Review of Previous Class

- Basic Manufacturing Classifications
- Little’s Law
- Advantages of simplicity
Review: Little’s Law

Little’s Law: $WIP = \text{Production Rate} \times \text{Manufacturing Lead Time}$

Implications:
- If not near capacity, then increasing WIP increases rate without time increase. (Everything keeps busy).
- If near capacity, then rate cannot increase more – so increasing WIP increases throughput time!

Problems with Complexity

- Problems with Complexity:
  - Difficulty in maintaining and managing complexity
  - Decreased Reliability
  - “If you can’t make a simple machine work, then you can’t make a more complex one work.”
    - Throwing technology at problems without understanding them first may be a costly mistake.

- Law 3: The Larger the System Scope, the Less Reliable the System.
- Law 9: Combining, Simplifying, and Eliminating Save Time, Money, and Energy
**Review**

Classification of Discrete Manufacturing
- Mass Production
- Batch Production
- Job Shop

**Modeling**

- What do we mean by a model?
- A representation of a system
  - Used to understand the important factors and their relationships (Insight)
  - Used to predict performance
  - Used to optimize decision values
  - Used to determine control rules
  - Used to justify investments
- What kinds of models?
  - Physical models
  - Mathematical models
    - Deterministic, stochastic, simulation, …
Mathematical Models

- What is a good model?
  - Depends on what questions will be asked.
  - Analytical vs. Experimental
    - Experimental: describes, such as simulation
      - Good for “What if” questions
    - Analytical: determines an answer through analysis or heuristic
      - What is the lowest cost product mix?
      - What is the fastest throughput?

Learning Objectives on Serial Systems

- What are the characteristics of a serial system?
- What is the takt time?
- What is a minimum manning?
- What are the benefits/drawbacks of paced vs. unpaced lines?
- Given a set of tasks, how do we assign them to sequential workstations?
  - Show how this is done with COMSOAL method
  - Show how this is done with RPW method
  - Show how this is done with tree searches
  - [Show how this is done for mixed models]
Serial Assembly Systems

- Classic “assembly line”
- Good for single-product or restricted family of products.
- Advantage: Very efficient, low WIP, low cost
- Disadvantage:
  - requires: limited variation on times and types
  - Demands fairly good reliability

What is a Serial Assembly System?

- Product moves down assembly line.
- Division of work into work elements
  - Smallest unit of productive work
  - Examples:
- Work elements assigned to stations.
- Work done in sequence as it moves down line.
- History:
  - Chicago meat-packing plants → Henry Ford automotive
Taylorism (“Scientific Management”)

- F.W. Taylor: developer of “scientific management” (published in 1911)
- Taylorism assumes:
  - Mfg. enterprise can be subdivided into independent functions, tasks, and subtasks
  - Boundaries between mgt. Levels and functions should be well defined and tasks should be formally codified into workrules
  - Most efficient way to do work is to train each worker to do one task or subtask
  - There is one “best” way to accomplish a given task, and we can find it by time and motion studies
  - Organization should be directed from top by managers who have absolute control over every aspect of the business

Problems of Taylorism

- Taylorism fails to extract best from workers since it is based on restrictive notions of their abilities
- Poor flow of information
  - Organizational boundaries impede flow
  - Assumes info flows only top down
- Optimization of individual steps may not be optimum of whole process
- Hierarchy best only if static environment
  - Today’s environment is not static
- Neglects issues of quality, inventory, waste, overhead, etc.
  - Focuses primarily on labor costs
  - Reflected by traditional accounting rules that don't consider reducing costs that add no value
Side effects of Taylorism

- Problems of Taylorism:
  - Breakdown of tasks and functions leads to management hierarchy and sometimes extreme division of labor
    - Poor communication
  - Ignores advantages of synergism between functions
  - **Optimization of subfunctions does not imply global optimization**

- Side effects:
  - “Thinking” or “deciding” function lies with mgmt. only
    - No feedback from workers
  - “Quality function lies with separate group”
    - Quality is not responsibility of worker
    - Worker only must produce, keep line running

Efficiencies of Serial Assembly System

- Focus on single part type or family of part types
- Repetitive Operations
- Minimal setups
- Potential for high equipment and person utilization
- Standard rate (takt time) for production
- Minimal WIP
Issues with Serial Lines

• Balancing:
  – How to divide work between stations?

• Sequencing:
  – If multiple parts, then what order to process?

• Paced vs. Unpaced?

