Scheduling (parallel machines)

Logistics

Giovanni Righini

Università degli Studi di Milano

Scheduling problems with parallel machines

Scheduling problems with parallel machines involve two types of decisions

- allocation of jobs to machines;
- sequencing the jobs on each machine.

Model with and without pre-emptions may have different optimal solutions even if all jobs are released at $t = 0$.

Most models have optimal schedules that are non-delay, but problems with unrelated machines and no pre-emptions may have optimal solutions with delays.

We assume $p_1 \geq p_2 \geq \ldots \geq p_n$.

Makespan minimization (no preemptions)

$Pm||C_{max}$.

The problem is NP-hard even for $m = 2$.
Several heuristics have been developed.

Longest Processing Time (LPT) rule. At time $t = 0$, assign the m longest jobs to the m machines. Whenever a machine is freed, assign it the next longest job.

The worst-case approximation achieved by this heuristic is $\frac{4}{3} - \frac{1}{3m}$.

Makespan minimization (with precedences)

$Pm|prec|C_{max}$.

The version with no limits on the number of machines, $Pm|prec|C_{max}$, is easily solvable with the greedy algorithm known as Critical Path Method (CPM).

For generic $2 \leq m \leq n$ the problem is NP-hard.
Even the case with $p_j = 1 \ \forall j$ is not easy.

The special case when the precedences define a tree (in either direction) is polynomially solvable by the Critical Path rule: assign first the job at the head of the longest path of jobs in the precedence graph.

Makespan minimization (with precedences)

The CP rule applied to problems with arbitrary precedence constraints and $m = 2$ achieves an approximation factor of $4/3$.

Another heuristic priority rule is **Largest Number of Successors (LNS)**: it is based on the total number of jobs in the subtree rooted at each job in the precedence graph.

The LNS rule is optimal for $Pm|intree|C_{max}$ and for $Pm|outtree, p_j = 1|C_{max}$.

Variations of these heuristic rules prioritize jobs on the basis of total processing time of the successors of each job.

Makespan minimization (with incompatibilites)

$Pm||C_{max}$, where job j can be processed only on a machine subset M_j.

Consider the case $p_j = 1$ and *nested* subsets. For each job pair (j, k) exactly one of the following conditions holds:

1. $M_j = M_k$;
2. $M_j \subset M_k$;
3. $M_k \subset M_j$;
4. $M_j \cap M_k = \emptyset$.

An optimal solution is computed by the **Least Flexible Job (LFJ) rule**: every time a machine is freed, select the compatible job that can be processed on the smallest number of machines.

The LFJ rule is optimal for $P2|p_j = 1, M_j|C_{max}$, because subsets are always nested for $m = 2$.

For $m > 2$ it can provide sub-optimal solutions.

Makespan minimization (with preemptions)

$Pm|prmp|C_{max}$ is easy. It is an LP whose variables represent the amount of processing time of each job on each machine. Its objective function is min-max.

In $Qm|prmp|C_{max}$, assume to sort the n jobs so that $p1 \geq p_2 \geq \ldots \geq p_n$ and to sort the m uniform machines so that $v1 \geq v_2 \geq \ldots \geq v_m$.

$$C_{max} \geq \max\left\{\frac{p_1}{v_1}, \frac{p_1+p_2}{v_1+v_2}, \ldots, \frac{\sum_{j=1}^{m-1} p_j}{\sum_{j=1}^{m-1} v_j}, \frac{\sum_{j=1}^{n} p_j}{\sum_{j=1}^{m} v_j}\right\}$$

provides a valid lower bound.

If the largest term in the lower bound is given by $\frac{\sum_{j=1}^{k} p_j}{\sum_{j=1}^{k} v_j}$, then the $n-k$ shortest jobs are not processed on any of the fastest k machines.

Makespan minimization (with preemptions)

The **Longest Remaining Processing Time on the Fastest Machine (LRPT-FM) rule** is optimal for $Qm|prmp|C_{max}$.

However, in a continuous time context, it would imply an infinite number of preemptions.

This problem is fixed by processor sharing: a number m^* of machines process a number n^* of jobs, so that they simultaneously start and end.

The rule is optimal also in a discrete time context (preemptions are allowed only at integer values of time t).

