# Scheduling

Axel Parmentier

January 16, 2019

According to `Wikipedia`, a "schedule is a time management tool consisting of a list of times at which events are to occur, or an order in which they are to occur." Optimization of schedules has therefore countless applications, among which

- project management (see Section 1)
- production scheduling (Section 2 onwards)
- manpower scheduling (intersects production scheduling, but with specificities due to working rules)
- transport scheduling (see chapter on routing)
- computer scheduling (how to schedule threads, processes, and data flows)
- etc.

In this lecture, we will focus on the two first applications.

## 1 Minimum duration of a project

Let $J$ be a set of tasks $j$ representing a project. Each task has a processing time $p_j$. Furthermore, we have some precedence constraints: some tasks must be finished before others can be started. The objective is to find the minimum duration of the project.

1. Explain why this problem can be solved as a longest $s$-$t$ path problem on a digraph $D = (V, A)$ where $V = J \cup \{s, t\}$, where $s$ is a source vertex, $t$ is a sink vertex, and $A$ is a set of arcs which should be described, as well as the arc lengths.

   *Solution.* There is an arc $(j, j')$ is task $j$ must be performed before task $j'$, as well as arc $(s, j)$ and $(j, t)$ for all $t$. The length of arc $(s, j)$ is 0, the length of an arc $(j, \cdot)$ is $p_j$. □

| Tasks | Description | Processing time | Preceding tasks |
|---|---|---|---|
| $A$ | foundations | 6 | – |
| $B$ | wall constructions | 10 | A |
| $C$ | exterior plumbing | 4 | B |
| $D$ | interior plumbing | 5 | A |
| $E$ | electricity | 7 | A |
| $F$ | roof | 6 | B |
| $G$ | exterior painting | 16 | B,C,F |
| $H$ | panels | 8 | D,E |
| $I$ | floor | 4 | D,E |
| $J$ | interior | 11 | H,I |

Table 1: House construction project

2. Which algorithm should be used to solve that problem?

   *Solution.* The most efficient algorithm to compute a longest path is an acyclic digraph is dynamic programming. □

3. How can you identify the critical tasks, that is, the tasks for which there is no margin on their ending time: if they are just a little late, then the project will be late.

   *Solution.* The tasks on a critical path, i.e., a longest $s$-$t$ path. □

4. What is the minimum time on the house construction project on Table 1? Identify critical tasks.

   *Solution.* Length: 38. Critical tasks: A, B, F, G □

## 2 Production scheduling terminology

Production scheduling is the problem of affecting jobs to machines. Jobs and machines are generic terms: jobs can be any kind of tasks to be accomplished, and machine can be anything that is required for some task.

Jobs are indexed by $j$ and machines by $i$. There are $m$ machines and $n$ jobs. If explicitly mentioned, a job $j$ must be operated between its *release date* $r_j$ and its *due date* $d_j$. Job $j$ has sometimes a *weight*, denoted by $w_j$. The *processing time* of job $j$ on machine $i$ is denoted by $p_{ij}$: it is the time needed to operate $j$ on $i$. The processing time of $j$ may not depend on the machine, and we denote it in this case $p_j$.

On complicated problems, a job may require several *processing steps* on different machines. Pair $(i, j)$ then refers to the processing step or operation of job $j$ on machine $i$, and its duration is again denoted by $p_{ij}$.

In the literature, a scheduling problem is described by a triplet $\alpha|\beta|\gamma$ where $\alpha$ is the *machine environment*, $\beta$ contains additional constraints, and $\gamma$ indicated the *objective to minimize*.

The most frequent machine environment are

- *Single machine* (1)
- *Parallel machines*: identical machines in parallel ($Pm$), parallel machines with different speeds ($Qm$), i.e., $\frac{p_{ij}}{p_{i'j}}$ does not depend on $j$, or unrelated machines in parallel ($Rm$).
- *Open shop* ($Om$) each job must be processed on each of the $m$ machines in any order.
- *Flow shop* ($Fm$): there are $m$ machine in series, and each job has to be processes on machines $1, \ldots, m$ in this order. Generally, the order in which jobs are processed is identical on all machines. If one job can pass another, $\beta$ contains the entry *prmu*
- *Job shop* ($Jm$): each job must be processes on each machine, and the order in which it must be processed is fixed but job dependent.
- *Flexible flow shop* ($FFc$) and *flexible job shop* ($FJc$) are analogues of flow shop and job shop, the main difference being that, if there are $c$ types of machines, there are several machines of each type that are available and can work in parallel.

