

DM87
SCHEDULING,
TIMETABLING AND ROUTING

Lecture 11

Flow Shop and Job Shop Models

Marco Chiarandini

Outline

1. Flow Shop

2. Job Shop

Outline

1. Flow Shop

2. Job Shop

Resume

Permutation Flow Shop:

- ▶ Directed graph representation and C_{\max} computation
- ▶ Johnson's rule for $F2 \parallel C_{\max}$
- ▶ Construction heuristics:
 - ▶ Slope heuristic
 - ▶ Campbell, Dudeck and Smith's heuristic
 - ▶ Nawasz, Ensore and Ham's heuristic

Outline

1. Flow Shop

2. Job Shop

$Jm \mid \mid C_{\max}$

[Job shop makespan]

Given:

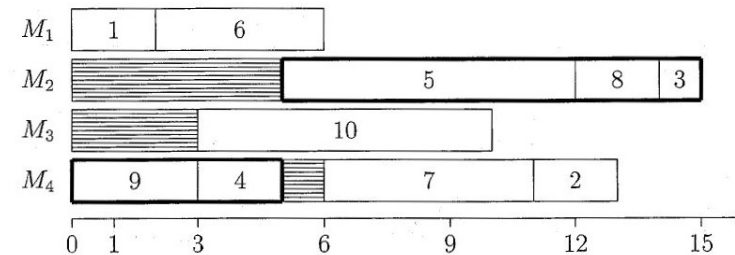
- ▶ $J = \{1, \dots, N\}$ set of jobs
- ▶ $M = \{1, \dots, m\}$ set of machines
- ▶ $J_j = \{O_{ij} \mid i = 1, \dots, n_j\}$ set of operations for each job
- ▶ $O_{1j} \rightarrow O_{2j} \rightarrow \dots \rightarrow O_{n_j,j}$ precedences (without loss of generality)
- ▶ p_{ij} processing times of operations O_{ij}
- ▶ $\mu_{ij} \in \{M_1, \dots, M_m\}$ with $\mu_{ij} \neq \mu_{i+1,j}$ eligibility for each operations (one machine per operation)
- ▶ without repetition and with unlimited buffers

Task:

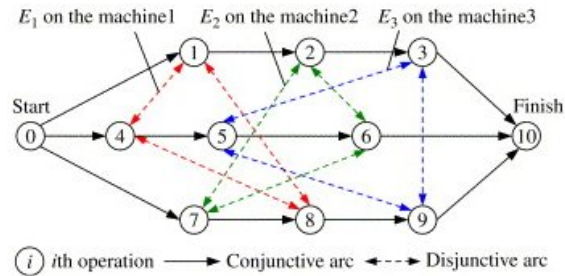
- ▶ Find a **schedule** $S = (S_{ij})$, indicating the starting times of O_{ij} , such that:
 - it is feasible, that is,
 - ▶ $S_{ij} + p_{ij} \leq S_{i+1,j}$ for all $O_{ij} \rightarrow O_{i+1,j}$
 - ▶ $S_{ij} + p_{ij} \leq S_{uv}$ or $S_{uv} + p_{uv} \leq S_{ij}$ for all operations with $\mu_{ij} = \mu_{uv}$.
- and has minimum makespan.

A schedule can be also represented by an m -tuple $\pi = (\pi^1, \pi^2, \dots, \pi^m)$ where π^i defines the processing order on machine i .

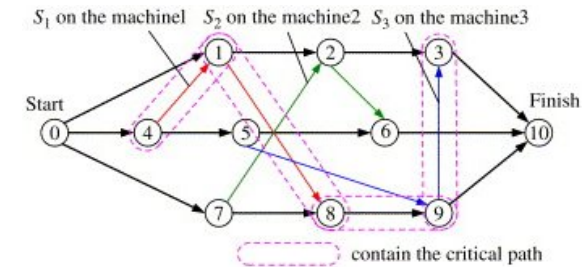
Then a semi-active schedule is found by computing the feasible earliest start time for each operation in π .



