Scheduling

Dr Feryal Erhun
fe251@cam.ac.uk

Recap of Last Lecture

- There are different order fulfilment strategies dependent on the P:D ratio
- Reliable forecasting is critical for many organisations at both strategic and operational levels
  - ➢ The forecast is always wrong!
  - ➢ The longer the forecast horizon, the worse the forecast
  - ➢ The less aggregated, the worse the forecast
- Forecasting methods can be quantitative or qualitative
  - ➢ Qualitative methods where little historical data available
  - ➢ Quantitative methods can be: extrapolation methods (ES, MA); regression analysis; time series, decomposition analysis
  - ➢ Each comes with its own limitations

Objectives for Today

- Develop an understanding of how to schedule machines in a job shop environment
- Be able to "balance" the design of an assembly line
- Understand the advantages and disadvantages of different physical layouts

Scheduling

Scheduling Problems

- General question now: how to calculate ***time-phased requirements***?
- This is different from ***consumption-based methods***, such as Reorder Point (ROP)

N=1 Project management

N= low volume, high variety – Job shop

- Single machine (Moore's algorithm)
- Sequential machines (Johnson's rule)
- Parallel machines

N= high volume / batch production, standard products with options

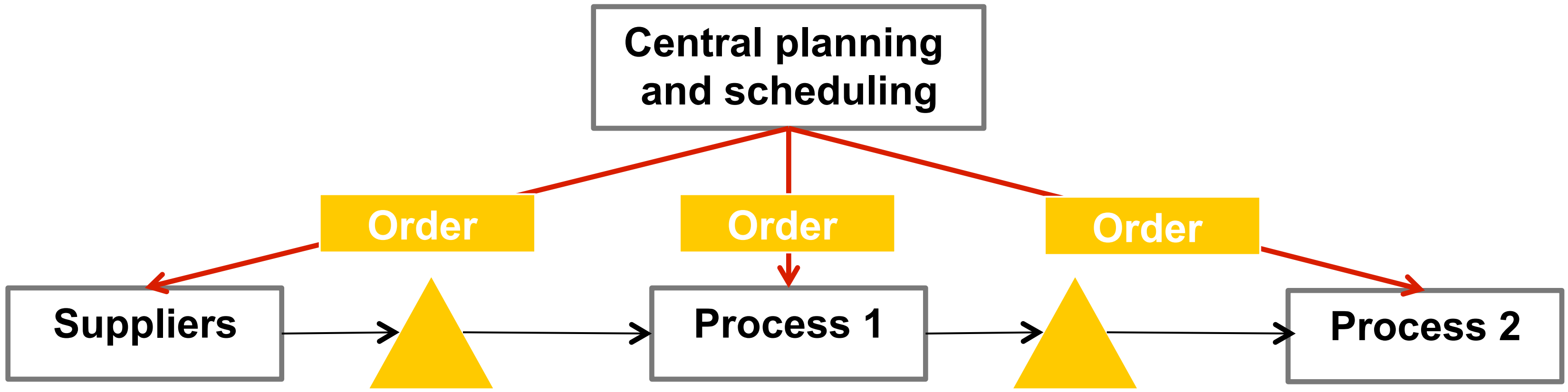
- MRP (Push)
- JIT (Pull)

N= very high volume, few or no variants

- Control theory

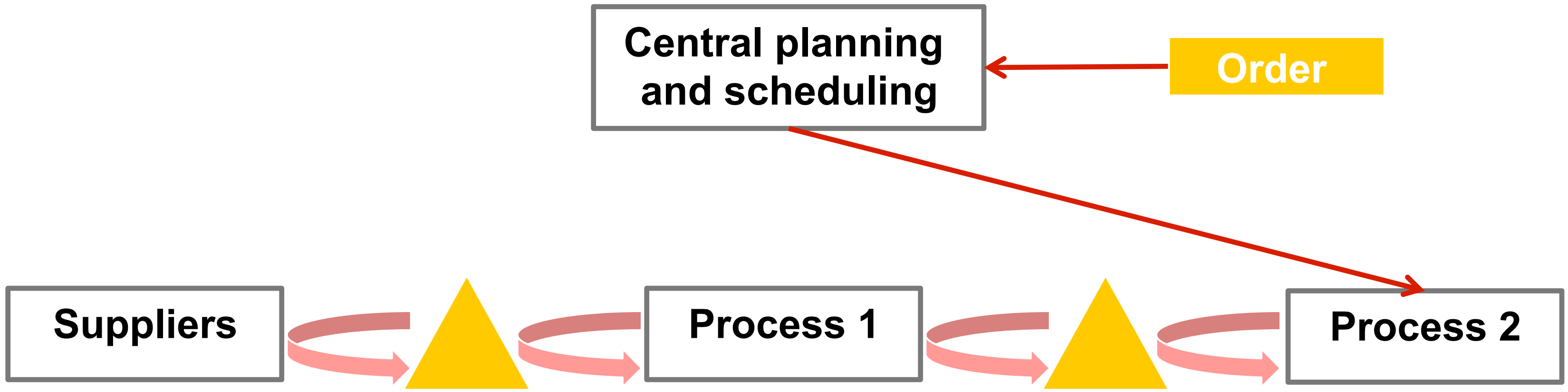
Associated scheduling problem: assembly line balancing

"Push" Scheduling



- In a **push** system, the orders are planned and issued centrally
- Upon completion, the order is moved forward, until the next process is issued with the order start processing it
- Hence, the **longest time** process sees the new order **first**
- This is called **backward** scheduling

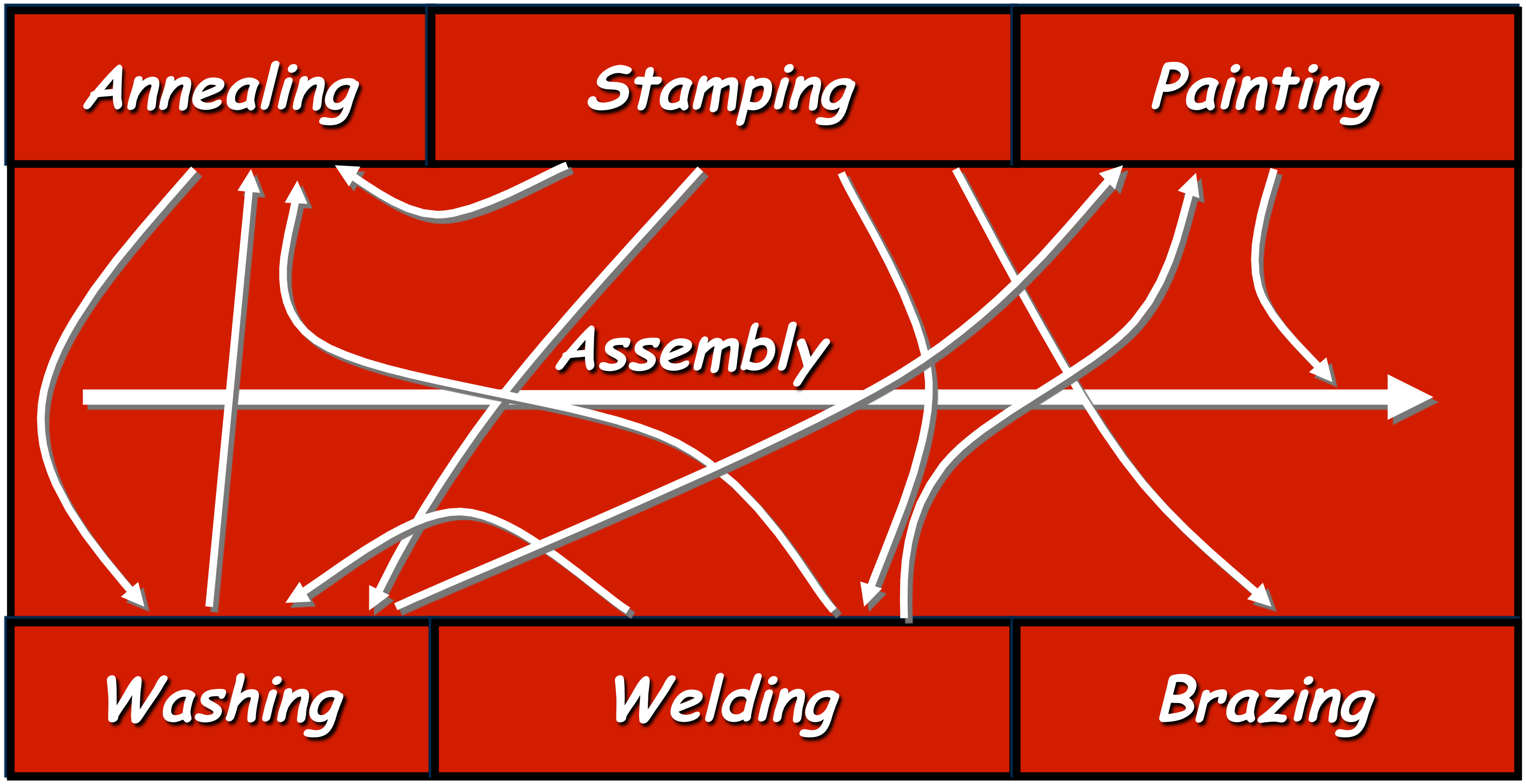
“Pull” Scheduling



- In a **pull** system, processes are triggered by a replenishment signal
- Upon withdrawal of material from inventory, the preceding process is authorised to start processing, and **only then**!
- Hence, the **final** process sees the new order **first**
- This is called **forward** scheduling
- Simplest form: two-bin approach