Parameter $\beta$ can contain

- *Release dates* ($r_j$)
- *Preemptions* (*prmp*). Jobs are generally assumed to be completed at once on a machine. Parameter *prmp* indicates that, on the contrary, a job can be stopped and restarted.
- *Precedence constraints* (*prec*): some jobs must be performed before others. Such constraints are easily modeled using an acyclic digraph.
- etc.

We denote by $C_{ij}$ the completion time of job $j$ on $i$; and by $C_j$ the *completion time* of $j$ on the last machine it visits. The *lateness* $L_j$ of a job is

$$L_j = C_j - D_j$$

and its *tardiness* is $T_j = \max(L_j, 0)$.

Frequent minimized objective $\gamma$ include

- Makespan ($C_{\max}$), defined as $C_{\max} = \max_j(C_j)$,
- Total weigthed completion time ($\sum_j w_j C_j$)
- Maximum lateness ($L_{\max}$), defined as $L_{\max} = \max_j(L_j)$
- Total weighted tardiness ($\sum_j w_j T_j$)

# 3 Single machine problems

## 3.1 Minimum weighted completion time

Consider the problem $1||\sum_j w_j C_j$: there is no release date, each job has processing time $p_j$ and weight $w_j$.

5. Show that processing the jobs in decreasing $\frac{w_j}{p_j}$ order gives an optimal solution.

   *Solution.* Consider a solution such that $j'$ is right after $j$, and $\frac{w_j}{p_j} < \frac{w'_j}{p'_j}$. Consider the solution obtained by exchanging $j$ and $j'$. Then the difference of the cost of the new solution minus the one of the solution is between the two solutions $w_j p_{j'} - w_{j'} p_j < 0$, and the solution is not optimal. Hence an optimal solution is obtained by processing the jobs in decreasing $\frac{w_j}{p_j}$ order. □

## 3.2 Precedence constraints – dynamic programming

Remark: there was an error on this question, which has been corrected.

We consider the problem $1|prec|\max_j T_j$, where each job $j$ has a given due date $d_j$, tardiness $T_j$ is defined in Section 2 and precedence constraints indicated by an acyclic digraph $D = ([n], A)$. Let $h_j$ be the mapping $t \mapsto \max(t - d_j, 0)$, such that $T_j = h_j(C_j)$.

6. Show that there exists an optimal schedule such that, for each integer $k < n$, we have

$$h_{j_k}\left(\sum_{j' \in J^c} p_{j'}\right) = \min_{j \in J^c :\, \delta^+(j) \subseteq J} h_j\left(\sum_{j' \in J^c} p_{j'}\right). \tag{1}$$

   where $j_k$ denotes the $k$th jobs operated in the schedule, $J$ denotes the jobs after $k$ in the schedule, and $J^c$ its complementary: $J^c = [n] \backslash J$.

   *Solution.* Consider an optimal schedule $s = j_1, \ldots, j_n$. Let $k$ be the largest integer such that (1) is not satisfied in $s$. We are going to build

an optimal schedule $\overline{s}$ such that the largest integer $\overline{k}$ such that (1) is not satisfied in $\overline{s}$ is such that $\overline{k} < k$. Iterating this procedure at most $n$ times gives an optimal schedule satisfying the desired property. We now explain how to build schedule $\overline{s}$. We denote by $C_j$ and $T_j$ the completion time and tardiness for schedule $s$. Then schedule $\overline{s}$ defined by $j_1, \ldots, j_{\overline{k}-1}, j_{\overline{k}+1}, \ldots, j_k, j_{\overline{k}}, j_{k+1}, \ldots, j_n$ is a schedule that respects the precedence constraints. We denotes by $\overline{C_j}$ and $\overline{T_j}$ the completion time and tardiness of job $j$ under this new schedule.