- ▶ Often simplified notation: $N = \{1, \dots, n\}$ denotes the set of operations
- ▶ **Disjunctive graph** representation: $G = (N, A, E)$
 - ▶ vertices N : operations with two dummy operations 0 and $n + 1$ denoting "start" and "finish".
 - ▶ directed arcs A , conjunctions
 - ▶ undirected arcs E , disjunctions
 - ▶ length of (i, j) in A is p_i



- ▶ A **complete selection** corresponds to choosing one direction for each arc of E .
- ▶ A complete selection that makes D acyclic corresponds to a feasible schedule and is called **consistent**.
- ▶ Complete, consistent selection \Leftrightarrow semi-active schedule (feasible earliest start schedule).
- ▶ Length of **longest path** $0-(n + 1)$ in D corresponds to the makespan



Longest path computation

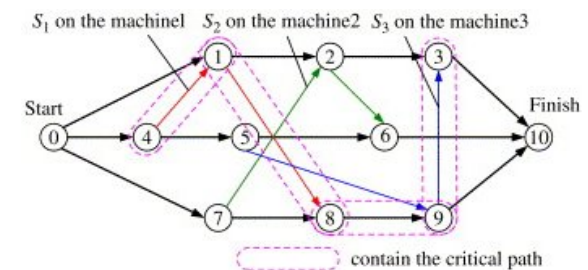
In an acyclic digraph:

- ▶ construct topological ordering ($i < j$ for all $i \rightarrow j \in A$)
- ▶ recursion:

$$r_0 = 0$$

$$r_l = \max_{\{j \mid j \rightarrow l \in A\}} \{r_j + p_j\} \quad \text{for } l = 1, \dots, n + 1$$

- ▶ A **block** is a maximal sequence of adjacent critical operations processed on the same machine.
- ▶ In the Fig. below: $B_1 = \{4, 1, 8\}$ and $B_2 = \{9, 3\}$



- ▶ Any operation, u , has two immediate predecessors and successors:
 - ▶ its job predecessor $JP(u)$ and successor $JS(u)$
 - ▶ its machine predecessor $MP(u)$ and successor $MS(u)$

Exact methods

► Disjunctive programming

$$\begin{array}{ll}
 \min & C_{\max} \\
 \text{s.t.} & x_{ij} + p_{ij} \leq C_{\max} \quad \forall O_{ij} \in N \\
 & x_{ij} + p_{ij} \leq x_{lj} \quad \forall (O_{ij}, O_{lj}) \in A \\
 & x_{ij} + p_{ij} \leq x_{ik} \vee x_{ij} + p_{ij} \leq x_{lk} \quad \forall (O_{ij}, O_{lk}) \in E \\
 & x_{ij} \leq 0 \quad \forall i = 1, \dots, m \quad j = 1, \dots, N
 \end{array}$$

► Constraint Programming

► Branch and Bound [Carlier and Pinson, 1983]

Typically unable to schedule optimally more than 10 jobs on 10 machines.
Best result is around 250 operations.

Shifting Bottleneck Heuristic

► A complete selection is made by the union of selections S_k for each clique E_k that corresponds to machines.

► **Idea:** use a priority rule for ordering the machines.
choose each time the bottleneck machine and schedule jobs on that machine.

► Measure bottleneck quality of a machine k by finding optimal schedule to a certain single machine problem.

► Critical machine, if at least one of its arcs is on the critical path.

– $M_0 \subset M$ set of machines already sequenced.

– $k \in M \setminus M_0$

– $P(k, M_0)$ is problem $1 \mid r_j \mid L_{\max}$ obtained by:

► the selections in M_0

► removing any disjunctive arc in $p \in M \setminus M_0$

– $v(k, M_0)$ is the optimum of $P(k, M_0)$

– bottleneck $m = \arg \max_{k \in M \setminus M_0} \{v(k, M_0)\}$

– $M_0 = \emptyset$

Step 1: Identify bottleneck m among $k \in M \setminus M_0$ and sequence it optimally. Set $M_0 \leftarrow M_0 \cup \{m\}$

Step 2: Reoptimize the sequence of each critical machine $k \in M_0$ in turn: set $M'_0 = M_0 - \{k\}$ and solve $P(k, M'_0)$.
Stop if $M_0 = M$ otherwise Step 1.

