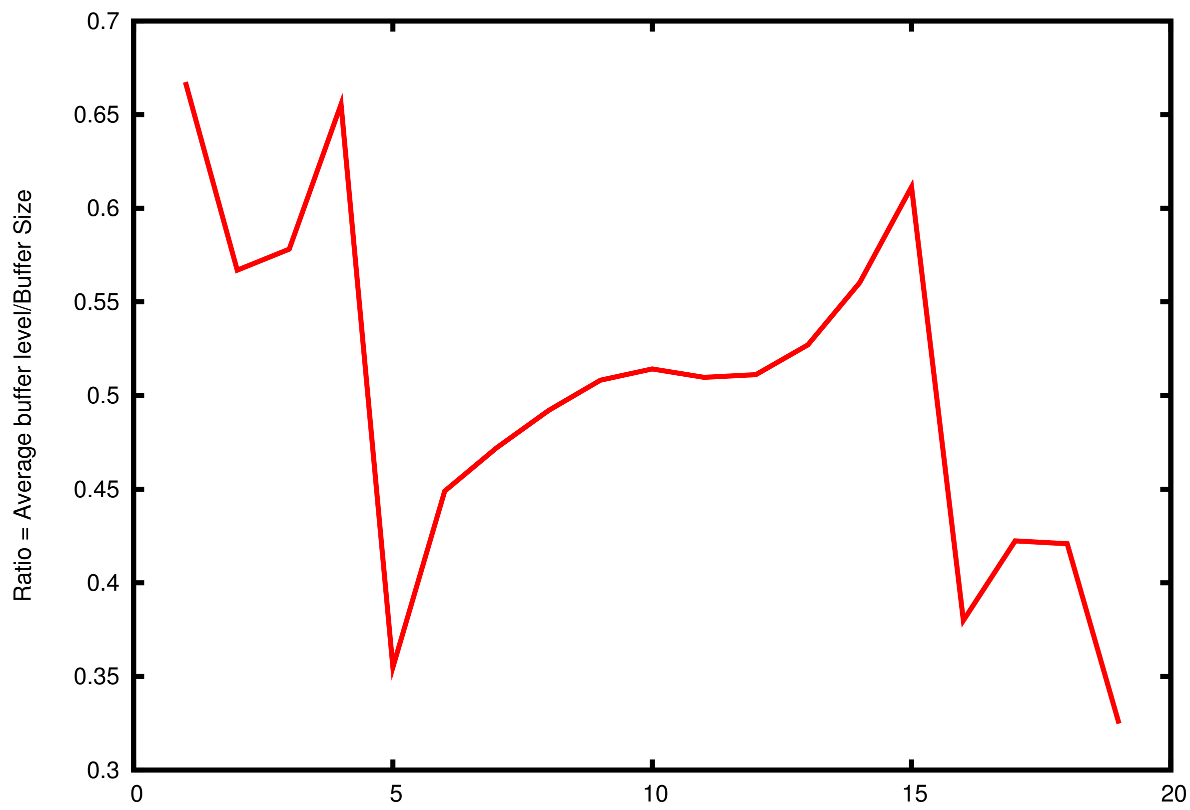
- *Case 5:* MTTF bottleneck at Machine 5, MTTR bottleneck at Machine 15.
- This shows the optimal distribution of buffer *space* and the resulting distribution of *average inventory* for Case 5.