Job Shop Scheduling

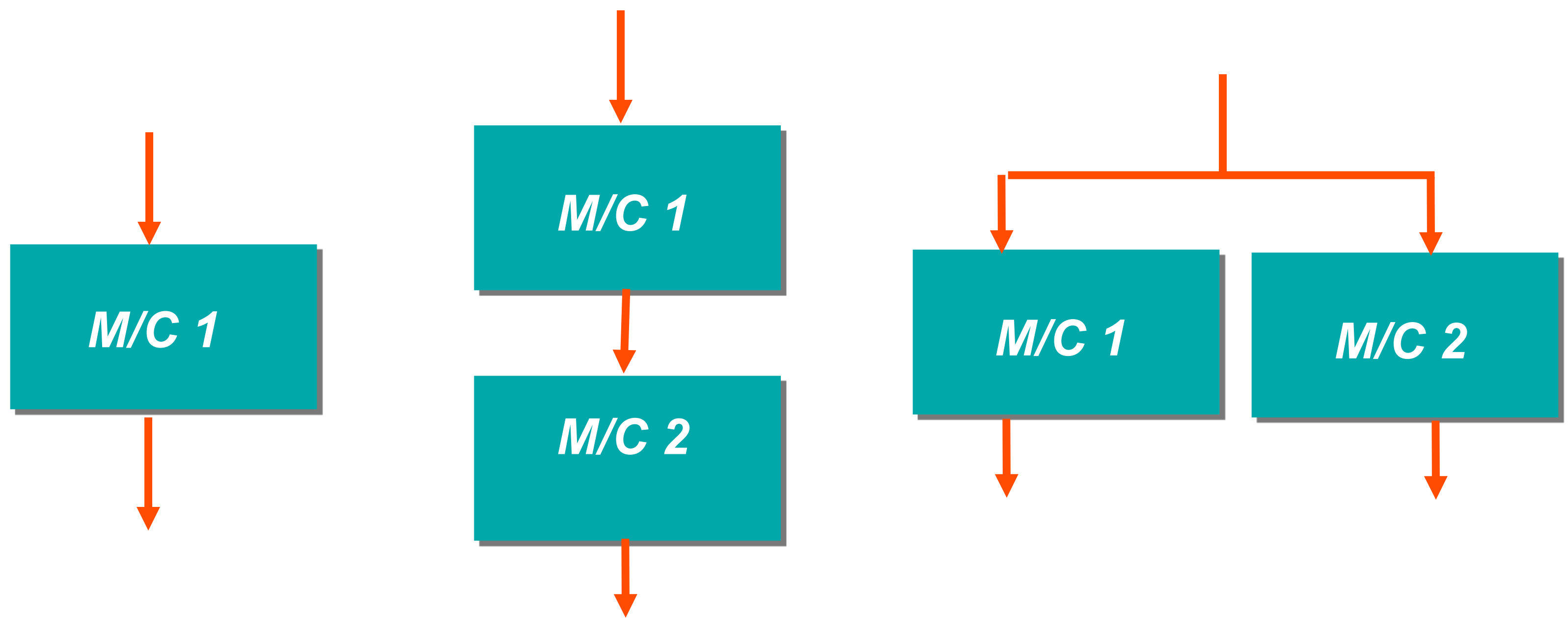
Job Shop - Flow Chart



Process split into independent work centres, complex routing and scheduling

Types of Problems in Job Shop

- In a job shop situation, different jobs have different routings and different processing times
- Analysis of complex cases is often based on concepts from simple cases such as just one machine, or two sequential machines, or two parallel machines



Scheduling Objectives

The objectives we might face in scheduling are:

- Minimise average completion/flow time of a set of jobs
- Minimise the maximum lateness
- Minimise the number of late jobs
- Minimise average tardiness
- Minimise "makespan" (time between first job starting and last job finishing)

For each objective, an algorithm or heuristic exists to help us achieve our objective

- Algorithm: gives us the optimal solution
- Heuristic: gives us a reasonable, but not necessarily optimal solution

Job Shop Scheduling with One Machine

Minimising Average Completion Time (1 m/c)

Job	A	B	C	D
Processing Time	11	3	4	2

- If we do jobs in alphabetical order, and start at t=0, the schedule is:

  $A_{11}B_{14}C_{18}D_{20}$ NB useful notation: $\text{Job}_{\text{Completion time}}$
- The *overall* completion time (20) does not depend on the schedule, but we might want to minimise *average* completion time
- For our schedule, average completion time = (11+14+18+20)/4 = 15.75
- We can do better: put jobs with **S**hortest **P**rocessing **T**imes (SPT) earlier:

  $D_2B_5C_9A_{20}$
- For new schedule, average completion time = (2+5+9+20)/4 = 9
- **SPT Rule:** minimises average completion/flow time on one machine
- Flow time = time that a job spends in the system

Minimising Maximum Lateness (1 m/c)

Job	A	B	C	D
Processing Time	3	3	4	5
Due Date	9	8	16	9

- If we do jobs in alphabetical order, the schedule is:

  $A_3B_6C_{10}D_{15}$
- For this schedule, maximum lateness (job D) = 15 – 9 = 6
- For this problem, it is *impossible* to avoid some job being late
- We might want to minimise the maximum lateness: to do this, put the jobs in order of **E**arliest **D**ue **D**ate (EDD):

  $B_3A_6D_{11}C_{15}$
- For new schedule, maximum lateness (job D) = 11-9 = 2
- **EDD Rule:** minimises maximum lateness on one machine

Minimising the Number of Late Jobs (1 m/c)

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33

- For this problem, it is *impossible* to avoid some jobs being late
- We might want to minimise the number of late jobs (e.g., to minimise the cost of subcontracting them)
- **Moore's Algorithm:**
  1. Schedule jobs by EDD
  2. If no job is late, goto step 6
  3. Find the first late job (call it the k^{th} job)
  4. From amongst jobs 1 to k, remove the job with the longest processing time
  5. Return to step 1 with one fewer job to consider
  6. The schedule is the EDD schedule, plus removed jobs (in any order)

Moore's Algorithm I

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33

1. **Schedule jobs by EDD:** Jobs already listed in EDD order, so start with this schedule

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Completion Time	2	9	12	20	24				
Due Date	5	10	15	22	23	24	25	30	33

2. **If no job is late, goto step 6:** There are late jobs
3. **Find the first late job (call it the k^{th} job):** Job E is the first late job
4. **From amongst jobs 1 to k, remove the job with the longest processing time:** Of the jobs up to E, the longest is D, so remove job D and start again

Moore's Algorithm II

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33

1. **Schedule jobs by EDD:** Jobs already listed in EDD order, so start with this schedule

Job	A	B	C	E	F	G	H	I
Processing Time	2	7	3	4	4	6	8	5
Completion Time	2	9	12	16	20	26		
Due Date	5	10	15	23	24	25	30	33

2. **If no job is late, goto step 6:** There are late jobs
3. **Find the first late job (call it the k^{th} job):** Job G is the first late job
4. **From amongst jobs 1 to k, remove the job with the longest processing time:** Of the jobs up to G, the longest is B, so remove job B and start again

Moore’s Algorithm III

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33

1. **Schedule jobs by EDD:** Jobs already listed in EDD order, so start with this schedule

Job	A	C	E	F	G	H	I
Processing Time	2	3	4	4	6	8	5
Completion Time	2	5	9	13	19	27	32
Due Date	5	15	23	24	25	30	33

2. **If no job is late, goto step 6:** There are no late jobs
6. **The schedule is the EDD schedule, plus removed jobs (in any order)**

Moore's Algorithm IV

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33

Final schedule:

Job	A	C	E	F	G	H	I	B	D
Processing Time	2	3	4	4	6	8	5	7	8
Completion Time	2	5	9	13	19	27	32	39	47
Due Date	5	15	23	24	25	30	33	10	22

Moore's Algorithm: minimises the number of late jobs on one machine

Minimising Average Tardiness

- Observation: minimising average lateness is a daft idea!
- Lateness = (completion date) – (due date)
  hence it could be negative
- But in real life, there is often little/no benefit from finishing a job early
- Instead, define tardiness of a job = maximum of 0 and the Lateness of the job (we write: max(0,Lateness))
- Minimising average tardiness is a meaningful objective
- How do we achieve it?

Minimising Average Tardiness

- We could try EDD; if there is only one late job, EDD minimises the average tardiness
- But for more than one late job, there is <u>no algorithm</u> for this problem
- However, there is a heuristic: the **M**odified **D**ue **D**ate (MDD) Rule: put jobs in order of increasing MDD
- At time t, MDD = the maximum of the due date and the earliest time at which we can complete a job:

i.e., $MDD_i = \max(d_i, t+p_i)$
where d_i = due date, p_i = processing time for job i

MDD Example I

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33

MDD Example II

- At t=0, Modified Due Date = Due Date for each job j:

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date	**5**	**10**	**15**	**22**	**23**	**24**	**25**	**30**	**33**

- Here, A has the smallest MDD (5), and completes at t=2: A_2

MDD Example III

- t=2

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date		**10**	**15**	**22**	**23**	**24**	**25**	**30**	**33**

- Here, B has the smallest MDD (10), and completes at t=9: A_2B_9

MDD Example IV

- t=9

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date			**15**	**22**	**23**	**24**	**25**	**30**	**33**

- Here, C has the smallest MDD (15), and completes at t=12: $A_2B_9C_{12}$

MDD Example V

- t=12

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date				**22**	**23**	**24**	**25**	**30**	**33**

- Here, D has the smallest MDD (22), and completes at t=20: $A_2B_9C_{12}D_{20}$
- So far, all jobs have had MDD=DD, but time is moving on…

MDD Example VI

- t=20

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date					**24**	**24**	**26**	**30**	**33**

- Here, E has the smallest MDD (24), and completes at t=24: $A_2B_9C_{12}D_{20}E_{24}$
- … and yes, we could have chosen F instead!

MDD Example VII

- t=24

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date						**28**	**30**	**32**	**33**

- Here, F has the smallest MDD (28), and completes at t=28: $A_2B_9C_{12}D_{20}E_{24}F_{28}$

MDD Example VIII

- t=28

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date							**34**	**36**	**33**

- Here, I has the smallest MDD (33), and completes at t=33: $A_2B_9C_{12}D_{20}E_{24}F_{28}I_{33}$

MDD Example IX

- t=33

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date							**39**	**41**	

- Here, G has the smallest MDD (39), and completes at t=39: $A_2B_9C_{12}D_{20}E_{24}F_{28}I_{33}G_{39}$
- When will all parts be finished?

MDD Example X

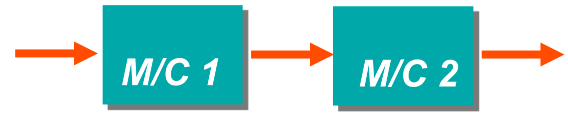
- t=39

Job	A	B	C	D	E	F	G	H	I
Processing Time	2	7	3	8	4	4	6	8	5
Due Date	5	10	15	22	23	24	25	30	33
Modified Due Date								**47**	

- How to measure the quality of my schedule?
- Final Schedule: $A_2\ B_9\ C_{12}\ D_{20}\ E_{24} F_{28}\ I_{33}\ G_{39}\ H_{47}$
- Original Due Dates: $A_2\ B_{10}\ C_{15}\ D_{22}\ E_{23} F_{24}\ I_{33}\ G_{25}\ H_{30}$
- Total tardiness is 36, the average tardiness is 36/9 = 4.