– Local Reoptimization Procedure

Construction of $P(k, M_0)$

$1 \mid r_j \mid L_{\max}$:

► $r_j = L(0, j)$

► $d_j = L(0, n) - L(j, n) + p_j$

$L(i, j)$ length of longest path in G : Computable in $O(n)$

An acyclic complete directed graph is the transitive closure of its unique directed Hamilton path.

Hence, only predecessors and successor are to be checked.

The graph is not constructed explicitly, but by maintaining a list of jobs per machines and a list machines per jobs.

$1 \mid r_j \mid L_{\max}$ can be solved optimally very efficiently.
Results reported up to 1000 jobs.

$1 | r_j | L_{\max}$

From Lecture 9

[Maximum lateness with release dates]

- ▶ Strongly NP-hard (reduction from 3-partition)
- ▶ might have optimal schedule which is not non-delay
- ▶ **Branch and bound** algorithm (valid also for $1 | r_j, prec | L_{\max}$)
 - ▶ **Branching:**
schedule from the beginning (level k , $n!/(k-1)!$ nodes)
elimination criterion: do not consider job j_k if:

$$r_j > \min_{l \in J} \{ \max(t, r_l) + p_l \} \quad J \text{ jobs to schedule, } t \text{ current time}$$

- ▶ **Lower bounding:** relaxation to preemptive case for which EDD is optimal

Tabu Search for Job Shop

Neighborhoods

Change the orientation of certain disjunctive arcs of the current complete selection

Issues:

1. Can it be decided easily if the new disjunctive graph $G(S')$ is acyclic?
2. Can the neighborhood selection S' improve the makespan?
3. Is the neighborhood connected?

Swap Neighborhood

[Novicki, Smutnicki]

Reverse one oriented disjunctive arc (i, j) on *some critical path*.

Theorem: All neighbors are consistent selections.

Note: If the neighborhood is empty then there are no disjunctive arcs, nothing can be improved and the schedule is already optimal.

Theorem: The swap neighborhood is connected.

Insertion Neighborhood

[Balas, Vazacopoulos, 1998]

For some nodes u, v in the critical path:

- ▶ move u right after v (forward insert)
- ▶ move v right before u (backward insert)

Theorem: If a critical path containing u and v also contain $JS(v)$ and

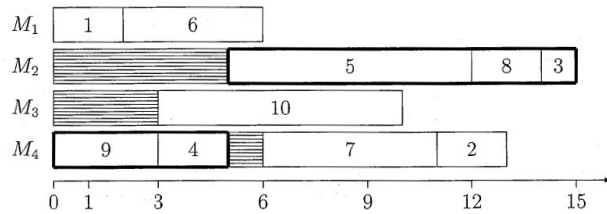
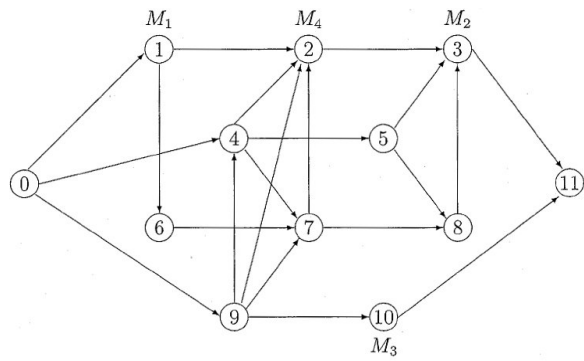
$$L(v, n) \geq L(JS(u), n)$$

then a forward insert of u after v yields an acyclic complete selection.

Theorem: If a critical path containing u and v also contain $JS(v)$ and

$$L(0, u) + p_u \geq L(0, JP(v)) + p_{JP(v)}$$

then a backward insert of v before v yields an acyclic complete selection.



Theorem: (Elimination criterion) If $C_{\max}(S') < C_{\max}(S)$ then at least one operation of a machine block B on the critical path has to be processed before the first or after the last operation of B.

- ▶ Swap neighborhood can be restricted to first and last operations in the block
- ▶ Insert neighborhood can be restricted to moves similar to those saw for the flow shop. [Grabowski, Wodecki]

Tabu Search requires a best improvement strategy hence the neighborhood must be search very fast.

Neighbor evaluation:

- ▶ exact recomputation of the makespan $O(n)$
- ▶ approximate evaluation (rather involved procedure but much faster and effective in practice)

The implementation of Tabu Search follows the one saw for flow shop.