
MFS605/EE605
Systems for Factory Information and Control

Lecture 2
Fall 2004

Larry Holloway
Dept. of Electrical Engineering and
Center for Robotics and Manufacturing Systems

1

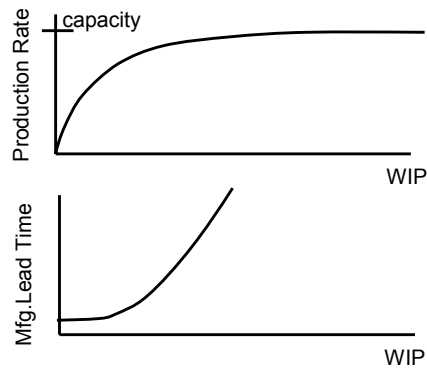
Review of Previous Class

- **Basic Manufacturing Classifications**
- **Little's Law**
- **Advantages of simplicity**

2

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!

3

Problems with Complexity

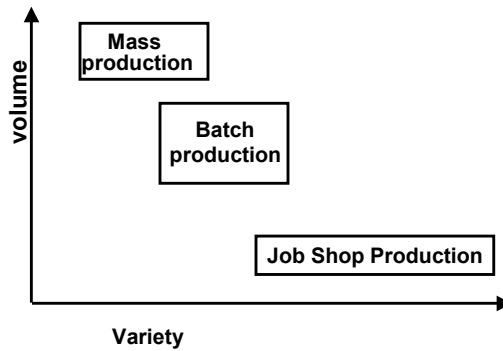
- Problems with Complexity:
 - Difficulty in maintaining and managing complexity
 - Decreased Reliability
 - “If you can’t make a simple machine work, then you can’t make a more complex one work.”
 - -- Throwing technology at problems without understanding them first may be a costly mistake.
- Law 3: The Larger the System Scope, the Less Reliable the System.
- Law 9: Combining, Simplifying, and Eliminating Save Time, Money, and Energy

4

Review

Classification of Discrete Manufacturing

- Mass Production
- Batch Production
- Job Shop



5

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, ...

6

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?

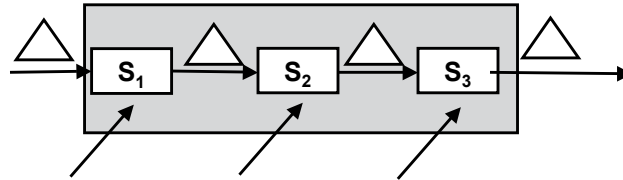
7

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]*

8

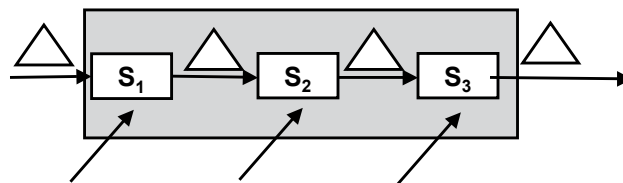
Serial Assembly Systems



- 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*

9

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

10

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

11

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

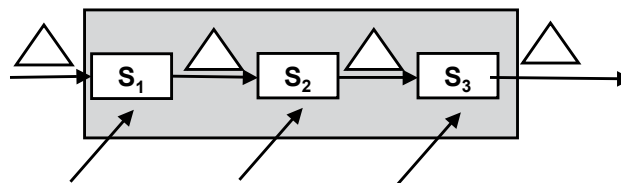
12

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 mgt. only
 - no feedback from workers
 - “Quality function lies with separate group”
 - quality is not responsibility of worker
 - worker only must produce, keep line running

13

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

14

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

15

Basic Calculations

- **TargetRate = Demand / period**
- **$C = 1/(\text{TargetRate})$**
- *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%
 - $C = 2.22 \times 95\% = 2.11$ minutes

16

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?**

17

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 2.1 Advantages/Disadvantages of Multiple Parallel Lines

Advantages	Disadvantages
Easier to balance work load between stations	Higher setup cost
Increased scheduling flexibility	Higher equipment costs
Job enrichment	Higher skill requirements
Higher line availability—worker independence	Slower learning
Increased accountability	More complex supervision

18

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

19

Unpaced example

20

Other issues

- **Layouts**
 - **Linear**
 - **Serpentine**
 - **U-shaped**
- **Buffer sizing**
 - **Cushion stations from each other**
 - **Depends on variation issues**
- **Production Control**
- **Parallel work at workstation**

21

Example layouts of serial lines

- **Straight line**
- **U-shaped**
- **Serpentine**

22

Line Balancing Problem

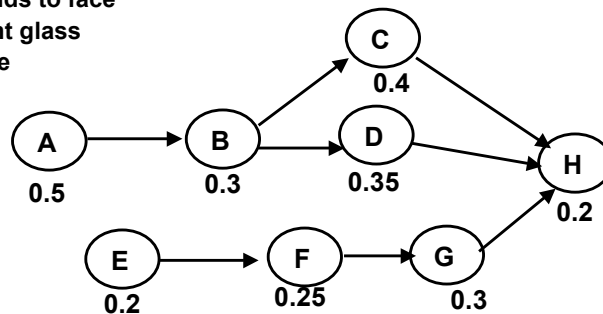
- **Line Balancing Problem:**
 - How do we allocate tasks to stations?
 - **Concerns:**
 - Ordering constraints
 - Example:
 - Zoning restrictions
 - Forbidden groupings
 - Required groupings
 - **Performance Criteria:**
Balance Delay = $\frac{\text{total workstation time} - \text{total work content}}{\text{total workstation time}}$