- If $\ell < \overline{k}$ or $\ell > k$, we have $\overline{T}_{j_\ell} = T_{j_\ell}$
- If $\overline{k} < \ell \leq k$, we have $\overline{T}_{j_\ell} \leq T_{j_\ell}$ by monotonicity of $h_\ell$
- $\overline{T}_{j_{\overline{k}}} \leq T_{j_k}$ by hypothesis.

Hence $\max_j \overline{T}_j \leq \max_j T_j$, and $\overline{s}$ satisfies the desired property. □

7. Give an algorithm solving $1|prec|\max_j T_j$ to optimality.

*Solution.* Initialize $J = \emptyset$. For $k$ from $n$ to 1, do

- select $j_k$ in the argmin of $\min_{j \in J_c \colon \delta^+(j) \subseteq J} h_j\left(\sum_{j' \in J^c} p_{j'}\right)$,
- $J \leftarrow J \cup \{j_k\}$
- $k \leftarrow j - 1$

We can then prove that the algorithm returns the optimal result by iteration on the number of jobs. If there is one job, the result is trivial. Suppose now that there is $n+1$ jobs. Let $\tilde{s} = \tilde{j}_1, \ldots, \tilde{j}_{n+1}$ be the schedule produce by that algorithm. Let $s = j_1, \ldots, j_n$ be an arbitrary schedule, and $k$ be such that $j_k = \tilde{j}_{n+1}$. Using the same proof as in the previous question, we obtain $\overline{s} = j_1, \ldots, j_{k-1}, \ldots, j_{n+1}, j_k$ has a smaller maximum tardiness than $s$. Furthermore by, induction hypothesis, $\tilde{j}_1, \ldots, \tilde{j}_n$ is an optimal scheduling of $\{j_1, \ldots, \tilde{j}_n\}$, and hence $\tilde{s}$ has smaller maximum tardiness that $\overline{s}$. This gives the induction hypothesis and concludes the proof . □

### 3.3 Precedence constraints – mathematical programming

We consider now $1|prec|\sum_j w_j T_j$ with $p_j \in \mathbb{R}_+$, where we recall that $T_j$ is the tardiness.

8. Give a MILP modeling this problem (indication: "big $M$" constraints may be helpful).

*Solution.* Let $M = \sum_{j\in[n]} p_j$ We use continuous variables $T_j$ and $C_j$, and binary variables $x_{jk}$ indicating if task $j$ is operated before task $k$.

$$\min_{C_j} \sum_{j\in[n]} w_j T_j \tag{2a}$$

$$\text{s.c. } C_j \geq C_k + p_j - M x_{jk}, \quad \forall j < k \tag{2b}$$

$$C_k \geq C_j + p_k - M(1 - x_{jk}), \quad \forall j < k \tag{2c}$$

$$C_k \geq C_j + p_k \quad \text{for all } (j,k) \in A \tag{2d}$$

$$C_j - p_j \geq 0 \quad \text{for all } j \text{ in } [n] \tag{2e}$$

$$T_j \geq C_j - d_j \tag{2f}$$

$$t_j \geq 0 \tag{2g}$$

$$C_j \geq 0 \tag{2h}$$

$$x_{jk} \in \{0,1\} \tag{2i}$$

□

We now consider the simpler case where $p_j \in \mathbb{Z}_+$ for all $j$. Let $T = \sum_j p_j$ Consider the following MILP.

$$\min \sum_{j\in J} w_j \sum_{t\in[T]} \max(t - d_j, 0) x_{jt} \tag{3a}$$

$$\text{s.t. } \sum_{t=0}^{T} x_{jt} = 1 \quad \forall j \in J \tag{3b}$$

$$\sum_{t'=0}^{t+p_k} x_{kt'} \leq \sum_{t'=0}^{t} x_{jt'} \quad \forall (j,k) \in A, \forall t \in [T - p_k] \tag{3c}$$

$$\sum_{j\in J} \sum_{t'=t}^{t+p_j-1} x_{jt'} \leq 1 \quad \forall t \in [T] \tag{3d}$$

$$x_{jt} \in \{0,1\} \quad \forall j \in J, \forall t \in \{0,\dots,T\} \tag{3e}$$

9. Explain the meaning of the binary variable $x_{jt}$ and of the different constraints.

   *Solution.* Binary variable $x_{jt}$ indicates if $C_j = t$. □

10. An interval matrix is a matrix such that each line is of the form $(0,\dots,0,1,\dots,1,0,\dots,0)$. show that an interval matrix is totally unimodular.

