ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΠΡΟΓΡΑΜ-ΜΑΤΙΣΜΟΣ ΠΑΡΑΓΩΓΗΣ

Βραχυχρόνιος Προγραμματισμός Παραγωγής

Γιώργος Λυμπερόπουλος

Πανεπιστήμιο Θεσσαλίας

Τμήμα Μηχανολόγων Μηχανικών

PRODUCTION PLANNING AND SCHEDULING

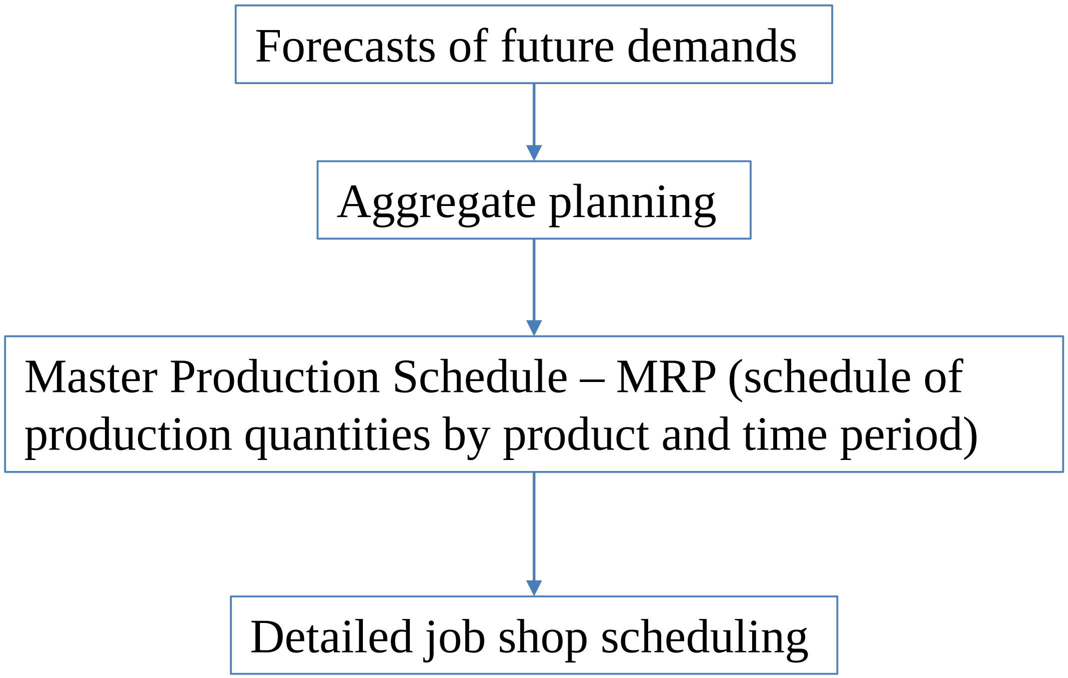
Short-Term Production Scheduling

George Liberopoulos

University of Thessaly

Department of Mechanical Engineering

Hierarchy of Production Decisions



Characteristics of job shop scheduling problems

- Job arrival pattern (static/dynamic problems)
- Number and variety of machines in the shop
- Number of workers in the shop (ability, speed of workers)
- Particular flow patterns (serial/mixed/rework)
- Evaluation of alternative rules

Objectives of job shop scheduling

1. Meet due dates
2. Minimize average flow time through the system
3. Minimize work-in-process (WIP) inventory
4. Minimize machine/worker idle time (maximize utilization)
5. Reduce setup times
6. Minimize production and worker costs
7. Provide for accurate job status information

Impossible to optimize all objectives simultaneously

- 1-2 $\Longrightarrow$ high customer service
- 3-6 $\Longrightarrow$ high level of plant efficiency (minimize cost)
- Conflicting objectives: e.g. 3 & 4

Job shop terminology

Flow shop

- *n* jobs must be processed through *m* machines in the same order

Job shop

- Jobs may not all require *m* operations
- Some jobs may require multiple operations on the same machine
- Each job may have different sequencing of operations