Job Shop Scheduling
With More Than One Machine

Job Shop Scheduling with More Than One Machine

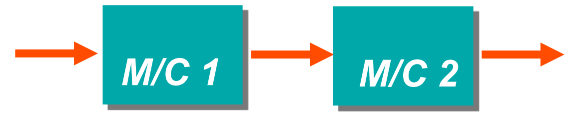


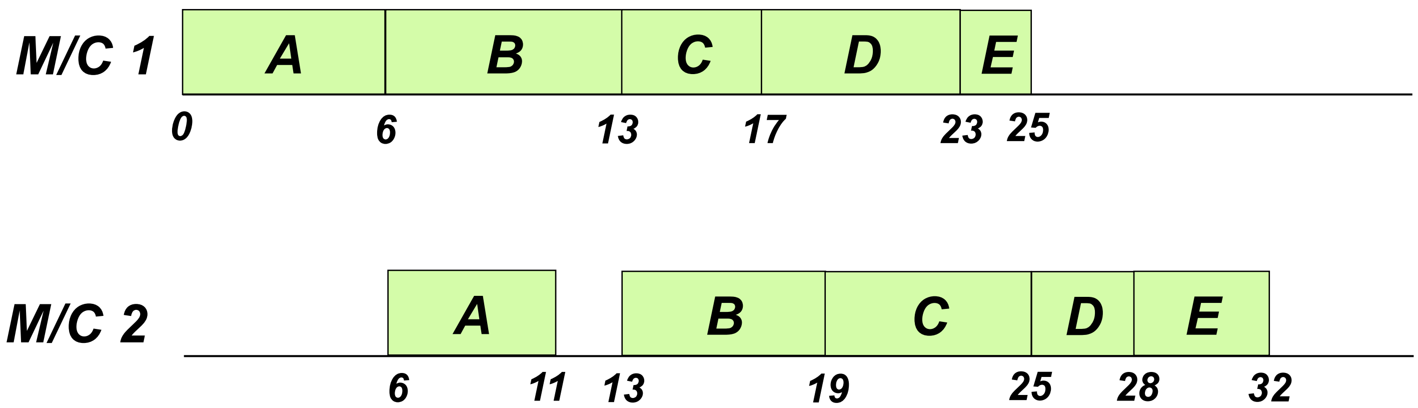
- Forget due dates
- **Makespan:** time between first job starting and last job finishing
- **Flowshop:** every job visits the same machines in the same order
- Problem: Minimise makespan in a 2-machine flowshop

Jobs	**A**	**B**	**C**	**D**	**E**
Processing Time - m/c 1	6	7	4	6	2
Processing Time - m/c 2	5	6	6	3	4

Try ABCDE …

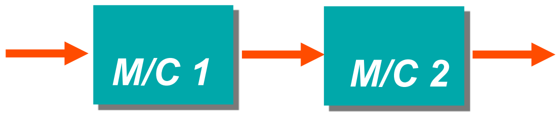
Jobs	A	B	C	D	E
Processing Time - m/c 1	6	7	4	6	2
Processing Time - m/c 2	5	6	6	3	4





- Makespan = 32. Can we do better?

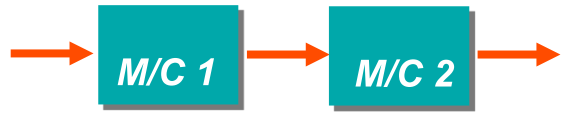
Minimising Makespan



Observations

- The makespan on m/c 1 is fixed
- Thus, to minimise overall makespan, we need to minimise makespan on m/c 2
- Two ways to do that
  - Minimise amount of time m/c 2 continues to process beyond m/c 1
  - Minimise amount of time m/c 2 needs to wait before starting to process jobs
- Conclusion:
  - Last jobs in schedule should have short processing times on m/c 2
  - First jobs in the schedule should have short processing times on m/c 1

Johnson's Rule I



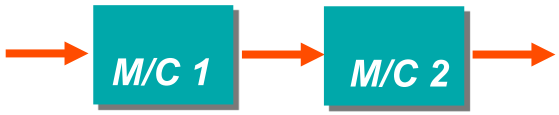
1. From the jobs not yet assigned, find the job with shortest processing time on either m/c

2. Which machine does that processing time occur on?
   - If that shortest processing time occurs on m/c 1, assign the job to the next free slot in the schedule
   - If that shortest processing time occurs on m/c 2, assign the job to the last free slot in the schedule

3. Go back to Step 1 until all the jobs are assigned

Johnson’s Rule II

Jobs	A	B	C	D	E
Processing Time - m/c 1	6	7	4	6	2
Processing Time - m/c 2	5	6	6	3	4

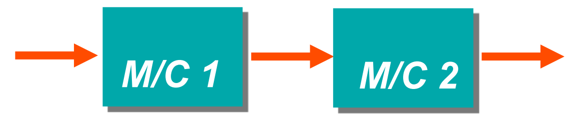


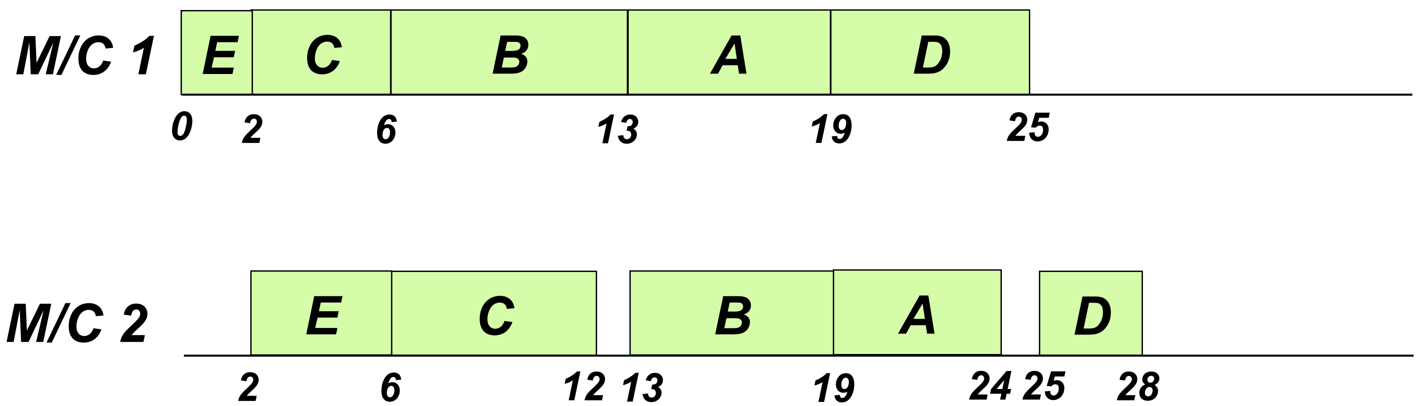
- Job E (2) on m/c 1 → start
- Job D (3) on m/c 2 → end
- Job C (4) on m/c 1 → start
- Job A (5) on m/c 2 → end
- Job B → between jobs already allocated

This yields the schedule: ECBAD

Johnson’s Rule III

Jobs	A	B	C	D	E
Processing Time - m/c 1	6	7	4	6	2
Processing Time - m/c 2	5	6	6	3	4





Makespan = 28

Johnson’s Rule: minimises makespan in a two machine flowshop

Summary of Scheduling Rules

- **SPT Rule:** minimises average completion/flow time on one machine
- **EDD Rule:** minimises maximum lateness on one machine
- **Moore's Algorithm:** minimises the number of late jobs on one machine
- **MDD Rule:** (heuristic) minimises average tardiness on one machine
- **Johnson's Rule:** minimises makespan in a two machine flowshop

- **Flow time** = time that a job spends in the system
- **Lateness** = (completion date) – (due date)
- **Tardiness** = max(0, lateness)
- **Makespan** = time between first job starting and last job finishing