*Solution.* Subtract to each column (except the first) the previous column. The matrix obtained has at most one 1 and one -1 in its non-zero terms. □

11. Show that the matrix defined by constraints (3d) is totally unimodular

    *Solution.* It is an interval matrix, and therefore totally unimodular. □

12. Explain why the subproblem of the Lagrangian Relaxation of constraints (3b) and (3c) can be solved in polynomial time.

    *Solution.* The matrix of the pricing subproblem is totally unimodular, hence an optimal solution of its linear relaxation gives an optimal solution of the subproblem. Such a solution can be computed in polynomial time using the ellipsoid or interior point algorithms. □

# 4 Easy multiple machines problems

## 4.1 Operating scheduled jobs with a minimum number of machines

Suppose that we have a set of $n$ jobs with fixed starting and ending times $d_j = r_j + p_j$ that have to be processed by identical machines.

13. Show that the minimum number of machines required to operate all these jobs (and the schedules of these machines) can be computed in polynomial time.

    *Solution.* We identify the set of jobs by $[n]$. Let $D$ be the digraph with vertex set $([n] \times \{0,1\}) \cup \{s,t\}$, where $s$ and $t$ are respectively a source and a sink. Let $A$ be the arc set obtained by adding, for all $j$ in $[n]$

    - an arc $a = \big(s,(j,0)\big)$ for all $j$ in $[n]$ with lower capacity $\ell_a = 0$, upper capacity $u_a = +\infty$, and cost $c_a = 0$,
    - an arc $a = \big((j,0),(j,1)\big)$ with $\ell_a = 1$, $u_a = 1$, and $c_a = 0$,
    - an arc $a = \big((j,1),t\big)$ with $\ell_a = 0$ $u_a = +\infty$, and $c_a = 0$,
    - arc $a = \big((j,1),(j',0)\big)$ with $\ell_a = 0$ $u_a = +\infty$, and $c_a = 0$ for all $j'$ such that $d_j \leq r_{j'}$,

    as well as an arc $a = (t,s)$ with $\ell_a = 0$, $u_a = \infty$, and $c_a = 1$. A minimum cost circulation with lower capacity $(l_a)$, upper capacity $u_a$, and

cost ($c_a$) gives an optimal solution – we recall that a circulation is a $\boldsymbol{b}$-flow with $\boldsymbol{b} = \mathbf{0}$. Indeed, as capacity are integers, such a circulation can be decomposed into cycles, and $\ell_a = 1$, $u_a = 1$ for $a = \big((j,0),(j,1)\big)$ of the form $s,(j_1,0),(j_1,1),\ldots,(j_k,0),(j_k,1),t$ ensures that this decomposition partitions the job set: given a job $j$, $(j,0)$ and $(j,1)$ is on a unique cycle. Furthermore, the definition of arcs $\big((j,1),(j',0)\big)$ ensure that there is a bijection between sequences of jobs that can be operated by one machine and cycles.

Such a circulation can be found in polynomial time using a minimum cost flow algorithm. □

## 4.2 Minimum makespan with preemption allowed

Consider the problem $Pm|prmp|C_{\max}$, where $m$ jobs are processed on unrelated machines with processing times $p_j$, each job has release date $r_j$.

14. Let $C$ be an upper bound on the value of an instance of $Pm|prmp|C_{\max}$. Give a simple algorithm to rebuild a solution with value $C$.