Parallel vs. sequential processing

Flow time of job *i*

- Time from the initiation of the first job until the completion of job *i*

Makespan

- Flow time of job completed last

Tardiness & Lateness

- Lateness = Completion time – due date
- Tardiness = (Completion time – due date)$^+$

Myopic sequencing rules

- First come – first served (FCFS)
- Shortest Processing Time (SPT)
- Earliest Due Date (EDD)
- Smallest Critical Ratio (CR)
  - CR = (Due date – Current time) / Processing time

Sequencing theory for a single machine, n jobs ($n!$ Permutations)

t_i = Processing time for job i

d_i = Due date for job i

W_i = Waiting time for job i

F_i = Flow time for job i $\quad F_i = W_i + t_i$

L_i = Lateness of job i $\; L_i = F_i - d_i$

T_i = Tardiness of job i $T_i = (L_i)^+ = \max(L_i, 0)$

E_i = Earliness of job i $E_i = (-L_i)^+ = \max(-L_i, 0)$

Performance Measures Examples

- Maximum Tardiness $T_{\max} = \max(T_1, T_2, \ldots, T_n)$
- Mean flow time $F' = \frac{1}{n}\sum_{i=1}^{n} F_i$

Sequencing theory for a single machine, n jobs (n! Permutations)

Some results

- Rule that minimizes **mean flow time** F': <u>**SPT**</u>
- The following measures are equivalent
    1. **Mean flow time** $F' = \frac{1}{n}\sum_{i=1}^{n} F_i$
    2. **Mean waiting time** $W' = \frac{1}{n}\sum_{i=1}^{n} W_i$
    3. **Mean lateness** $L' = \frac{1}{n}\sum_{i=1}^{n} L_i$
- Rule that minimizes **maximum lateness** $L_{\max}$: <u>**EDD**</u>

  ($L_{\max} = \max(L_1, L_2, \ldots, L_n)$)

Sequencing theory for a single machine, n jobs ($n!$ Permutations)

- Minimize **number of tardy jobs**: No simple rule

  Moore's algorithm

  1. Sequence jobs according to EDD to get initial solution.
  2. Find first tardy job, say $[i]$. If none exists, GOTO 4.
  3. Consider jobs $[1], [2], \ldots, [i]$. Reject job with largest processing time. GOTO 2.
  4. Optimal sequence: Current sequence + rejected jobs in any order

Sequencing theory for a single machine, n jobs ($n!$ Permutations)

Moore's algorithm: Example

i	1	2	3	4	5	6
d_i	15	6	9	23	20	30
t_i	10	3	4	8	10	6

i	2	3	1	5	4	6
d_i	6	9	15	20	23	30
t_i	3	4	$\boxed{10}$	10	8	6
Completion time	3	7	$\boxed{17}$	27	35	41

Jobs to consider

First tardy job

Reject job 1

Largest processing time

Sequencing theory for a single machine, n jobs (n! Permutations)

Moore's algorithm: Example (cont'd)

i	2	3	5	4	6
d_i	6	9	20	23	30
t_i	3	4	$\boxed{10}$	8	6
Completion time	3	7	17	$\boxed{25}$	31

Jobs to consider

Largest processing time

First tardy job

Reject job 5

Sequencing theory for a single machine, n jobs (n! Permutations)

Moore's algorithm: Example (cont'd)

i	2	3	4	6
d_i	6	9	23	30
t_i	3	4	8	6
Completion time	3	7	15	21

No tardy jobs!

