Ch.4: Machine Scheduling Problem

Suppose that m machines $M_i (i = 1, \ldots, m)$ have to process n jobs $j (j = 1, \ldots, n)$. A ***schedule*** is for each job an allocation of one or more time intervals to one or more machines. A schedule is ***feasible*** if at any time, there is at most one job on each machine; each job is run on at most one machine. A schedule is optimal if it minimizes (or maximizes) a given optimality criterion. A scheduling problem type can be specified using three- field classification $\alpha / \beta / \gamma$ composed of machine environment, the job characteristics, and the optimality criterion.

4.1 Job Data

Let n denote the number of jobs. The following data is specified for each job j ($j=1, 2, \ldots, n$):

p_{ij} A processing time of its ith operation, $i=1, 2, \ldots, m_j$, where m_j is the number of operations on job j .If $m_j=1$, we shall write p_j instead of p_{ij} .

r_j A release date on which job j become available for processing.

d_j A due date, the time by which job j ideally be completed.

$\tilde{d}_j$ A deadline, the time by which j must be completed.

w_j The weight of job j representing the importance of job j relative to another job.

f_j A non-decreasing real cost function measuring the cost $f_j(t)$ incurred if job j completed at time t.

In general $p_{ij}, d_j, r_j, \tilde{d}_j$ and w_j are given positive integer constants.

4.2 Machine environment

The first field $\alpha = \alpha_1 \alpha_2$ represents the machine environment. If $\alpha_1 \in \{\phi, P, Q, R\}$, each job j consists of a single operation which can be processed on any machine M_i. Let p_{ij} denote the time to process job j on M_i.

$\alpha_1 = \phi$: ***Single machine***, there is only one machine, $p_{ij} = p_j$ for all j.

$\alpha_1 = P$: ***Identical parallel machines***; there are multiple machines operate at the same speed, $p_{ij} = p_j$ ($i=1, 2, \ldots, m$).

$\alpha_1 = Q$: ***Uniform parallel machines***; there are multiple machines, each machine M_i has its own speed v_i, $p_{ij} = p_j / v_i$ for all M_i and jobs *j*.

$\alpha_1 = R$: ***Unrelated parallel machines***; there are multiple machines with different job-related speeds, that is the processing times are unrelated. If machine M_i runs job *j* with a job-dependent speed v_{ij}, $p_{ij} = p_j / v_{ij}$ for all M_i and jobs *j*.

In parallel machine environment, a job can be processed in any of the m machines.

If $\alpha_1 \in \{J, F, O\}$, each job *j* consists of a set of operations $\{O_{1j}, O_{2j}, \ldots, O_{m_j j}\}$.

$\alpha_1 = J$: ***Job-shop***; each job *j* consists of a chain of operations $\{O_{1j}, O_{2j}, \ldots, O_{m_j j}\}$, which must be processed in that order. Each operation O_{ij} must be processed on a designated machine for p_{ij} units of time. The order in which operations are processed is fixed by the ordering of the chain, but the order may be different for different jobs.

$\alpha_1 = F$: ***Flow-shop***; is a special case of job-shop, each job *j* consists of a chain of operations $\{O_{1j}, O_{2j}, \ldots, O_{mj}\}$, where O_{ij} is to be processed on machine M_i for p_{ij} units of time. The order of the operations is the same for every job.

$\alpha_1 = O$: ***Open-shop***; each job *j* composed of a chain of operations $\{O_{1j}, O_{2j}, \ldots, O_{mj}\}$, where O_{ij} is to be processed on M_i for p_{ij} units of time. The order in which operations are executed is arbitrary.

$\alpha_2 \in \{\phi\} \cup \aleph$, where $\aleph$ is the set of natural numbers.

$\alpha_2 \in \aleph$: *m*, the number of machines, is constant and equal to α_2.

$\alpha_2 = \phi$: *m* is variable.

4.3 Job Characteristics

The second field $\beta \in \{\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6\}$ indicates certain job characteristics which are defined as follows:

$\beta_1 \in \{pmtn, \phi\}$

$\beta_1 = pmtn$: Preemptions are allowed, the processing of any job can be interrupted at no cost and resumed at a later time on any machine or at the same time on a different machine.

$\beta_1 = \phi$: Preemptions are not allowed, once a job is started on a machine, the job occupies that machine until it is finished.

$\beta_2 \in \{ prec, tree, chain, \phi \}$

$\beta_2 = prec$: A general precedence relation $\prec$ exists between the jobs, that is, if $i \prec j$ (job i precedes job j), then job i must be completed before job j can be started.

$\beta_2 = tree$: A precedence tree describes the precedence relation between jobs , that is each vertex in the associated graph has outdegree or indegree of at most one.

$\beta_2 = chain$: Precedence constraints between jobs are of chain-type where each vertex in the associated graph has outdegree and indegree of at most one.

$\beta_2 = \phi$: There is no precedence relation for the jobs; jobs are independent.