The proof is based on the replacement of each machine i with speed v_i by v_i parallel machines with unit speed. At any time t any job is allowed to be processed on more than one unit-speed machine corresponding to the same machine i.

Total completion time (without preemptions)

$Pm||\sum_j C_j$. The problem is solved to optimality by the SPT rule.

By contrast, the WSTP rule does not extend to the parallel machines case.

It provides a heuristic with approximation guarantee $\frac{1}{2}(1+\sqrt{2})$.

$Pm|prec|\sum_j C_j$ is strongly NP-hard.

$Pm|outtree, p_j = 1|\sum_j C_j$ is solved to optimality by the CP rule.
The result does not hold for $Pm|intree, p_j = 1|\sum_j C_j$.

Proof

Let t_1 be the first point in time when more than m jobs can be scheduled.

Up to t_1 all job assignments comply with the CP rule.

Let t_2 be the last point in time when a rule R prescriibes a decision not complying with the CP rule.
Let $\overline{CP}$ the schedule from t_2 onwards.

Then, at t_2 there are m jobs, that are *not* heading the m longest paths in the precedence graph, assigned to the m machines.
In $\overline{CP}$ consider the longest precedence path p' headed by a job that is *not* assigned at t_2 and the shortest precedence path p'' headed by a job that *is* assigned at t_2.
Let $\overline{C}'$ and $\overline{C}''$ be the completion times of the last jobs of p' and p'' in $\overline{CP}$.
In $\overline{CP}$, $\overline{C}' \geq \overline{C}''$.

Proof

Owing to the CP rule, the job heading p' must start at time $t_2 + 1$ in $\overline{CP}$.

For the CP rule after t_2, all machines have to be busy at least up to $\overline{C}'' - 1$.

If $\overline{C}' \geq \overline{C}'' + 1$, then applying the CP rule at t_2 yields an improved solution, because the last job of p' is completed one time unit earlier, so that one more job is completed within $\overline{C}' - 1$ with respect to $\overline{CP}$.

Otherwise, $\overline{C}' = \overline{C}''$ and the two schedules are equivalent.

Unrelated machines

$Pm|p_j = 1, M_j|\sum C_j$ is easy when the subsets M_j are nested. It is solved to optimality by the LFJ rule.

The problem with subsets is a special case of $Rm||\sum C_j$, in which processing times are infinite (or "large enough") for some job-machine pairs.

It can be formulated as a weighted bipartite matching problem and solved at optimality in polynomial time.

If job j is processed on machine i and there are k jobs after it on the same machine, then it contributes kp_{ij} to the total completion time.

Let x_{ikj} be a binary variable indicating whether job j is scheduled as the k to last job on machine i.

Unrelated machines

$$
\begin{aligned}
\text{minimize } z = & \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} k p_{ij} x_{ikj} \\
\text{s.t. } & \sum_{i=1}^{m} \sum_{k=1}^{n} x_{ikj} = 1 && \forall j = 1, \dots, n \\
& \sum_{j=1}^{n} x_{ikj} \leq 1 && \forall i = 1, \dots, m \;\; \forall k = 1, \dots, n \\
& x_{ikj} \in \{0, 1\} && \forall i = 1, \dots, m \;\; \forall k = 1, \dots, n \;\; \forall j = 1, \dots, n
\end{aligned}
$$

The model corresponds to a bipartite matching problem and is polynomially solvable.

The optimal schedule may not be a non-delay schedule.

Total completion time (with preemptions)

The SPT rule is optimal also when preemptions are allowed.

$Qm|prmp|\sum C_j$ is solved by the Shortest Remaining Processing Time on the Fastest Machine (SRPT-FM) rule.

Every time the fastest machine completes a job, all jobs are prempted and moved to the next faster machine.

Lemma. There exists an optimal schedule in which $C_j \leq C_k$ when $p_j \leq p_k$ for all j and k.

Proof

Since $p_1 \geq p_2 \geq \ldots \geq p_n$, owing to the SRPT-FM rule, $C_n \leq C_{n-1} \leq \ldots \leq C_1$.