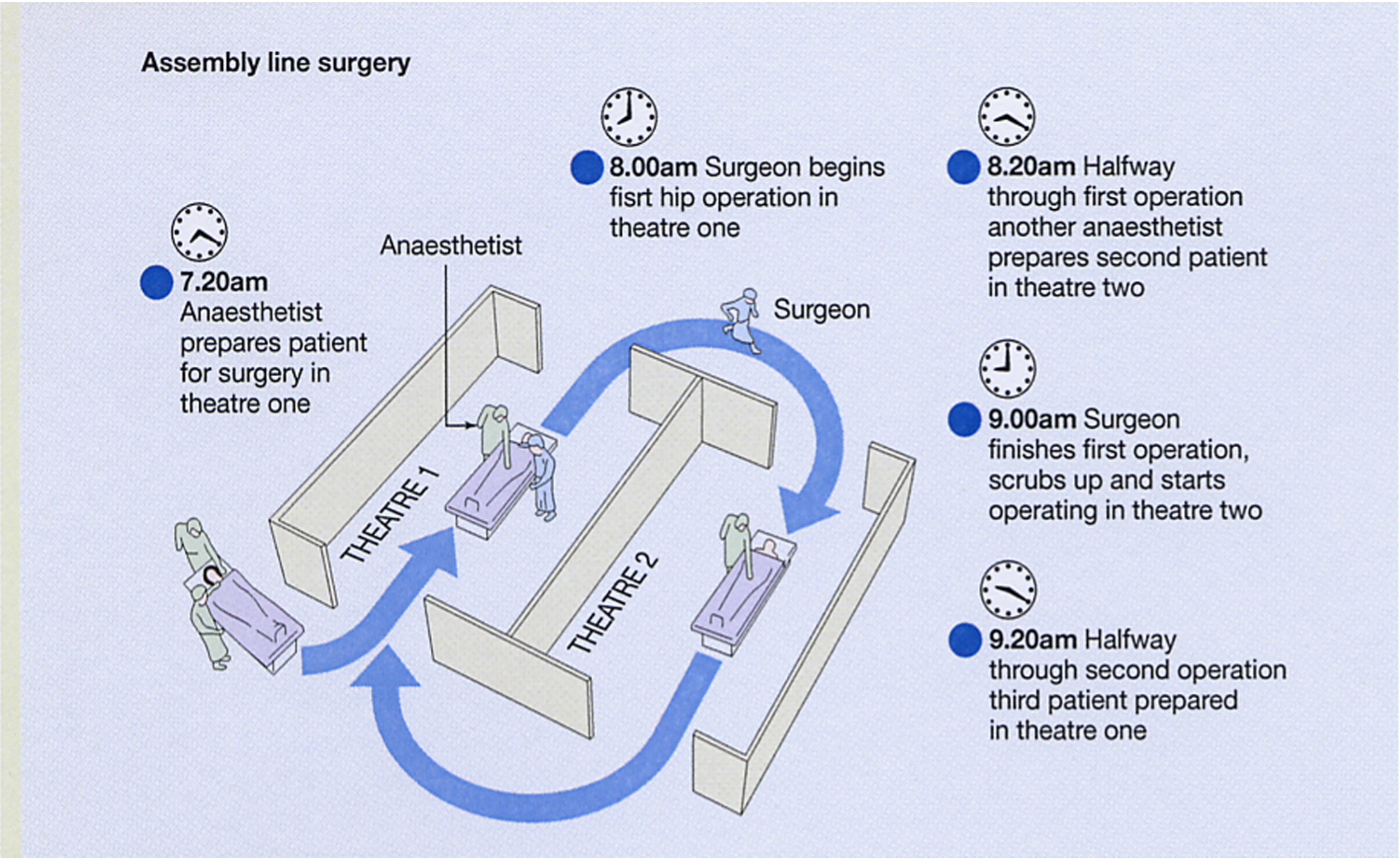
Assembly Line Balancing

Assembly Lines

- An assembly line consists of a sequence of operations such as fitting of components, painting, or inspection
- Some operations modify a single part or item; other operations assemble two or more parts together
- Could be moving assembly lines, in assembly cells, or stationary in groups
- Machine-related (hard) and workforce-related (soft) aspects
- Some assembly lines are coupled with automated or at least semi-automated stations
- Lines generally run at a fixed speed: “Takt time”
  - Takt time = demand per period [units] / available time per period [sec]
  - Thus, every “X seconds, a product is finished”
  - Typical Takt time in automotive: 40-70 sec for volume cars

Traditional Assembly Line

Assembly Line Surgery



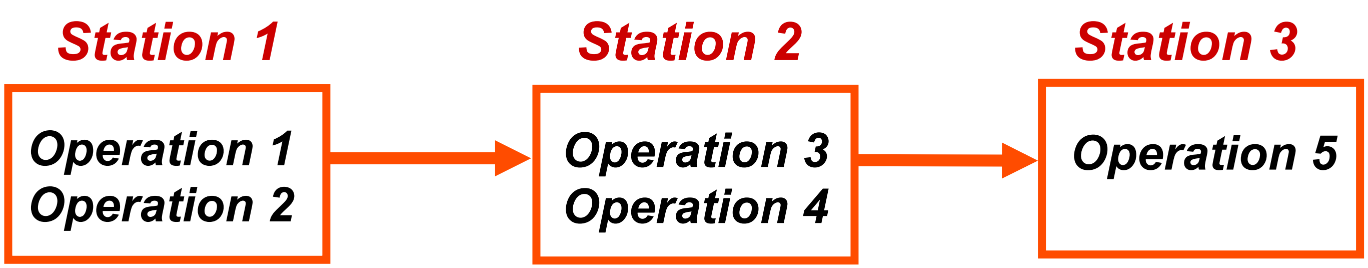
Source: Slack et al, Operations Management: Prentice-Hall

Assembly Line Balancing

- The technique of assigning work to individual stations so that the variability across all stations in an assembly line is minimized
- Challenge: ensuring that there is an even flow of work

Operations and Stations

- Crucial question: how many people do we need to do the work?
- Naïve answer: n operations need n people
- Problem:
  - ➢ Some operations may take less time than others, so some operators will be idle part of their time
  - ➢ We might not need to produce items at such a high rate
- Thus, one person can cover more than one operation
- Hence we group operations into Stations



Balancing the Assembly Line

- Ideally, the time required for stations to carry out their operations should not vary much from one station to another, otherwise:
  - some stations will be partly idle
  - and/or some stations will be overloaded

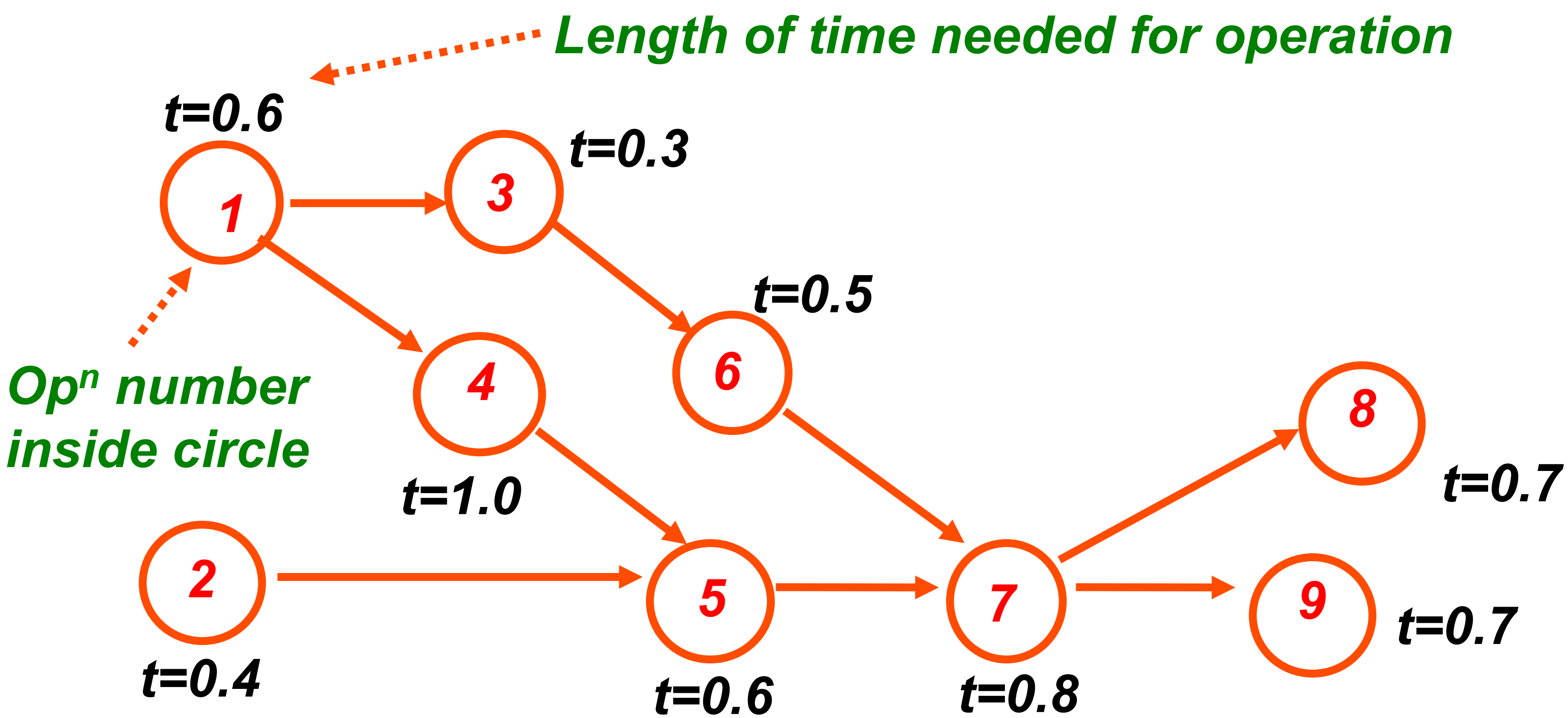
- We say we need to “balance” the assembly line

Balancing the Assembly Line

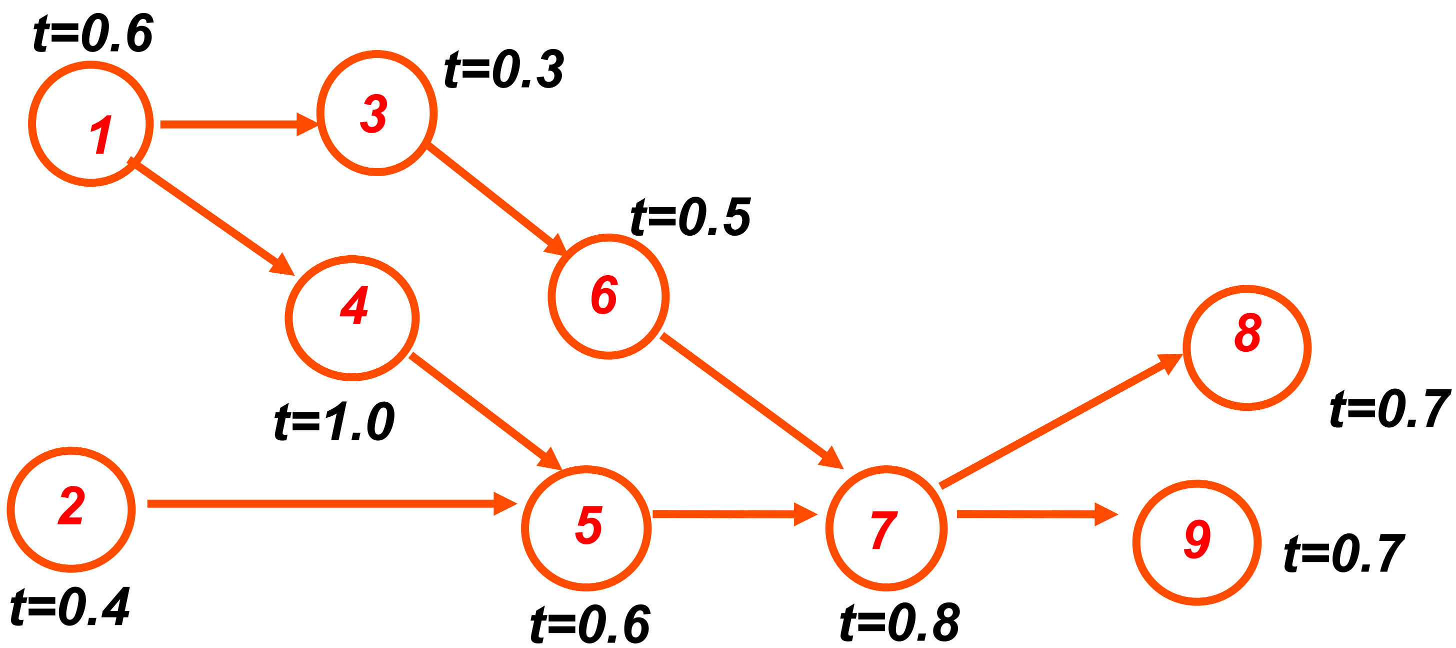
- We ask each station to complete its work within a fixed time, known as the **cycle time**
- Then we allocate operations to stations, ensuring that:
  - each station can complete its operations with the cycle time
  - there is a similar amount of idle time at each station
- NB: the cycle time is also the time between successive items reaching the end of the assembly line
- Hence by reducing the cycle time, we increase the production rate
- **"Takt time"** = adjusting the cycle time needed to meet customer demand