Buffer Space Allocation

Example

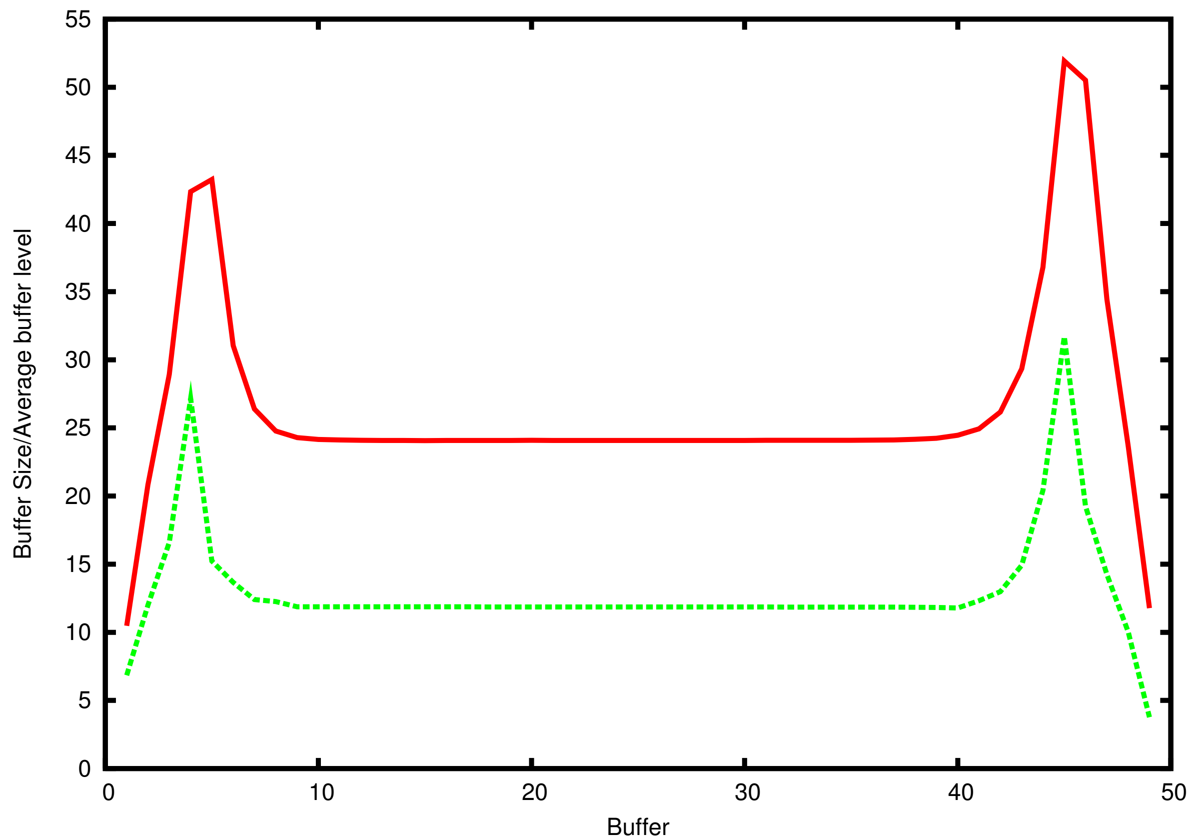
- This shows the ratio of *average inventory* to buffer size with optimal buffers for Case 5.



Buffer Space Allocation

Example

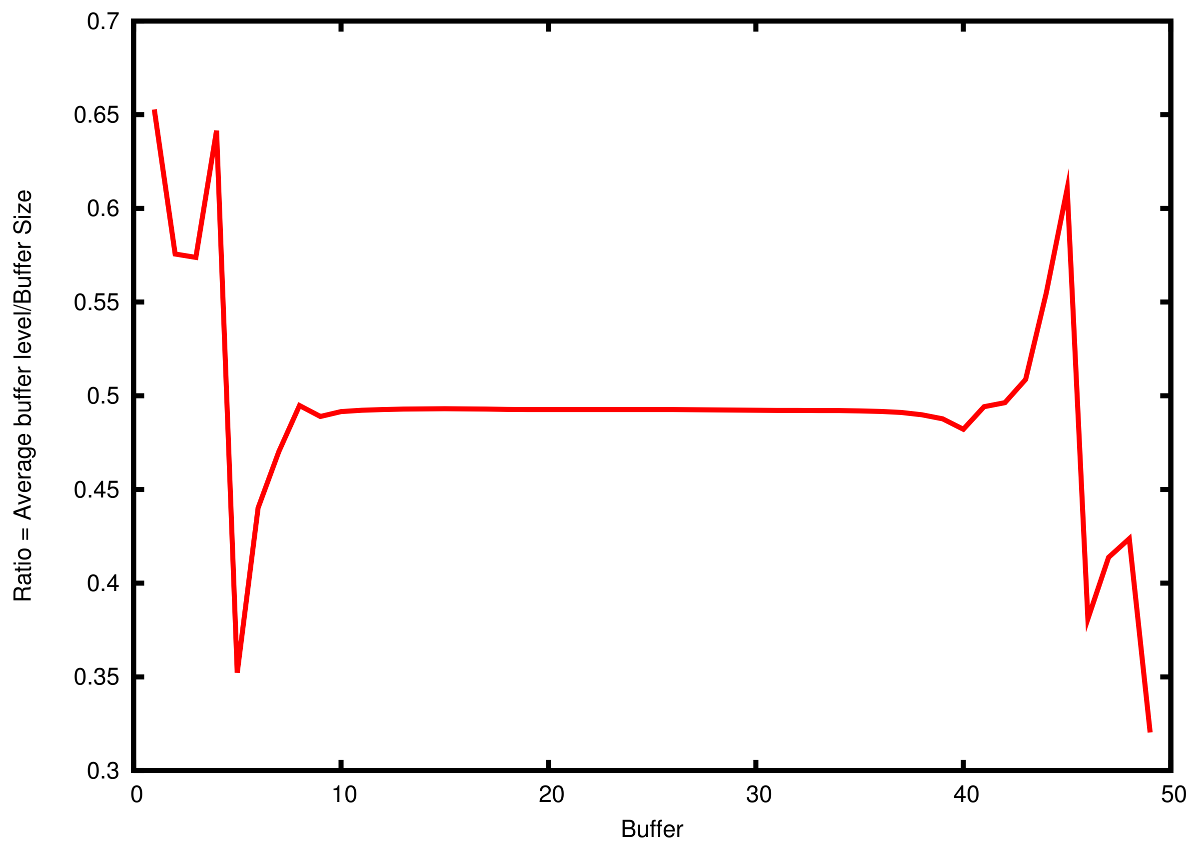
- *Case 6:* Like Case 6, but 50 machines, MTTR bottleneck at Machine 45.
- This shows the optimal distribution of buffer *space* and the resulting distribution of *average inventory* for Case 6.