Optimal sequence: 2, 3, 4, 6 | {1, 5}

Number of tardy jobs: 2

Sequencing theory for a single machine, n jobs (n! Permutations)

- **Precedence** constraints: Lawler's Algorithm

Minimize $\max_{1 \leq i \leq n} g_i(F_i)$

$g_i(F_i)$: non decreasing function

Examples:

$g_i(F_i) = F_i$ (flow time)

$g_i(F_i) = F_i - d_i = L_i$ (lateness)

$g_i(F_i) = (F_i - d_i)^+ = T_i$ (tardiness)

Sequencing theory for a single machine, n jobs (n! Permutations)

- **Precedence** constraints: Lawler's Algorithm (cont'd)
  1. Determine set of jobs, say V, not required to precede any other.
  2. Among the jobs in V, choose job k satisfying
     $g_k(\tau) = \min_{i \in V}(g_i(\tau))$, where $\tau = \sum_{i=1}^{n} t_i$
  3. Schedule job k last.
  4. If there are more jobs to schedule, GOTO 1.

Sequencing theory for a single machine, n jobs (n! Permutations)

- Lawler's Algorithm: Example

i	1	2	3	4	5	6
t_i	2	3	4	3	2	1
d_i	3	6	9	7	11	7

Precedence constraints:

1 → 2 → 3

4 → 5

4 → 6

Problem: Minimize $\max_{1 \leq i \leq n} (F_i - d_i)^+$ (minimize maximum tardiness)

Sequencing theory for a single machine, n jobs (n! Permutations)

- Lawler's Algorithm: Example (cont'd)

i	1	2	3	4	5	6
t_i	2	3	4	3	2	1
d_i	3	6	9	7	11	7

$1 \rightarrow 2 \rightarrow 3$

$4 \rightarrow 5$

$4 \rightarrow 6$

1. $V = \{3, 5, 6\}$
2. $\tau = \{t_1 + t_2 + t_3 + t_4 + t_5 + t_6) = 15$

$$\min \left\{ \begin{array}{ll} (15-9)^+ = 6, & i = 3 \\ (15-11)^+ = 4, & i = 5 \\ (15-7)^+ = 8, & i = 6 \end{array} \right\} = 4 \Rightarrow k = 5$$

3. Optimal sequence: _ _ _ _ _ 5

Sequencing theory for a single machine, n jobs (n! Permutations)

- Lawler's Algorithm: Example

i	1	2	3	4	5	6
t_i	2	3	4	3	2	1
d_i	3	6	9	7	11	7

1. $V = \{3, 6\}$
2. $\tau = \{t_1 + t_2 + t_3 + t_4 + t_6) = 13$

$$\min \begin{Bmatrix} (13-9)^+ = 4, & i = 3 \\ (13-7)^+ = 6, & i = 6 \end{Bmatrix} = 4 \Rightarrow k = 3$$

3. Optimal sequence: _ _ _ _ 3 5

1 → 2 → 3
4 → ~~5~~
4 → 6

Sequencing theory for a single machine, n jobs (n! Permutations)

- Lawler's Algorithm: Example

i	1	2	3	4	5	6
t_i	2	3	4	3	2	1
d_i	3	6	9	7	11	7

$1 \rightarrow 2 \rightarrow \cancel{3}$

$4 \rightarrow \cancel{5}$

$4 \rightarrow 6$

1. $V = \{2, 6\}$
2. $\tau = \{t_1 + t_2 + t_4 + t_6) = 9$

$$\min \begin{Bmatrix} (9-6)^+ = 3, & i = 2 \\ (9-7)^+ = 2, & i = 6 \end{Bmatrix} = 2 \Rightarrow k = 6$$

3. Optimal sequence: _ _ _ 6 3 5

Sequencing theory for a single machine, n jobs (n! Permutations)

- Lawler's Algorithm: Example

i	1	2	3	4	5	6
t_i	2	3	4	3	2	1
d_i	3	6	9	7	11	7

1. $V = \{2, 4\}$
2. $\tau = \{t_1 + t_2 + t_4) = 8$

$$\min\begin{Bmatrix}(8-6)^+ = 2, & i = 2 \\ (8-7)^+ = 1, & i = 4\end{Bmatrix} = 1 \Rightarrow k = 4$$

3. Optimal sequence: _ _ 4 6 3 5

$1 \rightarrow 2 \rightarrow \cancel{3}$

$4 \rightarrow \cancel{5}$

$4 \rightarrow \cancel{6}$

Sequencing theory for a single machine, n jobs (n! Permutations)