Precedence Constraints

- Some operations cannot be done before others
- Arrow means that one operation has to be completed before another can start

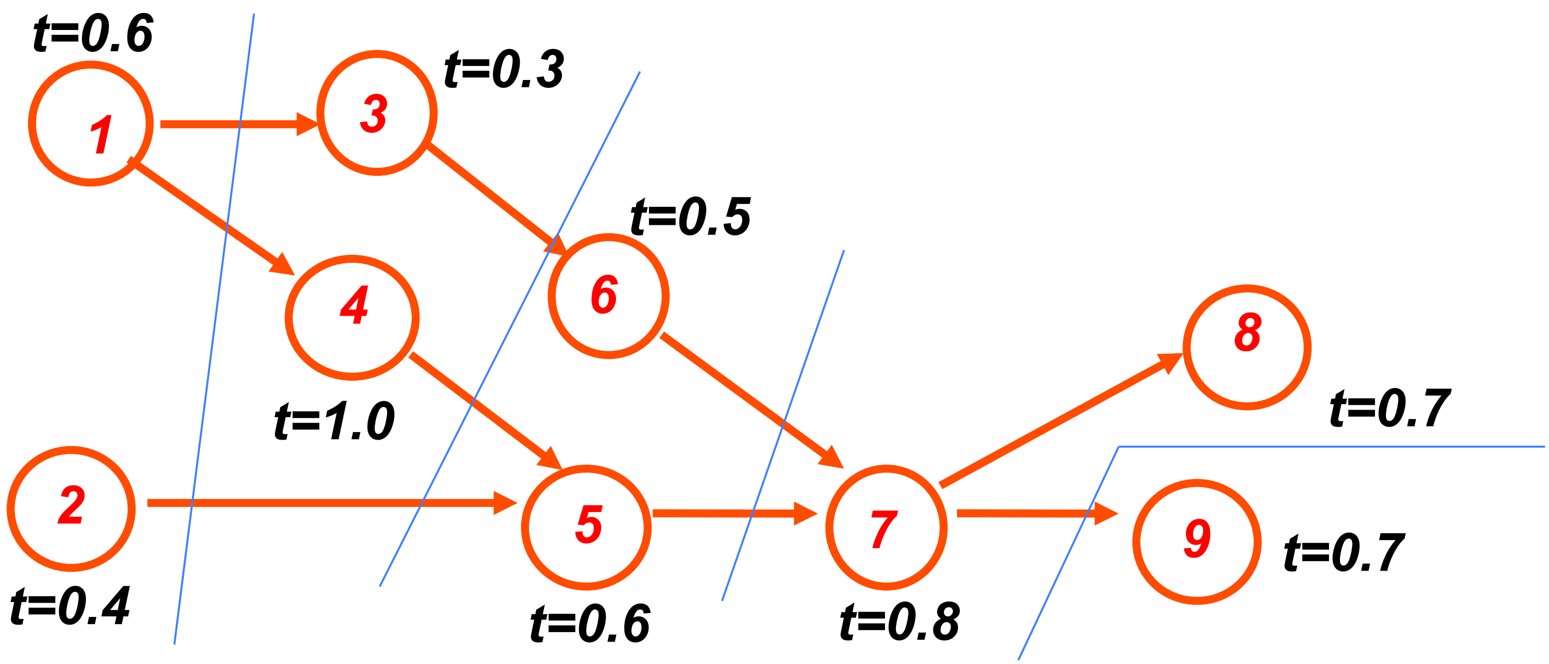


Example



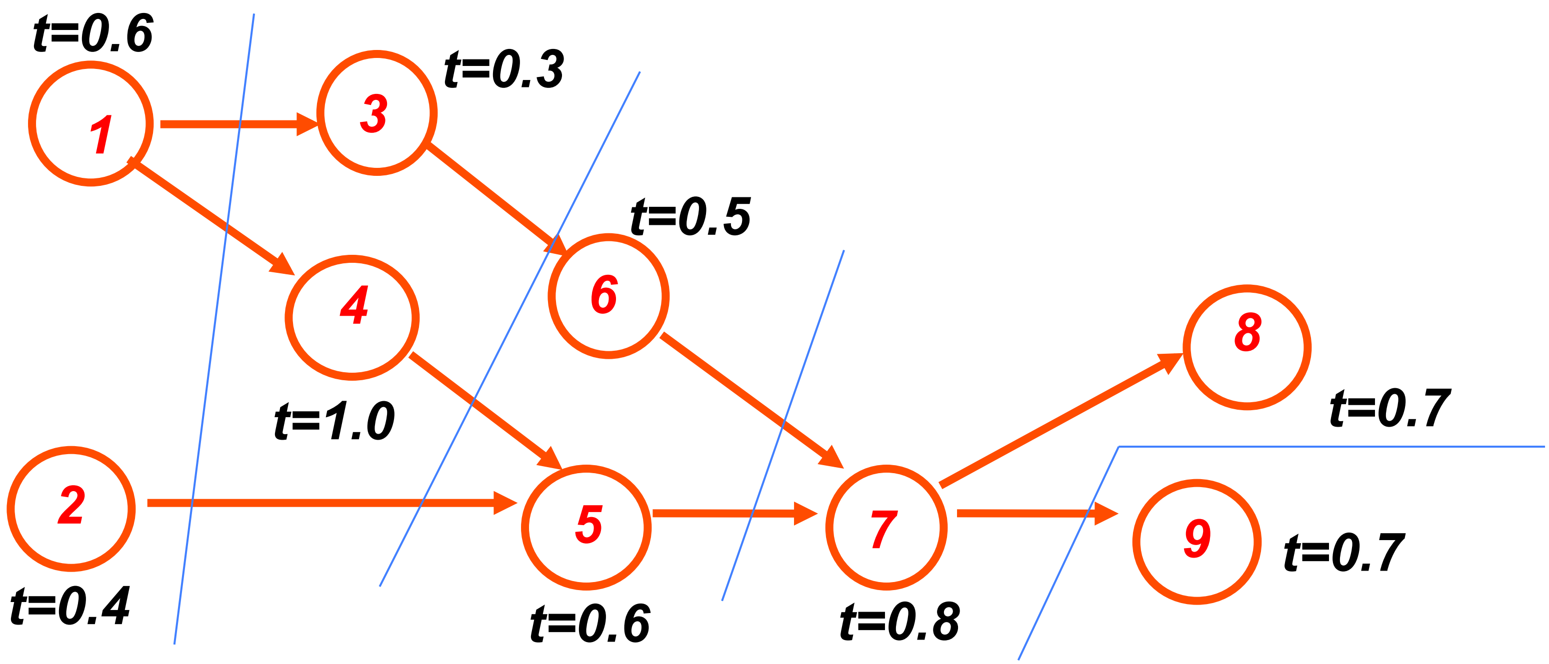
- Suppose we want to produce 40 items/hour
- 60min per hour / 40 items per hour = 1.5 min cycle time
- Work content = sum of all operation (processing) times
  = 5.6 minutes
- Minimum number of stations 5.6min / 1.5min = 3.73 (i.e., 4 stations)

A Possible Allocation of Operations I



Station	Operations	Station Time
1	1,2	1 min
2	3, 4	1.3 mins
3	5,6	1.1 mins
4	7, 8	1.5 mins
5	9	0.7 mins

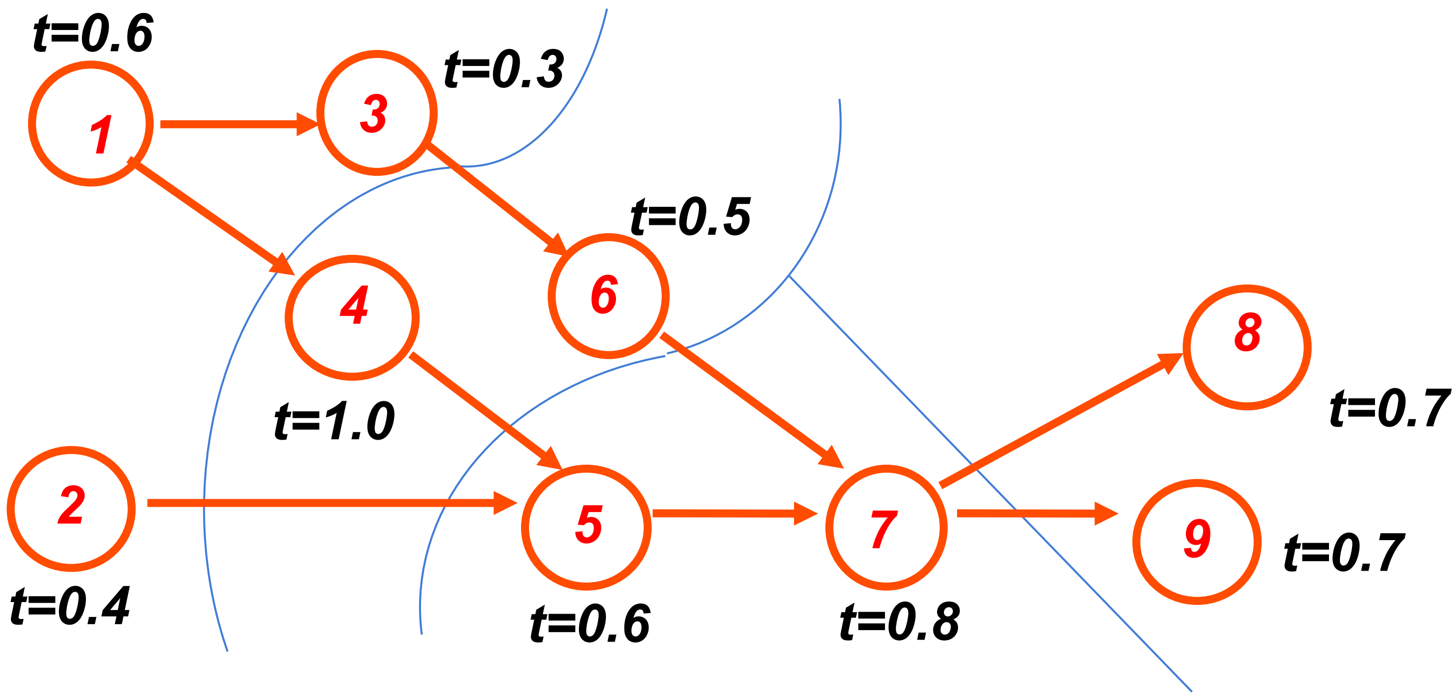
Evaluating Our Allocation I



$$\text{Balancing Loss} = 1 - \frac{\textit{Total processing time per item}}{\textit{Number of stations * Cycle time}}$$