• Single line vs. Multiple lines:
  – Advantages of Multiple:
    • Multiple means longer work time per part to meet demand
    • Thus allows greater work content per station
    • Simplifies balancing, more scheduling flexibility
    • Added robustness/reliability
  – Possible Disadvantages:
    • more equipment and setup cost
    • Higher skill requirements

Basic Calculations

• TargetRate = Demand / period
• C = 1/(TargetRate)
• Note: Our book refers to C as the cycle time, but often it is referred to as the Takt Time

• Example:
  • 1000/week at 37 hours per week → 27/hour
  • C = .037 hours = 2.22 minutes each

• Other issues:
  – Line efficiency corrections
    • To correct for assembly, quality, equipment, labor issues etc.
    • Typical values: 90-98%
    • C = 2.22 x 95% = 2.11 minutes
Minimum Manning

- Manning/Stations:
  - $T_{wc} =$ total work content
  - $T_{wc}/C =$ number of workers (round up).
  - Example: $T_{wc} =$ 20 minutes. $\rightarrow$ 9 workers minimum

- Can we achieve this?
  - Losses:
    - Repositioning Losses:
    - Line Balancing Losses:
    - Task Time Variability:
    - Quality Problems:

- $\rightarrow$ Benefits in:
  - reducing wasted motion
  - reducing variability
  - ensuring consistent quality

- How do we best allocate tasks to stations?

How many lines?

- Task times must be less than Takt time, C
- If task times not fine enough, may be unable to fill C

- One possible solution: multiple parallel lines

<table>
<thead>
<tr>
<th>Table 2.1 Advantages/Disadvantages of Multiple Parallel Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advantages</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Easier to balance work load between stations</td>
</tr>
<tr>
<td>Increased scheduling flexibility</td>
</tr>
<tr>
<td>Job enrichment</td>
</tr>
<tr>
<td>Higher line availability—worker independence</td>
</tr>
<tr>
<td>Increased accountability</td>
</tr>
</tbody>
</table>
Paced vs. Unpaced?

• **Paced:**
  – Worker given strict time to complete work.
  – Synchronous work transport
  – Problem: variation in times, stress, incomplete work…
  – Possible solutions:
    • Extra time included
    • Work station boundaries extended
    • Small buffers to avoid starving / blocking
    • Problem response system / Andon cord

• **Unpaced / Asynchronous**
  – Workers work when product available, work until tasks done
  – Interesting result: (section 2.6):
    • Time in line and WIP is less for unpaced
  – Disadvantage: lack of timing feedback, lack of tying to demand rate
Other issues

• Layouts
  – Linear
  – Serpentine
  – U-shaped

• Buffer sizing
  – Cushion stations from each other
  – Depends on variation issues

• Production Control

• Parallel work at workstation

Example layouts of serial lines

• Straight line

• U-shaped

• Serpentine
Line Balancing Problem

- Line Balancing Problem:
  - How do we allocate tasks to stations?

- Concerns:
  - Ordering constraints
    - Example:
  - Zoning restrictions
    - Forbidden groupings
    - Required groupings

- Performance Criteria:
  Balance Delay = \( \frac{\text{total workstation time} - \text{total work content}}{\text{total workstation time}} \)

Where total workstation time = \( C \times (\# \text{ of stations}) \)
Balance Delay is idle time over paid time.

Example:

- Alarm Clock Construction
- Tasks:
  - A: attach cord to mechanism
  - B: add back to mechanism
  - C: add base through back
  - D: add knobs through back
  - E: add face to mechanism
  - F: add hands to face
  - G: add front glass
  - H: package

![Diagram of Alarm Clock Construction Tasks]
• Total Work Time: From diagram: = 2.5 min.
• Takt time: = 1 min
• Min. work manning: 2.5/1 → 3 people

Approaches to Line Balancing

• Approaches to Line Balancing

• Exhaustive Search
  – Problem: For N tasks, there are N! sequences
  – Example: for clock: N=8, N! = 4032

• Intelligent Search
  – As we explore, stop whenever we know we will do worse
  – Benefit: Optimal (within limitations of our model)
  – Example: later slide

• Heuristic
  – Ranked Positional Weight (RPW)
  – COMSOAL
  – Others...
  – No guarantee of best solution, but potentially good
Intelligent Search

Example:
- Explore branches of tree of possibilities – Depth first
- Load up open workstations before opening new
- Truncate a branch if not better than before
- Truncate branch if not feasible schedule
- (Other fathoming rules in book)

Ranked Positional Weights (RPW)

- For each node, sum up all times of all nodes following
  - This sum is the RPW for the node
  - Note: don't double count H for B.
- In order of decreasing RPW, assign nodes to stations

Example:
RPW Example: Cont.

