Scheduling: job shops

Logistics

Giovanni Righini

Università degli Studi
di Milano

Job shops

In job shop models each job must undergo a given sequence of operations, but different jobs may require different sequences.

Most studies concentrate on makespan minimization, which is computationally difficult.

Two machines

$J2||C_{max}$: some jobs (subset $J_{1,2}$) must be processed on machine 1 first, some others (subset $J_{2,1}$) on machine 2 first.

The problem can be reduced to $F2||C_{max}$.

All jobs in $J_{1,2}$ are scheduled on machine 1 before all jobs in $J_{2,1}$ and vice versa on machine 2.

Then two instances of $F2||C_{max}$ arise, one for jobs in $J_{1,2}$ with machine 1 first and machine 2 second and the other for jobs in $J_{2,1}$ with machine 2 first and machine 1 second.

These are solved by $SPT(1) - LPT(2)$ sequences for each of the two sets.

This is one of the few job shop problems with a polynomial time optimization algorithm.

Disjunctive graph

$Jm||C_{max}$.

Consider a digraph $G = (N, A \cup B)$ with

- a node $(i, j) \in N$ for each operation (machine i processes job j);
- conjunctive arcs in A representing the routes of the jobs;
- disjunctive bi-directional arcs in B connecting pairs of jobs to be processed on the same machine;
- a cost p_{ij} for each arc outgoing from (i, j);
- A dummy source U and a dummy sink V connected to the first and last operations of each job with zero cost conjunctive arcs.

A feasible solution corresponds to the selection of an orientation for each disjunctive arc so that the resulting graph is acyclic.

The makespan is the cost of the longest $U - V$ path in the resulting acyclic graph.

MIP formulation

minimize C_{max}

$$
\begin{aligned}
\text{s.t. } & t_{hj} - t_{ij} \geq p_{ij} && \forall (i,j) \prec (k,j) \in A \\
& C_{max} - t_{ij} \geq p_{ij} && \forall (i,j) \in N \\
& t_{ij} - t_{il} \geq p_{il} \vee t_{il} - t_{ij} \geq p_{ij} && \forall (i,j),(i,l) \in N, j \neq l \\
& t_{ij} \geq 0 && \forall (i,j) \in N
\end{aligned}
$$

where t_{ij} indicates the starting time of (i,j).

Owing to the disjunctive constraints this model is called disjunctive programming formulation.

Active schedules

A feasible schedule is called active if there is no other feasible schedule such that at least one operation is completed earlier and no operation is completed later.

It can be proven that there exists an optimal schedule that is active.

Hence, a common branching technique in BnB algorithms for $Jm||C_{max}$ is designed to generate all possible active schedules.

Ω: set of schedulable jobs.
r_{ij}: earliest start time for each $(i,j) \in \Omega$.
Ω': subset of Ω.

Branching

1. Set Ω as the set of the first operations of the jobs.
2. Set $r_{ij} := 0$ for all $(i,j) \in \Omega$.
3. Compute
$$t(\Omega) = \min_{(i,j)\in\Omega} \{r_{ij} + p_{ij}\}$$
and let i^* be the machine where the minimum is achieved.
4. Set Ω' as the set of all $(i^*,j) \in \Omega$ such that $r_{i^*,j} < t(\Omega)$.
5. For each $(i^*,j) \in \Omega'$, extend the schedule with (i^*,j).
6. Delete (i^*,j) from Ω, insert its successor in Ω and repeat from step 3.

This branching method generated all active schedules.

Bounding

A trivial lower bound $\overline{C}_{max}$ is the length of the critical $U - V$ path in the current digraph.

A tighter lower bound can be computed for each machine i:

- assume all other machines can violate disjunctions (i.e. they can process different operations at the same time);
- compute the earliest starting time r_{ij} for all operations on machine i (the longest path from U to (i,j) in the current digraph);
- compute the length Δ_{ij} of the longest path from (i,j) to V in the current digraph;
- set a due date $d_{ij} = \overline{C}_{max} - \Delta_{ij} + p_{ij}$;
- solve $1|r_{ij}|L_{max}$ (NP-hard);
- temporarily orient the disjunctive arcs accordingly and recompute $\overline{C}_{max}$.

The shifting bottleneck heuristic

At each iteration a machine is selected and its disjunctive arcs are fixed.

M_0: set of machines already fixed;
M: set of all machines.

Delete all disjunctive arcs of machines in $M - M_0$.

Compute the critical path. Let $C_{max}(M_0)$ be its length.

The shifting bottleneck heuristic

For each (i,j) with $i \notin M_0$ set the release date and the due date, according to the critical path on the digraph.

For each machine $i \notin M_0$, solve a $1|r_j|L_{max}$ (NP-hard) and let $L_{max}(i)$ its optimal value.

The machine with maximum value of $L_{max}(i)$ (the *bottleneck*) is selected and its disjunctive arcs are fixed accordingly.

Before starting another iteration, machines in M_0 are tentatively rescheduled: for each machine $k \in M_0$, its disjunctive arcs are temporarily deleted and $1|r_j|L_{max}$ is solved. If a shorter makespan is found, then the solution is updated.

Delayed precedence constraints

When solving the $1|r_j|L_{max}$ subproblems, it is necessary to also include delayed precedence constraints, generated by teh disjunctive arcs fixed in the machines in M_0.

A certain delay is required between the end of an operation and the beginning of another.

Disregarding these constraints, the shifting bottleneck heuristic could produce infeasible solutions containing cycles.

Delayed precedence constraints make the $1|r_j|L_{max}$ subproblem instances harder.