Owing to the job-machine assignments generated by the preemptions,

$$\begin{aligned} v_1 C_n &= p_n \\ v_2 C_n + v_1(C_{n-1} - C_n) &= p_{n-1} \\ \ldots & \quad \ldots \\ v_n C_n + v_{n-1}(C_{n-1} - C_n) + \ldots + v_1(C_1 - C_2) &= p_1 \end{aligned}$$

Proof

Adding these equations from row 1 to row k for all $k = 1, \ldots, n$ generates

$$\begin{aligned} v_1 C_n &= p_n \\ v_2 C_n + v_1 C_{n-1} &= p_n + p_{n-1} \\ \cdots &\quad \cdots \\ v_n C_n + v_{n-1} C_{n-1} + \ldots + v_1 C_1 &= p_n + p_{n-1} + \ldots + p_1 \end{aligned}$$

Proof

Assume a schedule S' is optimal. Then for the lemma

$$C'_n \leq C'_{n-1} \leq \ldots \leq C'_1.$$

The shortest job cannot be completed before p_n/v_1:

$$v_1 C'_n \geq p_n.$$

Since jobs n and $n-1$ are completed at C'_n and C'_{n-1} the amount of processing done on these two jobs cannot be larger than the amount of processing done on machine 1 and machine 2 in parallel up to C'_n and by machine 1 alone from C'_n up to C'_{n-1}, that is

$$(v_1 + v_2)C'_n + v_1(C'_{n-1} - C'_n).$$

Therefore

$$v_2 C'_n + v_1 C'_{n-1} \geq p_n + p_{n-1}.$$

Repeating the same argument we obtain

$$v_k C'_n + v_{k-1} C'_{n-1} + \ldots + v_1 C'_{n-k+1} \geq p_n + p_{n-1} + \ldots + p_{n-k+1}.$$

Proof

Now we can relate the equations coming from the SRPT-FM rule and the inequalities coming from the optimality assumption:

$$\begin{aligned} v_1 C'_n &\geq v_1 C_n \\ v_2 C'_n + v_1 C'_{n-1} &\geq v_2 C_n + v_1 C_{n-1} \\ \dots &\quad \dots \\ v_n C'_n + v_{n-1} C'_{n-1} + \ldots + v_1 C'_1 &\geq v_n C_n + v_{n-1} C_{n-1} + \ldots + v_1 C_1 \end{aligned}$$

If a vector of multipliers $\alpha \geq 0$ exists such that we can aggregate these rows to produce the inequality

$$\sum_j C'_j \geq \sum_j C_j$$

then we have proven that the SRPT-FM rule is optimal.

Proof

The multipliers are the solution of the system

$$\begin{aligned} v_1\alpha_1 + v_2\alpha_2 + v_3\alpha_3 + \ldots + v_n\alpha_n &= 1 \\ v_1\alpha_2 + v_2\alpha_3 + \ldots + v_{n-1}\alpha_n &= 1 \\ &\ldots \\ v_1\alpha_{n-1} + v_2\alpha_n &= 1 \\ v_1\alpha_n &= 1 \end{aligned}$$

Since $v_1 \geq v_2 \geq ldots \geq v_n$ such a solution $\alpha \geq 0$ exists.

Due dates related objectives

$Pm||L_{max}$ is not as easy as $1||L_{max}$.

Take all due dates equal to 0. Then minimizing L_{max} is the same as minimizing C_{max} and it is *NP*-hard.

One exception, that is easy to solve, is $Qm|prmp|L_{max}$.

It is solved by setting a bound $L_{max} = z$ and solving a feasibility problem.

By bisection, one can find the value z such that the problem is feasible for $L_{max} = z$ and infeasible for $L_{max} = z - 1$.

Due dates related objectives

In the feasibility problem, for each job j one must have $C_j \leq d_j + z$.

Set a deadline $D_j = d_j + z$ for each job.

Solving the feasibility problem is equivalent to solve $Qm|r_j, prmp|C_{max}$, where time has been reversed so that deadlines act as release dates.

Applying the LRPT-FM rule, one can find a feasible schedule if one exists.

Due dates related objectives

$Qm|r_j, prmp|L_{max}$ is also solved via parametric analysis, imposing $L_{max} = z$ and setting hard deadlines accordingly.

Reversing time in this problem does not help because release dates and deadlines are simply swapped.

It can be formulated as a network flow problem (polynomially solvable).