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

23

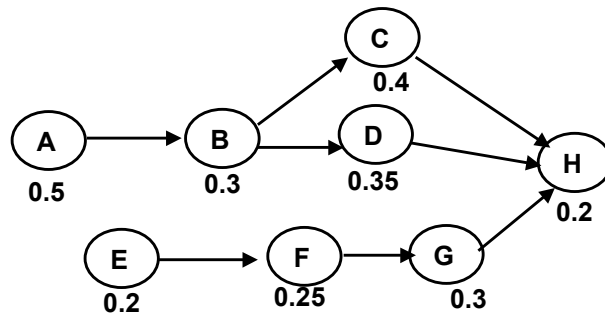
Example:

- **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



24

- Total Work Time: From diagram: = 2.5 min.
- Takt time: = 1 min
- Min. work manning: $2.5/1 \rightarrow 3$ people



25

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

26

Intelligent Search

Example:

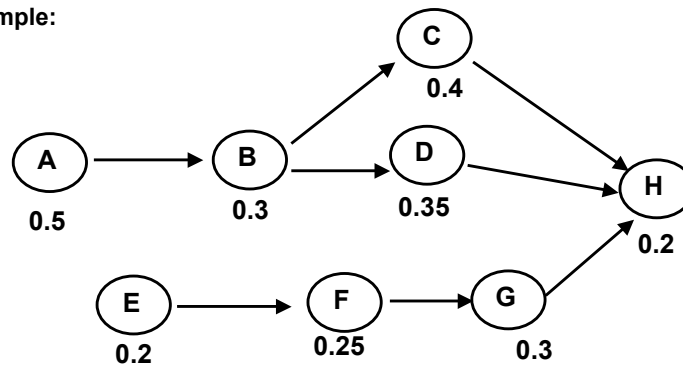
- Explore branches of tree of possibilities – Depth first
- Load up open workstations before opening new
- Truncate a branch if not better than before
- Truncate branch if not feasible schedule
- (Other fathoming rules in book)

27

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:



28

RPW Example: Cont.

Work El.	RPW	Task time	Assign	
A	1.75	.5	1	
B	1.25	.3	1	
E	0.95	.2	1	
F	0.75	.25	2	
C	0.6	.4	2	
D	0.55	.35	2	
G	0.5	.3	3	
H	.2	.2	3	

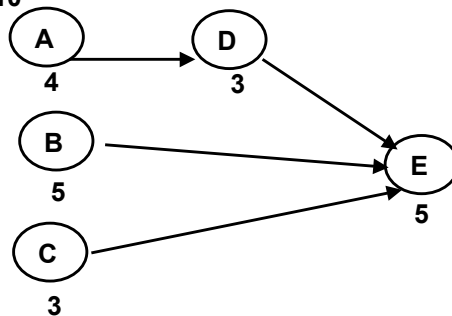
Note ordering of RPW ensures that predecessors always assigned first.

RPW gave optimal solution (3) this time – but no assurance in general

29

RPW non-optimality

- Takt time = 10



- RPW says 3 stations
- Optimal is 2 stations

30

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)

31

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
}
```

32

COMSOAL algorithm (sketch)

```

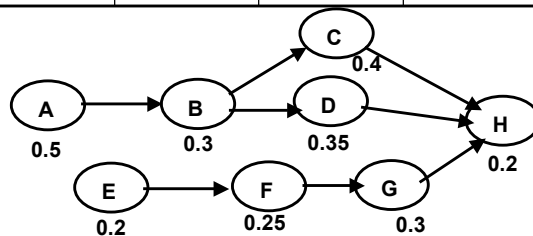
Extend_Seq( UnassignedSet, CurrStation)  {
  Define FitSet ← tasks in UnassignedSet such that:
    predecessors aren't in UnassignedSet, and
    time(CurrStation)+time(task) ≤ C
  If FitSet is nonempty {
    Randomly pick task from FitSet
    Remove task from UnassignedSet
    Add task to CurrStation
    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)
  }
}

```

33

COMSOAL example – Alarm Clock

Unassigned	FitSet	Pick	Assign to	WS total time



34

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?

35

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....

36

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

37

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?”

38

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 = 8hours*60 minutes = 480 min. / shift
 - 2 shifts = 960 min.
 - Required time:
 - A: 12 min. * 30/day = 360 min.
 - B: 8 min. * 60/day = 480 min.
 - Total: 360+480 = 840 min. per day
(assumes no blocking, waiting, etc., and ignores variability)
- > At least two shifts per day needed.

39

-
- **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....

40

-
- **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?

43

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

44

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?

45

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.

46

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**

47

-
- **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**

48

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.