Buffer Space Allocation

Example

- This shows the ratio of *average inventory* to buffer size with optimal buffers for Case 6.



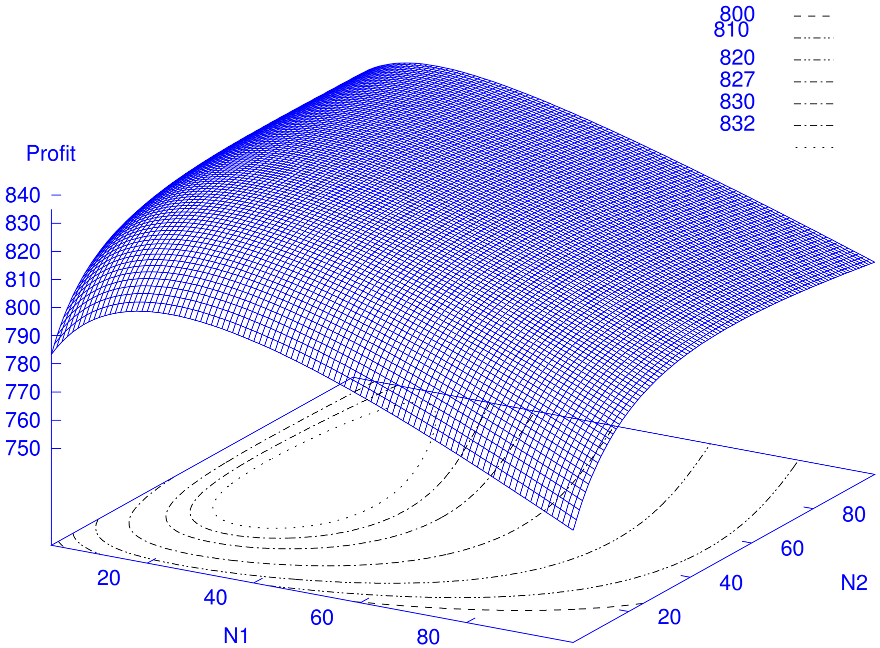
Buffer Space Allocation

Example

- Observation from studying buffer space allocation problems:
  - ⋆ *Buffer space is needed most where buffer level variability is greatest!*

Buffer Space Allocation

Profit as a function of buffer sizes



- Three-machine, continuous material line.
- $r_i = .1$, $p_i = .01$, $\mu_i = 1$.
- $\Pi = 1000 P(N_1, N_2) - (\bar{n}_1 + \bar{n}_2)$.

2.852 Manufacturing Systems Analysis
Spring 2010