$$= 1 - 5.6/(5 * 1.5) = 0.25 = 25\%$$

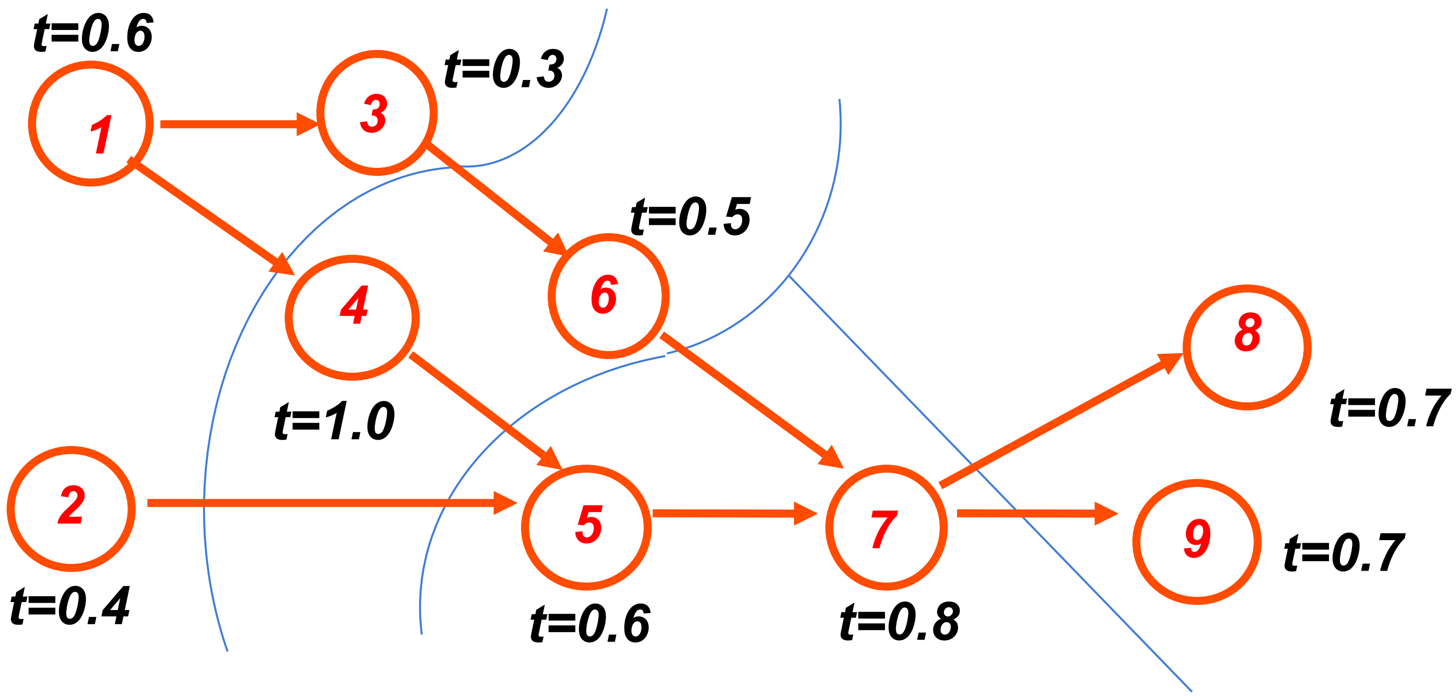
A Possible Allocation of Operations II



Is this OK? ➔

Station	Operations	Station Time
1	1,2,3	1.3 mins
2	4, 6	1.5 mins
3	5,7	1.4 mins
4	8,9	1.4 mins

Evaluating Our Allocation II



$$\text{Balancing Loss} = 1 - \frac{\textit{Total processing time per item}}{\textit{Number of stations * Cycle time}}$$

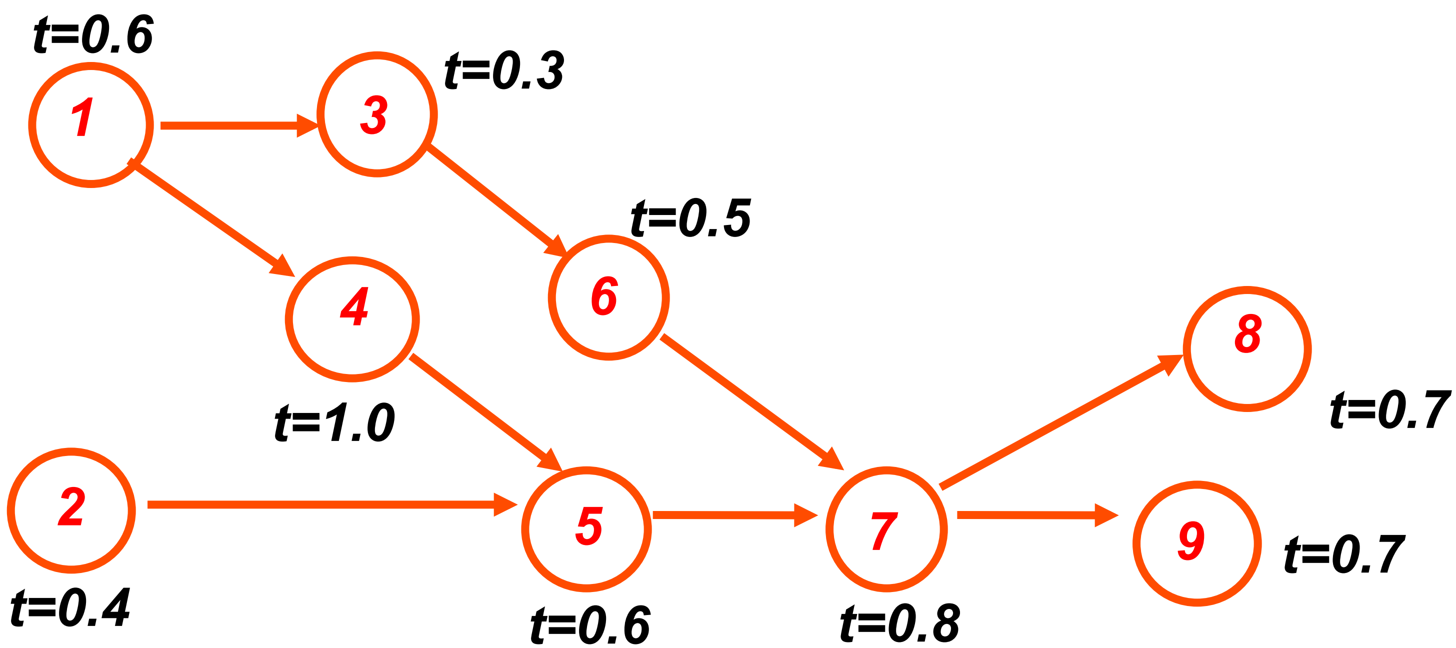
$$= 1 - 5.6/(4 * 1.5) = 0.067 = 6.7\%$$

Heuristic Methods for Assembly Line Balancing

- Small problems can be solved “by eye”; larger ones need heuristic methods
- In allocating operations to stations, allocate “most deserving” operation first, followed by second “most deserving”, etc.
- Examples of heuristics:

   1. Longest sequential chain of followers
   2. Total number of followers

Calculating Followers

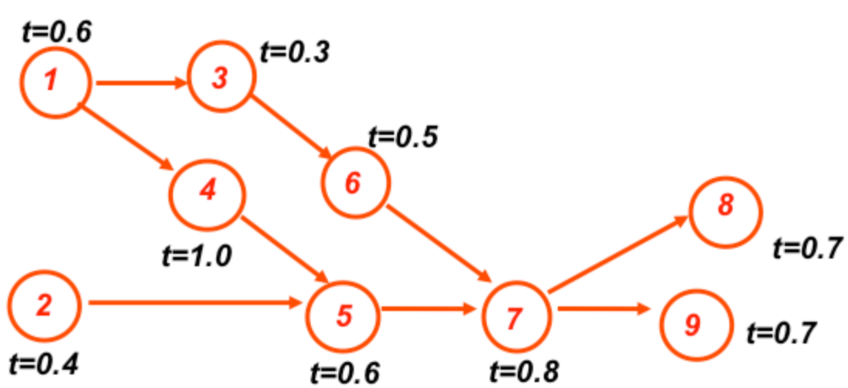


Operation	1	2	3	4	5	6	7	8	9
Followers (chain)	4	3	3	3	2	2	1	0	0
Followers (total)	7	4	4	4	3	3	2	0	0

Longest Sequential Chain of Followers

Operation	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
Followers (chain)	*4*	*3*	*3*	*3*	*2*	*2*	*1*	*0*	*0*
Followers (total)	*7*	*4*	*4*	*4*	*3*	*3*	*2*	*0*	*0*

- Operation: **1** (4 followers), **2,3,4** (3 followers), **5,6** (2 followers), **7** (1 follower), **8,9** (0 followers)
- Allocate operations with more followers first
- Break ties by taking operations with longest processing times first, thus **4** (1.0) before **2** (0.4) before **3** (0.3) …