    *Solution.* Let $m_j$ and $r_j$ be the quotient and the rest of the division of $\sum_{j'=1}^{j-1} p_j$ by $C_{\max}$, and $m'_j$ and $r'_j$ the quotient of the division of $\sum_{j'=1}^{j} p_j$ by $C_{\max}$. If $m_j = m'_j$, we schedule $j$ on $m_j$ between $r_j$ and $r'_j$. Otherwise, we schedule $j$ on $j$ between $r_j$ and $C$, and on $m'_j$ between 0 and $r'_j$. The solution is a schedule (no two machines on the same job at the same time) as $C$ is larger than $p_j$. □

15. Prove that the following linear program

$$\min \ z \tag{4a}$$

$$\text{s.t.} \ \sum_{i=1}^{n} x_{ij} = p_j, \quad \forall j \in [n], \tag{4b}$$

$$\sum_{i=1}^{n} x_{ij} \leq z, \quad \forall j \in [n], \tag{4c}$$

$$\sum_{j=1}^{m} x_{ij} \leq z, \quad \forall i \in [m], \tag{4d}$$

$$x_{ij} \geq 0, z \geq 0 \quad \forall i \in [m], \forall j \in [n]. \tag{4e}$$

    gives the optimal value of $Pm|prmp|C_{\max}$, and explain how to reconstruct an optimal solution from this value.

    *Solution.* It is immediate the a solution of $Pm|prmp|C_{\max}$ gives a solution of the linear program. Indeed, it suffices to define $x_{ij}$ as the time spent by machine $i$ in job $j$.

Conversely, given the optimal value $z^*$ of the linear program, and using the same technique as in the previous question, we rebuild a solution of $Pm|prmp|C_{\max}$ with value $z$: constraint (4c) ensures that a job is not scheduled at the same time on two machines, and constraint (4d) that we do not use more machine than we have. □

# 5 Branch and Bound and heuristics for the job shop problem

To conclude, we introduce a powerful tool that enables to solve many scheduling problems, the *disjunctive graphs*. We illustrate it on the job shop problem $Jm||C_{\max}$.

This graph contains

- A vertex for each processing step of each job, as well as a source vertex $s$ and a sink vertex $t$
- An arc between successive steps of each job.
- An $\big((i,j),(i,j')\big)$ edge between processing steps of different jobs on using the same machine.

An orientation of the edges of the disjunctive graph is an orientation (turning each edge into an arc) such that the resulting digraph is acyclic.

16. Explain why there is bijection between acyclic orientations of the edges of the disjunctive graph and solutions of the scheduling problem.

    *Solution.* Orienting an arc enables to indicate if a processing step is performed before another on a given machine. □

17. Given an orientation, how do we efficiently compute the makespan $C_{\max}$ of the corresponding scheduling.

    *Solution.* Longest $s$-$t$ path. □

18. Explain how the disjunctive graph can be used to build a Branch and Bound algorithm.

    *Solution.* Branch by orienting the edges. A bound is obtained by searching the longest path in the digraph where no-oriented edges are removed. □

19. Propose a metaheuristic to solve the problem:

(a) How is a solution encoded

*Solution.* Orientation of the arcs □

(b) Propose several neighborhoods

*Solution.* Reverse orientation of an arc. Reverse orientation of the disjunctive arcs of a critical path. □

20. Based on the disjunctive graph, propose a MILP modeling the problem.

*Solution.* For each $i$, $j$, and $j'$, let $x_{ijj'}$ be a binary indicating if step $(i,j)$ is performed before $(i,j')$. Let $M = \sum_i \sum_j p_{ij}$ The following MILP solves the problem.

$$
\begin{aligned}
\min\ & C_{\max} && && \text{(5a)}\\
\text{s.t. } & C_{ij} \le C_{\max} && \forall i \in [m], j \in [n] && \text{(5b)}\\
& x_{ijj'} = 1 - x_{ij'j} && \forall i \in [m], \forall j \in [n], \forall j' \in [n] && \text{(5c)}\\
& C_{i'j} \ge C_{ij} + p_{i'j} && \forall\big((i,j),(i',j)\big) \in A && \text{(5d)}\\
& C_{ij'} \ge C_{ij} + p_{ij'} - M x_{ij'j} && \forall\big((i,j),(i,j')\big) \in E && \text{(5e)}\\
& x_{ijj'} \in \{0,1\} && \forall\big((i,j),(i,j')\big) \in E && \text{(5f)}\\
& C_{ij} \ge 0 && \forall i \in [m], j \in [n] && \text{(5g)}
\end{aligned}
$$

□