$\beta_3 \in \{ r_j, \phi \}$

$\beta_3 = r_j$: Jobs have release dates.

$\beta_3 = \phi$: $r_j=0$, ($j=1,2,\ldots,n$); all jobs are released at the same time.

$\beta_4 \in \{ \tilde{d}_j, \phi \}$

$\beta_4 = \tilde{d}_j$: Jobs have deadlines.

$\beta_4 = \phi$: No deadlines are specified.

$\beta_5 \in \{ p_{ij} = 1, pl \le p_{ij} \le pu, \phi \}$

$\beta_5 = p_{ij} = 1$: Each operation has a unit processing time.

$\beta_5 = pl \le p_{ij} \le pu$: Processing times are bounded below by pl and above by pu .

$\beta_5 = \phi$: No bounds on processing times.

$\beta_6 \in \{ s_f, \phi \}$

$\beta_6 = s_f$: There are sequence independent family set-up times, jobs are subdivided into families and a set-up time is incurred whenever there is a switch from processing a job in a family to a job in another family

$\beta_6 = \phi$: There are no set-up times.

4.4 Optimality Criteria

The third field γ defines the optimality criterion or the objective, the value which is to be optimized (minimized). Given a schedule, the following can be computed for each job j:

C_j The completion time, the time at which the processing of job j is completed.

F_j The flow time, the time job j spends in the system, $F_j = C_j - r_j$.

L_j The lateness, $L_j = C_j - d_j$, the amount of time by which the completion time of job j exceed its due date. Lateness can be negative if job j finishes earlier than its due date.

T_j The tardiness, $T_j = max\{L_j, 0\}$.

E_j The earliness, $E_j = max\{-L_j, 0\}$.

U_j The unit penalty, a unit penalty of job j if it fails to meet its deadline. $U_j = 0$ if $C_j \le d_j$, $U_j = 1$ otherwise.

The cost f_j for each job j usually takes one of the variables described above or the product of the weight w_j with one of the variables. The optimality criterion can be any function of the costs $f_j, j = 1, \dots, n$. Common optimality criteria are usually in the form:

1. $f = f_{\max} = \max\{f_j | j = 1,\dots,n\}$.
2. $f = \sum f_j$.

The following objective functions have frequently been chosen to be minimized.

$f = \sum(w_j)C_j$: The total (weighted) completion time.

Introducing due dates d_j ($j=1,\dots,n$) we have the following objective functions:

$f = C_{max}$: The maximum completion time (makespan)

$f = L_{max} = \max_j\{L_j\}$: The maximum lateness.

$f = T_{max} = \max_j\{T_j\}$: The maximum tardiness.

$f = \sum T_j$: The total tardiness.

$f = \sum U_j$: The total number of late jobs.

We may also choose to minimize:

$f = \sum w_j T_j$: The total weighted tardiness.

$f = \sum w_j U_j$: The total weighted number of late jobs.

$f = \sum w_j E_j$: The total weighted earliness.

Example (4.1):

$1/r_j / \sum w_j C_j$ is the problem of minimizing the total weighted completion time on single machine subject to non-trivial release date.

$P3/pmtn, prec / L_{max}$ is the problem of minimizing maximum lateness on three identical parallel machines subject to general precedence constraint, allowing preemption.

Example (4.2):

Consider the following schedule:

j	1	2	3	4	5	6	7	8
P_j	7	3	2	9	5	1	2	6
d_j	5	13	20	5	30	21	29	25

Then to calculate the total completion time, maximum lateness, total earliness, total tardiness, and the total number of late jobs:

j	1	2	3	4	5	6	7	8
P_j	7	3	2	9	5	1	2	6
d_j	5	13	20	5	30	21	29	25
C_j	7	10	12	21	26	27	29	35
L_j	2	-3	-8	16	-4	6	0	10
E_j	0	3	8	0	4	0	0	0
T_j	2	0	0	16	0	6	0	10

Then: $\sum C_j = 7 + 10 + 12 + 21 + 26 + 27 + 29 + 35 = 167, L_{max} = 16, \sum E_j = 15, \sum T_j = 34, \sum U_j = 4.$

4.5 Single Machine Scheduling Problems

4.5.1 $1 / / \sum C_j$ Problem

This is the problem of sequencing n jobs on a single machine to minimize the total completion time. This problem is solved by the SPT (shortest processing time) rule. The jobs are sequenced in non-decreasing order of processing times P_j.

Example (4.3):

Solve the following $1//\sum C_j$ problem:

j	1	2	3	4	5	6	7	8
P_j	7	3	2	9	5	1	2	6

To minimize $\sum C_j$, we use the SPT rule as follows:

j	6	3	7	2	5	8	1	4
P_j	1	2	2	3	5	6	7	9
C_j	1	3	5	8	13	19	26	35

Then by SPT rule: $\sum C_j = 1 + 3 + 5 + 8 + 13 + 19 + 26 + 35 = 110$. That is the optimal schedule is s= (6,3,7,2,5,8,1,4) with $\sum C_j = 110$.