<table>
<thead>
<tr>
<th>Work El.</th>
<th>RPW</th>
<th>Task time</th>
<th>Assign</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.75</td>
<td>.5</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1.25</td>
<td>.3</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0.95</td>
<td>.2</td>
<td>1</td>
</tr>
<tr>
<td>F</td>
<td>0.75</td>
<td>.25</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>0.6</td>
<td>.4</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>0.55</td>
<td>.35</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>0.5</td>
<td>.3</td>
<td>3</td>
</tr>
<tr>
<td>H</td>
<td>.2</td>
<td>.2</td>
<td>3</td>
</tr>
</tbody>
</table>

Note ordering of RPW ensures that predecessors always assigned first.
RPW gave optimal solution (3) this time – but no assurance in general.

RPW non-optimality

- Takt time = 10

- RPW says 3 stations
- Optimal is 2 stations
COMSOAL

- RPW was deterministic: ranked tasks then assigned in order
- Would we do better if randomly picked next task to assign?

- COMSOAL:
  - *Computer Method of Sequencing Operations for Assembly Lines*
  - Developed by Chrysler Corp.
  - Key ideas:
    - Iterate through large # of alternative sequences
    - Keep track of best sequence so far
    - Randomly select next task to add to sequence
    - (could weight random numbers to prefer certain criteria)

COMSOAL algorithm (sketch)

Given: N is number of sequences to consider

COMSOAL_Main(N)
{
  Initialize UnassignedSet as set of all tasks
  best_so_far is used to store best sequence found so far
  Define CurrStation as 1
  For N times: {
    Extend_Seq(UnassignedSet, CurrStation)
  }

  best_so_far is then your result
}
COMSOAL algorithm (sketch)

```
Extend_Seq(UnassignedSet, CurrStation) {
    Define FitSet ← tasks in UnassignedSet such that:
    predecessors aren't in UnassignedSet, and
    time(CurrStation)+time(task)≤ C
    If FitSet is nonempty {
        Randomly pick task from FitSet
        Remove task from UnassignedSet
        Add task to CurrStation
        If UnassignedSet not empty
            Call Extend_Seq(UnassignedSet, CurrStation)
        else { // all tasks have been assigned
            if better than best so far, then store it as best so far
        }
    } else {
        Close CurrStation, adding to total_idle time
        If total_idle_time for this sequence > best_so_far
            then start over with new sequence
        Create NewStation
        Call Extend_Seq(UnassignedSet, NewStation)
    }
}
```

COMSOAL example – Alarm Clock

<table>
<thead>
<tr>
<th>Unassigned</th>
<th>FitSet</th>
<th>Pick</th>
<th>Assign to</th>
<th>WS total time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Line Balancing Considerations

- Issues:
  - Did not consider *zoning* constraints or preferences
  - Spread out idle time, or concentrate it?
  - What if change in demand?
  - What about time variations?
    - Remove if possible
    - Add additional time?
    - Add rework?
    - Add line-stoppage method?

Modeling

- What we just saw was modeling of serial assembly lines
- We modeled to help in system design: allocation of tasks to workstation

- What other issues are reasons for modeling?
  - *Design / Modification*
  - *Control*
- What are typical measures of performance?
  - Lead time
  - Work in process (WIP)
  - Cash flow
  - Production rates
  - Flexibility
  - Return on Investment
  - Cost....
Classes of Models

Physical Models:
- Physical Experimentation: Actually build prototype
- Disadvantages: costly, time consuming
- Advantages: provides tremendous insight

Mathematical Models:
- Set of math equations or logical relationships describe the system.
  - Descriptive:
    - For given inputs, model gives outputs (e.g. performance)
    - Example: Simulation models
  - Prescriptive:
    - Model indicates how to set the inputs of the system
    - Example: Indicates what product mix should we use.
    - May be optimization methods, or heuristics

Major Classes of Models

Deterministic Models:
- Often simple, rough, quick and easy
- Good for finding unacceptable alternatives
- Example: “Does the system have enough capacity?”
- Example: “Is two shifts enough?”

Example: “Our machine takes 12 minutes per part for A, and 8 minutes per part for B. We have avg. demand of 30 of A per day, and 60 of B. How many shifts do we need minimum?”
Example: “Our machine takes 12 minutes per part for A, and 8 minutes per part for B. We have avg. demand of 30 of A per day, and 60 of B. How many shifts do we need minimum?”

