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

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

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.

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

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 α is the *machine environment*, β contains additional constraints, and γ 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, β 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 β 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 γ include

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

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

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

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).

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.

$$
\begin{aligned}
\min \quad & \sum_{j\in J} w_j \sum_{t\in[T]} \max(t-d_j,0)x_{jt} && && \text{(2a)}\\
\text{s.t.} \quad & \sum_{t=0}^{T} x_{jt} = 1 && \forall j \in J && \text{(2b)}\\
& \sum_{t'=0}^{t+p_k} x_{kt'} \leq \sum_{t'=0}^{t} x_{jt'} && \forall (j,k) \in A, \forall t \in [T-p_k] && \text{(2c)}\\
& \sum_{j\in J} \sum_{t'=t}^{t+p_j-1} x_{jt'} \leq 1 && \forall t \in [T] && \text{(2d)}\\
& x_{jt} \in \{0,1\} && \forall j \in J, \forall t \in \{0,\dots,T\} && \text{(2e)}
\end{aligned}
$$

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

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.

11. Show that the matrix defined by constraints (2d) is totally unimodular

12. Explain why the subproblem of the Lagrangian Relaxation of constraints (2b) and (2c) can be solved in polynomial time.

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.

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.

15. Prove that the following linear program

$$
\begin{aligned}
\min\ & z && && \text{(3a)}\\
\text{s.t.}\ & \sum_{i=1}^{n} x_{ij} = p_j, && \forall j \in [n], && \text{(3b)}\\
& \sum_{i=1}^{n} x_{ij} \le z, && \forall j \in [n], && \text{(3c)}\\
& \sum_{j=1}^{m} x_{ij} \le z, && \forall i \in [m], && \text{(3d)}\\
& x_{ij} \ge 0, z \ge 0 && \forall i \in [m], \forall j \in [n]. && \text{(3e)}
\end{aligned}
$$

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

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.

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

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

19. Propose a metaheuristic to solve the problem:

(a) How is a solution encoded

(b) Propose several neighborhoods

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