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 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

Propose Design $D_0$

$n = 0$

Evaluate Design $D_n$

$P_n = P(D_n)$

Modify design as a function of past designs and performance.

$D_{n+1} = D(D_0, D_1, ..., D_n, P_0, P_1, ..., P_n)$

Is performance satisfactory?

no

yes

Quit

Typically, many designs are tested.
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

\( X \) is a set of possible choices. \( J \) is a scalar function defined on \( X \). \( h \) and \( g \) are vector functions defined on \( X \).

**Problem:** Find \( x \in X \) that satisfies

\[
J(x) \text{ is maximized (or minimized) — the objective}
\]

\[
\text{subject to}
\]

\[
h(x) = 0 — \text{equality constraints}
\]

\[
g(x) \leq 0 — \text{inequality constraints}
\]
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 $x$'s but none having complete control; etc.)
Continuous Variables and Objective

\[ X = \mathbb{R}^n. \ J \text{ is a scalar function defined on } \mathbb{R}^n. \ h(\in \mathbb{R}^m) \text{ and } \ g(\in \mathbb{R}^k) \text{ are vector functions defined on } \mathbb{R}^n. \]

**Problem:** Find \( x \in \mathbb{R}^n \) that satisfies

\[ J(x) \text{ is maximized (or minimized)} \]

**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

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:

\[ f(t) = 4 - t^2 \]

<table>
<thead>
<tr>
<th>( t_0 )</th>
<th>( t_2 )</th>
<th>( t_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>1.5</td>
<td>2.25</td>
<td>3</td>
</tr>
<tr>
<td>1.875</td>
<td>1.875</td>
<td>2.25</td>
</tr>
<tr>
<td>1.96875</td>
<td>1.96875</td>
<td>2.0625</td>
</tr>
<tr>
<td>1.9921875</td>
<td>2.00390625</td>
<td>2.015625</td>
</tr>
<tr>
<td>1.998046875</td>
<td>2.0009765625</td>
<td>2.000244140625</td>
</tr>
<tr>
<td>1.99951171875</td>
<td>2.00006103515625</td>
<td>2.00001525878906</td>
</tr>
<tr>
<td>1.99996948242188</td>
<td>2.0000095367432</td>
<td>2.00000381469727</td>
</tr>
</tbody>
</table>
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

Example:

\[ f(t) = 4 - t^2 \]

<table>
<thead>
<tr>
<th>( t_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>2.16666666666667</td>
</tr>
<tr>
<td>2.00641025641026</td>
</tr>
<tr>
<td>2.00001024002621</td>
</tr>
<tr>
<td>2.00000000002621</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
Continuous Variables and Unconstrained Objective

- 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

Example:

\[ f(t) = 4 - t^2 \]

<table>
<thead>
<tr>
<th>( t_0 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>1.33333333333333</td>
</tr>
<tr>
<td>1.84615384615385</td>
</tr>
<tr>
<td>2.03225806451613</td>
</tr>
<tr>
<td>1.99872040946897</td>
</tr>
<tr>
<td>1.99998976002621</td>
</tr>
<tr>
<td>2.0000000032768</td>
</tr>
<tr>
<td>1.99999999999999</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>
Continuous Unconstrained Variables and Objective

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) $t_n^*$, the value of $t$ that maximizes $J_n(t)$.

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

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

* also called steepest ascent or steepest descent.

Copyright ©2007 Stanley B. Gershwin.
Continuous Variables and Objective

Equality constrained: solution is on the constraint surface.