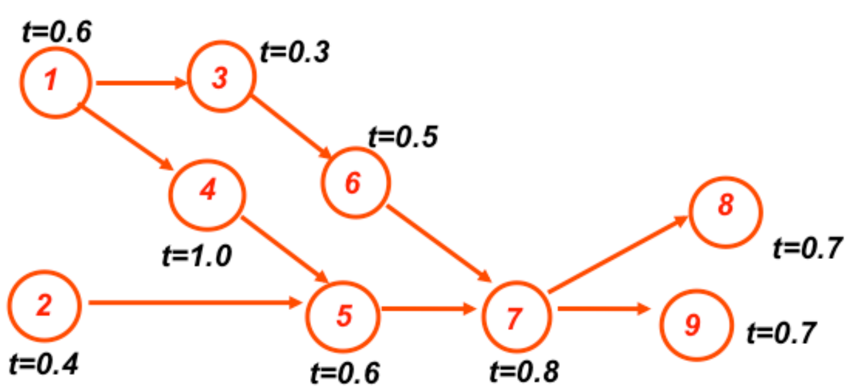


Station	Operations	Station Time
1	**1, 2, 3**	1.3 mins
2	**4, 6**	1.5 mins
3	**5,7**	1.4 mins
4	**8,9**	1.4 mins

Total Number of Followers

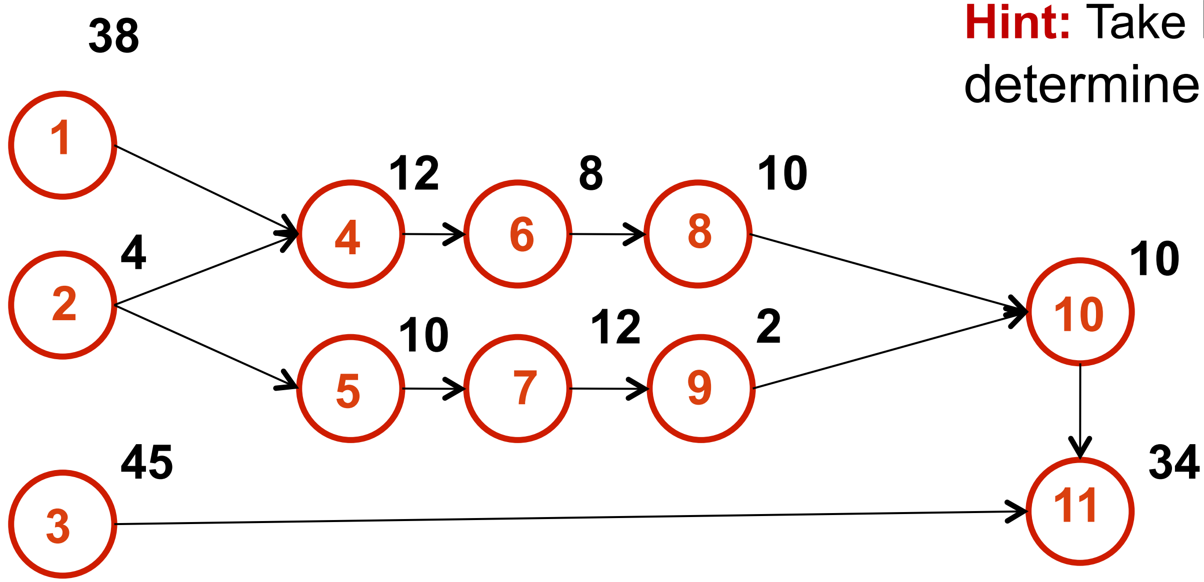
Operation	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
Followers (chain)	*4*	*3*	*3*	*3*	*2*	*2*	*1*	*0*	*0*
Followers (total)	*7*	*4*	*4*	*4*	*3*	*3*	*2*	*0*	*0*

- Operation: **1** (7 followers), **2,3,4** (4 followers), **5,6** (3 followers), **7** (2 followers), **8,9** (0 followers)
- Allocate operations with more followers first
- Break ties by taking operations with longest processing times first, thus **4** (1.0) before **2** (0.4) before **3** (0.3) …



Station	**Operations**	**Station Time**
1	**1, 2, 3**	1.3 mins
2	**4, 6**	1.5 mins
3	**5,7**	1.4 mins
4	**8,9**	1.4 mins

Have a Go!



Hint: Take bottleneck to determine cycle time

Layouts

Layout Types

- Where should specific pieces of equipment, people, and work centres be located?
- Important as:
  - ➢ Set up requires substantial investment of money and effort
  - ➢ Leads to long-term commitment
  - ➢ Has significant impact on cost and efficiency of operations
- Three basic layout types:
  1. Fixed-position
  2. Process (or Functional)
  3. Product
- Hybrid layouts combine features of these basic layouts

Fixed-Position Layout

- A layout that requires the product (e.g. because of its weight, size or delicacy) to remain at one location
- The equipment, machinery, plant and people who do the processing move as necessary
- Examples include:
  - Civil engineering projects
  - Shipbuilding
  - Restaurants with table service

Process Layout

- A layout that groups together machines, equipment or people with similar functions or goals
- Many products share usage of the same equipment.
- The order and length of processing at various steps will differ.
- Associated with job shops, batch processes and service shops

Product Layout

- A layout that dedicates equipment and workers to specific products on a linear route
- Can be dedicated to a single product or can be mixed-model lines
- When volumes for a particular product are high, resources can be specialized and designed for a specific product.
- Associated with continuous flow and service factories

Advantages/Disadvantages of Process vs. Product Layouts

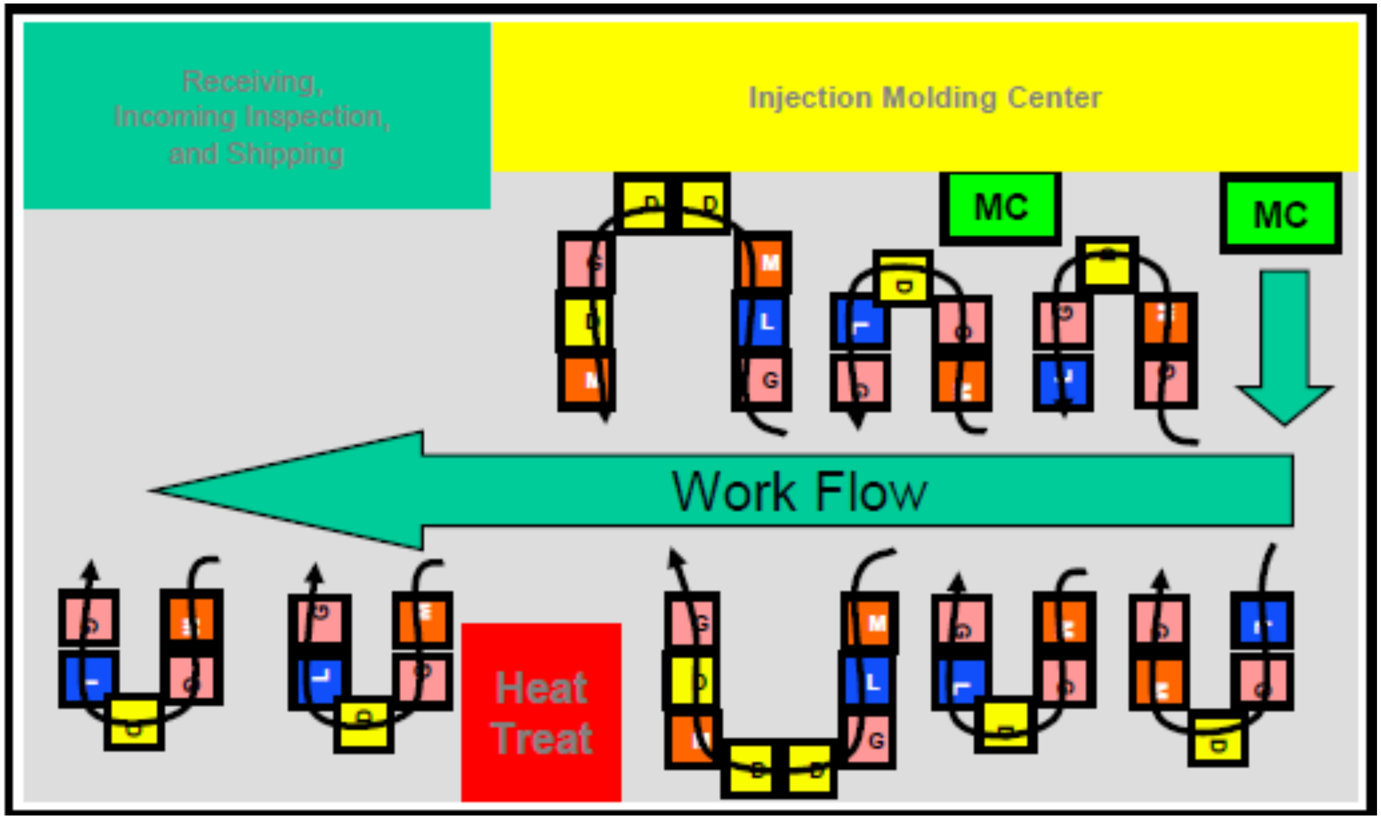
TABLE 4.1 Advantages/Disadvantages of Process vs. Product Layouts

Process Layouts	Product Layouts
Advantages	
• Resources are general purpose and less capital intensive	• Faster processing rates
• Flexibility is high—can adapt easily to changes in product mix or volume	• Lower inventories (and faster throughput)
• Capacity utilization is high because of the ability to perform multiple jobs/tasks on each piece of equipment	• Fewer changeovers, less material handling, more efficient
Disadvantages	
• Frequent changeovers result in lost production time	• Low flexibility
• High inventory levels are necessary to compensate for variable output rates	• Low utilization for lower-volume products
• Product throughput time can be very long	• High risk of layout redesign for products/ services with uncertain life cycles
• Production planning and control is challenging	

Hybrid Layouts

- Layouts that attempt to combine the advantages of process and product layouts by grouping disparate machines into **work centres** or **cells** to work on products that have similar shapes and processing equipment
- **Group technology**: an approach in which the product parts having similarities (shape, usage, and/or manufacturing process) are grouped together to achieve a higher level of integration between the product design and manufacturing