1 shift = 8 hours \times 60 \text{ minutes} = 480 \text{ min. / shift}
2 shifts = 960 \text{ min.}

Required time:

- A: 12 min. \times 30/\text{day} = 360 \text{ min.}
- B: 8 \text{ min.} \times 60/\text{day} = 480 \text{ min.}

Total: 360 + 480 = 840 \text{ min. per day}

\text{(assumes no blocking, waiting, etc., and ignores variability)}

\rightarrow \text{At least two shifts per day needed.}

• Math Programming Models:
  – Examples: Linear Programming, Mixed-Integer Linear Programming, etc.
  – Optimization Problems: may be solved exact or through heuristics
  – Typical form:
    Maximize f()
    Subject to constraints….
LP Example

What is the best product mix (given max of 140 units of either).
• Price less materials for A: $550
• Price less materials for B: $500
• Overhead: $100,000 / month
• 336 hours / month

Max: Profit = 550*A + 500*B - 100,000
s.t.: M1 time <= 336
M2 time <= 336
M3 time <= 336
A <= 140
B <= 140

Solution of LP through Excel

(uses "Solver" add-in)

| profitA | 550 |
| profitB | 500 |
| Overhead | 100,000 |

Variables
- Time on M1: 134.4 hours = 2*A
- Time on M2: 336 hours = 2.5*B
- Time on M3: 336 hours = 2*A + 1.5*B

#Product A = 67.2
#Product B = 134.4

Objective: Net Profit = 4160 + prodA*profitA + prodB*profitB - Overhead

Constraints:
- Time on M1: 134.4 <= 336 hours
- Time on M2: 336 <= 336 hours
- Time on M3: 336 <= 336 hours
- prodA: 67.2 <= 140
- prodB: 134.4 <= 140

Note that mix of 68 / 133 gives profit of 3,900, and mix of 67 / 134 gives 3,850

This is found by the solver.
• Note LP solution not an integer
• LP: objective function to maximize is linear combination of decision variables

• Other Programming methods:
  – Integer program
  – Non-linear program
  – ...
  – Some require heuristics since search space is difficult

• Math Programming methods give solution
  – Depends on correct formulation of constraints
  – Depends on correct definition of the objective function
    • Are you maximizing what you should be?

Major Classes of Performance Anal. (cont.)

• Queueing Models
  – Probabilistic -- primarily steady state averages
  – Advantages:
    • Analytical solution to problems with randomness & uncertainty
    • --> fast solution (if solution exists or is known!)
  – Disadvantages:
    • Requires simplifying assumptions
    • Best for smaller models
    • solutions not always possible
    • most suited for steady state analysis
  – Examples:
    • Avg. work in process (WIP)
    • Avg. time in system
Major Classes of Performance Anal (cont.2)

• **Qualitative Models**
  – Qualitative Behavior examples:
    • Will a system ever reach deadlock?
    • Will a system ever reach this particular state?
    • Does step A *always* happen before step B?

  – Example models: process algebra, Petri nets, ...

  – Sometimes models can incorporate probability also (like Markov process): **Stochastic Petri nets**
    • How often does the product X get routed to machine A?


Major classes of Performance Anal (cont.3)

• **Simulation**
  – run a mathematical model of the system on the computer
  – **Advantages:**
    • sophisticated models
    • sophisticated questions can be asked
    • intuition helped by watching the simulation runs
  – **Disadvantages:**
    • can’t ask questions like “*Is it possible that ….*?”
    • time consuming: model development then multiple simulation runs to analyze
    • relationships aren’t as apparent as in analytical solutions.
Uses of Models

• Optimization:
  – Find the “best” set of decision variables.
  – Issues:
    • Approximations (in model and in solution)
    • The “right” objective function?
    • Are our constraints worthwhile?

• Performance Prediction:
  – Answer “What if...?”
  – Descriptive models: Requires modeler to generate good set of scenarios
  – Prescriptive Models: Can get sensitivity information

• Control:
  – Which control policies work best?
    Evaluate performance under different scenarios.

• Insight:
  – Thinking about a model requires thinking about the system

• Justification: Selling and demonstrating
Responsibility of the Modeler

- What is “effective” versus what is “efficient”?

- Models of a system are not unique: require many choices
  - What are the right choices?
  - Validation: System and model correspond
  - Verification: model and its implementation correspond.

- Models should Not be built to prove a point – since that is biased from the beginning.