Problems are much easier when constraint is linear, i.e., 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

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

Minimize \(8(x + y) - \frac{(x^4 + y^4)}{4}\)

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

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.
Continuous Nonlinear and Linear Programming

Variables and Objective

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.
Continuous Variables and Objective

Multiple Optima

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

\[
\begin{align*}
\text{min } f(x) & \quad \text{max } j(x) \\
\text{subject to } j(x) & \geq J & \text{subject to } f(x) & \leq F
\end{align*}
\]

\(f(x), F, j(x),\) 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*. 
Continuous Variables and Objective

Generalization:

\[
\begin{align*}
\min f(x) & \quad \text{max } j(x) \\
\text{subject to } h(x) &= 0 \\
\quad g(x) &\leq 0 \\
\quad j(x) &\geq J \\
\end{align*}
\]

subject to \( h(x) = 0 \)

\( g(x) \leq 0 \)

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

**Primal problem:**

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

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

$N_i \geq N_{\text{MIN}}, i = 1, \ldots, k - 1.$

In the following, we treat the $N_i$s like a set of continuous variables.
Properties of $P(N_1, \ldots, N_{k-1})$

Buffer Space Allocation

\[ P(\infty, \ldots, \infty) = \min_{i=1,\ldots,k} e_i \]

\[ P(\underline{N}^{\text{MIN}}, \ldots, \underline{N}^{\text{MIN}}) \approx \frac{1}{k-1} < < P(\infty, \ldots, \infty) \]

\[ 1 + \sum_i \frac{p_i}{r_i} \]

- **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 $(N_1, \ldots, N_{k-1})$.
Buffer Space Allocation

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

Example — 3-machine line

\[ r_1 = .35 \quad r_2 = .15 \quad r_3 = .4 \]
\[ p_1 = .037 \quad p_2 = .015 \quad p_3 = .02 \]
\[ e_1 = .904 \quad e_2 = .909 \quad e_3 = .952 \]
Minimize
\[ \sum_{i=1}^{k-1} N_i \]
subject to
\[ P(N_1, \ldots, N_{k-1}) \geq P^* \]
\[ N_i \geq N_{\text{MIN}}, \quad i = 1, \ldots, k - 1. \]

**Difficulty:** If all the buffers are larger than \( N_{\text{MIN}} \), the solution will satisfy \( P(N_1, \ldots, N_{k-1}) = P^* \). *(Why?)* But \( P(N_1, \ldots, N_{k-1}) \) 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.
Maximize \( P(N_1, \ldots, N_{k-1}) \)

subject to

\[
\sum_{i=1}^{k-1} N_i \leq N^\text{TOTAL} \quad \text{specified}
\]

\( N_i \geq N^\text{MIN}, \; i = 1, \ldots, k - 1. \)

All the constraints are linear. The solution will satisfy the \( N^\text{TOTAL} \) 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

\[ N_1 + N_2 + N_3 = N \]

Constraint set (if \( N^{\text{MIN}} = 0 \)).
**Solution of the primal problem:**

1. Guess $N^{\text{TOTAL}}$.

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

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

Maximize

$$P(N_1, \ldots, N_{k-1})$$

subject to

$$\sum_{i=1}^{k-1} N_i = N_{\text{TOTAL}}$$ specified

$$N_i \geq N_{\text{MIN}}, i = 1, \ldots, k - 1.$$
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 $(N_1, \ldots, N_{k-1})$ 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 $(N'_1, \ldots, N'_{k-1})$ 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 $\geq N^\text{MIN}$).
● The gradient $g$ is the direction of greatest increase of $P$.
● The projected gradient $\hat{g}$ is the direction of greatest increase of $P$ within the constraint plane.
● Therefore, once we have a point $N$ on the constraint plane, the best improvement is to move in the direction of $\hat{g}$; that is, $N + A\hat{g}$.
● To find the best possible improvement, we find $A^*$, the value of $A$ that maximizes $P(N + A\hat{g})$. $A$ is a scalar, so this is a one-dimensional search.
● $N + A^*\hat{g}$ is the next guess for $N$, and the process repeats.
Find such that \( P(N + Ap) \) is maximized. Define
\[
\hat{N} = N + Ap
\]

Is \( \hat{N} \) close to \( N \)?

\( \text{YES} \) \( N \) is the solution. Terminate.

\( \text{NO} \) Set \( N = \hat{N} \)

Specify initial guess \( N = (N_1, \ldots, N_{k-1}) \) and search parameters.