- Lawler's Algorithm: Example

i	1	2	3	4	5	6
t_i	2	3	4	3	2	1
d_i	3	6	9	7	11	7

1. $V = \{2\}$
2. $k = 2$
3. Optimal sequence: _ $\underline{2}$ $\underline{4\ 6}$ $\underline{3}$ $\underline{5}$

1 → 2 → ~~3~~
~~4~~ → ~~5~~
~~4~~ → ~~6~~

1. $V = \{1\}$
2. $k = 1$
3. Optimal sequence: $\underline{1}$ $\underline{2}$ $\underline{4\ 6}$ $\underline{3}$ $\underline{5}$

Sequencing theory for a single machine, n jobs ($n!$ Permutations)

Note:

- Rule that minimizes **maximum lateness** or **maximum tardiness**, if there are **no precedence** constraints: **EDD**

- Rule that minimizes **maximum flow time** (**makespan**): **Any sequence**

Sequencing theory for multiple machines

- n jobs through m machines in series (**flow shop**): $(n!)^m$ possible schedules
- Example: $n = 5,\ m = 5 \ \Rightarrow\ 24883 \times 10^{10}$ (~25 billion) possible schedules

- Special case: $m = 2$ machines

$$\rightarrow \boxed{A} \rightarrow \boxed{B} \rightarrow$$

- **Result**: If the objective is to **minimize makespan**, the **sequence** of jobs is the **same on both machines** (**permutation schedule**)
- $\Rightarrow (n!)$ possible schedules

Sequencing theory for multiple machines

- Special case: $m = 2$ machines

  A_i= Processing time of job i on machine A

  B_i= Processing time of job i on machine B

Johnson's Algorithm:

Job i precedes job j if $\min\left(A_i, B_j\right) < \min\left(A_j, B_i\right)$

1. List the values of A_i and B_i in two columns
2. Find smallest remaining value. If in **column A**, schedule **next**. If in **column B**, schedule **last**.
3. Cross off scheduled jobs.

Sequencing theory for multiple machines

Johnson's Algorithm: Example

Job i	A_i	B_i
~~1~~	~~5~~	~~$\boxed{2}$~~
~~2~~	~~$\boxed{1}$~~	~~6~~
3	9	7
~~4~~	~~$\boxed{3}$~~	~~8~~
~~5~~	~~10~~	~~$\boxed{4}$~~

- Optimal schedule: 2 4 3 5 1

Sequencing theory for multiple machines

Extension to 3 machines

$\rightarrow \boxed{A} \rightarrow \boxed{B} \rightarrow \boxed{C} \rightarrow$

- Considerably more complex
- If the objective is to **minimize makespan**, the **sequence** of jobs is the **same on both machines** (**permutation schedule**)
- 3-machine problem can be reduced to 2-machine problem if:

$$\min A_i \geq \max B_i \text{ or } \min C_i \geq \max B_i$$

Reduction: $\rightarrow \boxed{A'} \rightarrow \boxed{B'} \rightarrow$

$$A_i' = A_i + B_i \quad \text{and} \quad B_i' = B_i + C_i$$

If the condition is not satisfied, the reduction yields good but possibly suboptimal solutions

Sequencing theory for multiple machines

Extension to 3 machines (example)

Job i	A_i	B_i	C_i
1	4	5	8
2	9	6	10
3	8	2	6
4	6	3	7
5	5	4	11

- $\min A_i = 4 \ngeq 6 = \max B_i$ or $\min C_i = 6 \geq 6 = \max B_i$

Job i	A'_i	B'_i
~~1~~	~~$\boxed{9}$~~	~~13~~
2	15	16
~~3~~	~~10~~	~~$\boxed{8}$~~
~~4~~	~~$\boxed{9}$~~	~~10~~
~~5~~	~~$\boxed{9}$~~	~~15~~

Optimal schedule: 1 4 5 2 3