Cellular Manufacturing

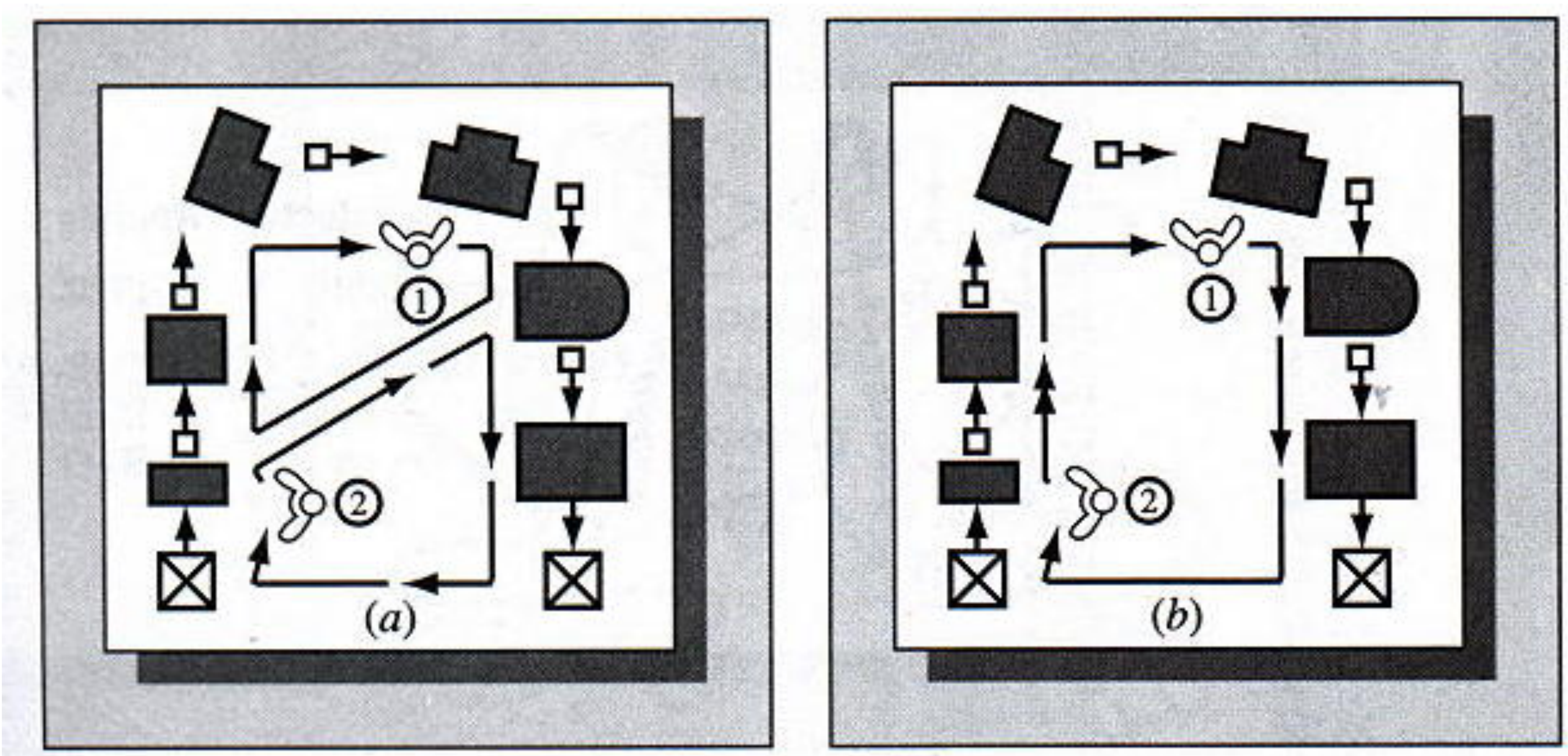


- Rather than route the materials required through the entire plant, materials flow to the head of each work cell, through each process in the cell, then to final assembly
- This eliminates most of the transportation and waiting in the traditional approach

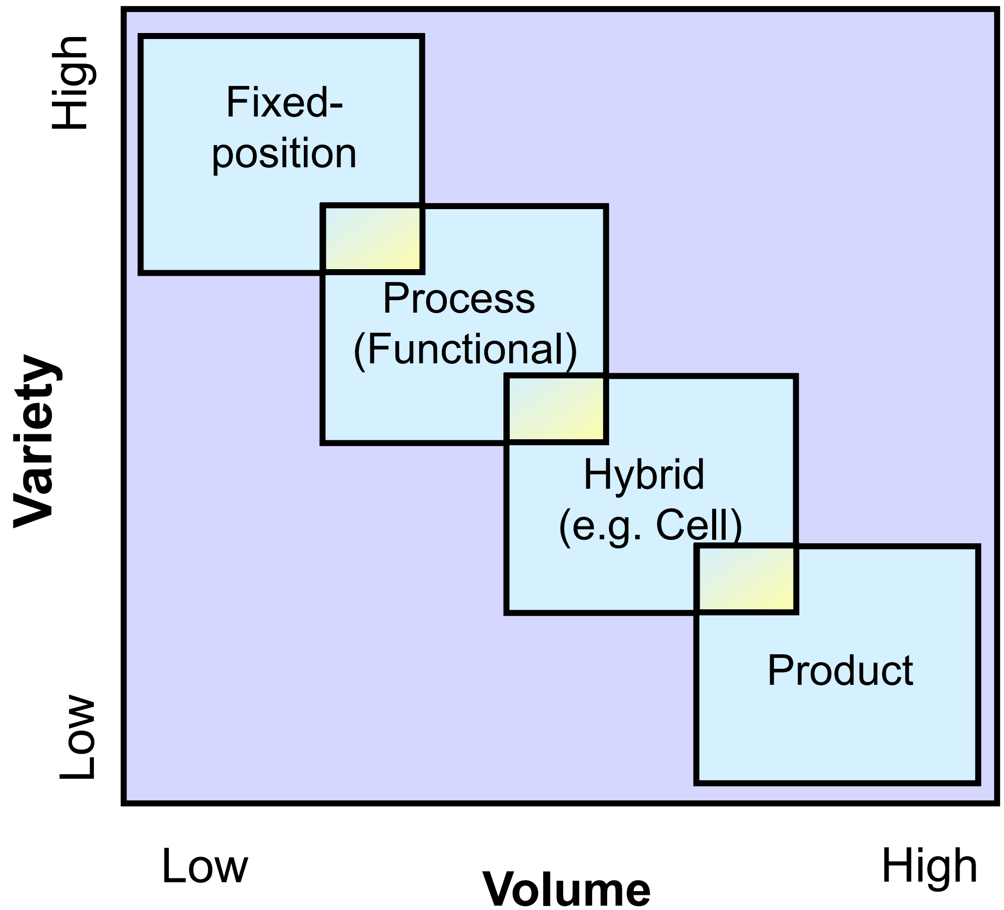
Key Elements of Cells

- Distances between workstations as small as possible
- Workers should stand and move between workstations
- Manual and machine work in parallel
- Hand to hand transfer – no conveyors!
- Preferable limit to workstations = 6
- Optimal number of operators between 5 and 7
- Ergonomics and presentation are important
- Operators perform multiple tasks; longer operator cycles reduce the waste of handling and passing; reduce boredom and repetitive strain

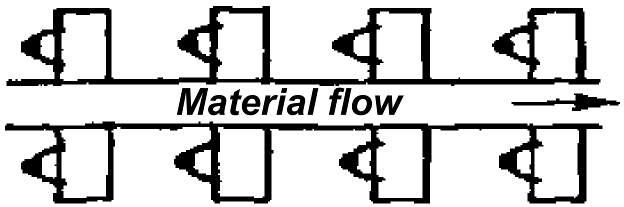
Multiskilling Gives Flexibility



Selecting a Layout Type

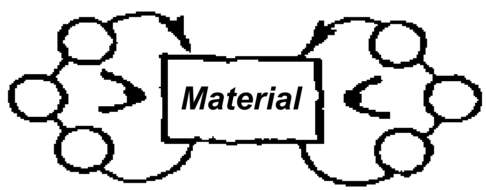


The Good, The Bad, The Ugly...



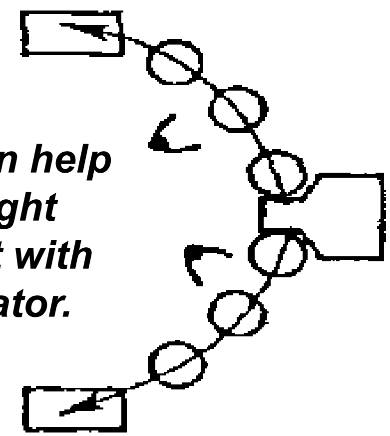
Bad: Operators caged. No chance to trade elements of work between them.

Better: Operators can trade elements of work. Can add and subtract operators. Trained workers will self-balance.



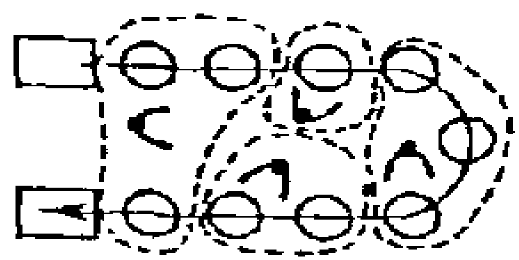
Bad: Operators birdcaged. No chance to increase input with a third operator.

Better: Operators can help each other. Might increase output with third operator.



Bad: Straight line difficult to balance.

Best: Effective operator access.



Why Layout is Important

- Anger over huge queues at Cambridge rail station

- The queue for tickets at 10.30am on a Saturday stretched a long way outside the main station building

Cambridge Railway Station

Takeaways from Today

- A number of rules (e.g., SPT, EDD and Moore's Algorithm) and heuristics (e.g., MDD) can be used for **single machine scheduling**
- Johnson's Rules is to minimise makespan when in a **two machine** flowshop
- **Push** systems schedule and issue orders centrally. **MRP** systems are computerised systems to support push scheduling
- In a **pull** system, processes are triggered by a replenishment signal, such as the **kanban** cards in a **JIT** scheduling
- **Assembly line balancing** involves grouping assembly tasks into workstations with as equal as possible workloads at each
- **Layout** is contingent on variety and volume

Operations Management

Dr Feryal Erhun
fe251@cam.ac.uk