Calculate gradient \( g \).

Calculate search direction \( p \).

(Here, \( p = \hat{g} \).)
Buffer Space Allocation

Solution

Dual Algorithm

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

![Diagram showing buffer space allocation solution]
Buffer Space Allocation

Solution

Primal algorithm

\[ P_{\text{MAX}}(N^{\text{TOTAL}}) \text{ as a function of } N^{\text{TOTAL}}. \]
Then, we can find, by 1-dimensional search, \( N^{\text{TOTAL}} \) such that

\[
P_{\text{MAX}}(N^{\text{TOTAL}}) = P^*.\]
Set $N^0 = (k-1)N^{\text{MIN}}$. Calculate $P_{\text{max}}(N^0)$. Specify initial guess $N^d$ and search parameters. Solve the dual to obtain $P_{\text{max}}(N^d)$. Set $j = 2$.

Calculate $N^j$ from (7).

Call the dual algorithm to evaluate $P_{\text{max}}(N^j)$.

Is $P_{\text{max}}(N^j)$ close enough to $P^*$?

YES

Minimum total buffer space is $N^j$.

Terminate.

NO

Increment $j$ by 1.

(Here, (7) refers to a one-dimensional search.)
• Problem: how to allocate space in a line with identical machines.
• Case: 20-machine continuous material line, \( r_i = 0.0952, p_i = 0.005 \), and \( \mu_i = 1, i = 1, \ldots, 20 \).

First, we show the average WIP distribution if all buffers are the same size: \( N_i = 53, i = 1, \ldots, 19 \).
Buffer Space Allocation

Example

The “Bowl Phenomena”

- This shows the optimal distribution of buffer space and the resulting distribution of average inventory.
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.
• 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.
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 ($P = .95$ parts per minute).
• **Case 1** MTTF = 200 minutes and MTTR = 10.5 minutes for all machines ($P = .95$ parts per minute).

• **Case 2** Like Case 1 except Machine 5. For Machine 5, MTTF = 100 and MTTR = 10.5 minutes ($P = .905$ parts per minute).
Buffer Space Allocation

Example

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

- **Case 2** Like Case 1 except Machine 5. For Machine 5, MTTF = 100 and MTTR = 10.5 minutes ($P = .905$ parts per minute).

- **Case 3** Like Case 1 except Machine 5. For Machine 5, MTTF = 200 and MTTR = 21 minutes ($P = .905$ parts per minute).
Are buffers really needed?

<table>
<thead>
<tr>
<th>Line</th>
<th>Production rate with no buffers, parts per minute</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>.487</td>
</tr>
<tr>
<td>Case 2</td>
<td>.475</td>
</tr>
<tr>
<td>Case 3</td>
<td>.475</td>
</tr>
</tbody>
</table>

Yes. *How were these numbers calculated?*
Buffer Space Allocation

Solution

Case 1 −− All Machines Identical
Case 2 −− Machine 5 Bottleneck −− MTTF = 100
Case 3 −− Machine 5 Bottleneck −− MTTR = 21

Buffer Size

Line | Space
---|---
Case 1 | 430
Case 2 | 485
Case 3 | 523

Copyright ©2007 Stanley B. Gershwin.
- This shows the optimal distribution of buffer space and the resulting distribution of average inventory for Case 3.
• This shows the ratio of *average inventory* to buffer size with optimal buffers for Case 3.
### Buffer Space Allocation

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

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

- **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.
• This shows the ratio of \textit{average inventory} to buffer size with optimal buffers for Case 6.

\begin{itemize}
\item \textbf{Buffer Space Allocation}
\end{itemize}
• Observation from studying buffer space allocation problems:
  ★ *Buffer space is needed most where buffer level variability is greatest!*
Profit as a function of buffer sizes

- Three-machine, continuous material line.
- \( r_i = 0.1, p_i = 0.01, \mu_i = 1. \)
- \( \Pi = 1000P(N_1, N_2) - (\bar{n}_1 + \bar{n}_2). \)