4.5.2 $1\ /\ /\ \sum w_j C_j$ Problem

This is the problem of sequencing n jobs on a single machine to minimize the weighted total completion time. This problem is solved by the SWPT (shortest weighted processing time) rule. The jobs are sequenced in non-decreasing order of processing times P_j/w_j.

Example (4.4):

Consider the following schedule:

j	1	2	3	4	5
P_j	6	10	12	18	4
w_j	2	4	3	3	4

To minimize $\sum w_j C_j$, we must first find P_j/w_j for each job j:

j	1	2	3	4	5
P_j	6	10	12	18	4
w_j	2	4	3	3	4
P_j/w_j	3	2.5	4	6	1

Then, use the SWPT rule as follows:

j	5	2	1	3	4
P_j/w_j	1	2.5	3	4	6
P_j	4	10	6	12	18
w_j	4	4	2	3	3
C_j	4	14	20	32	50
w_jC_j	16	56	40	96	150

Then by SWPT: $\sum w_jC_j = 358$. That is the optimal schedule is s= (5,2,1,3,4) with $\sum w_jC_j = 358$ ($\sum w_jC_j = 498$ for the original sequence).

4.5.3 $1\ /\ /\ L_{max}$ Problem

This is the problem of sequencing n jobs on a single machine to minimize the maximum lateness. This problem is solved by the EDD (earliest due date) rule. The jobs are sequenced in non-decreasing order of due dates d_j.

Example (4.5):

Consider the following schedule:

j	1	2	3	4
P_j	4	5	3	2
d_j	7	8	5	4

To minimize L_{max} we use the EDD rule:

j	4	3	1	2
P_j	2	3	4	5
d_j	4	5	7	8
C_j	2	5	9	14
L_j	-2	0	2	6

$\therefore L_{max} = 6$(for the original schedule $L_{max} = 10$). The optimal schedule is s = (4,3,1,2) with $L_{max} = 6$.

4.5.4 $1\ /\ /\ \sum U_j$ Problem

This is the problem of sequencing n jobs on a single machine to minimize the number of late jobs (minimize the total unit penalties). This problem is solved by Moore algorithm. Let E denote the set of early jobs and L denote the set of late jobs. The jobs of E are sequenced in EDD rule followed by the jobs of L.

Moore (and Hodgson) Algorithm

Step 1: Number the jobs in EDD order. Set $E = \phi , L = \phi , \ k = 0, t = 0$.

Step 2: Let $k = k + 1$. If $k > n$ go to step 4.

Step 3: Let $t = t + P_k$ and $E = E \cup \{k\}$. If $t \leq d_k$ go to step 2. If $t > d_k$, find $j \in E$ with P_j as large as possible and let $t = t - P_j , E = E - \{j\}, L = L \cup \{j\}$. Go to step 2.

Step 4: E is the set of early jobs and L is the set of late jobs.

Example (4.6):

Minimize $\sum U_j$ for the following schedule:

j	1	2	3	4	5	6	7	8
P_j	5	3	1	8	4	7	5	3
d_j	12	32	10	18	23	27	15	24

To minimize $\sum U_j$ we use Moore algorithm:

j	3	1	7	4	5	8	6	2
P_j	1	5	5	8	4	3	7	3
d_j	10	12	15	18	23	24	27	32
C_j	1	6	11	19				
C_j	1	6	11	*	15	18	25	28

$\therefore \sum U_j = 1, E = \{3,1,7,5,8,6,2\}, L = \{4\}$. The optimal schedule is: s= (3,1,7 ,5,8,6,2,4) (in the original schedule $\sum U_j = 3$) .

Example (4.7):

Minimize $\sum U_j$ for the following schedule:

j	1	2	3	4	5	6	7	8
P_j	4	2	7	6	4	7	5	5
d_j	12	27	10	15	30	22	8	28

Solution:

To minimize $\sum U_j$ we use Moore's algorithm:

j	7	3	1	4	6	2	8	5
P_j	5	7	4	6	7	2	5	4
d_j	8	10	12	15	22	27	28	30
C_j	5	12						
C_j	5	*	9	15	22	24	29	
C_j	5	*	9	15	*	17	22	26

Remark: 5^{th} job (Job 6) is selected although it is early since it has the greatest P_j among all jobs in E.

$\therefore \sum U_j = 2, E = \{7,1,4,2,8,5\}, L = \{3,6\}$.The optimal schedule is: s=(7,1,4,2,8,5,3,6). Also, s=(7,1,4,2,8,5,6,3) is an optimal schedule.

Example (4.8):

Minimize $\sum U_j$ for the following schedule:

j	1	2	3	4	5	6	7	8
P_j	4	3	1	5	2	3	1	3
d_j	7	6	4	7	9	6	4	5

Solution:

To minimize $\sum U_j$ we use Moore's algorithm:

j	3	7	8	2	6	1	4	5
P_j	1	1	3	3	3	4	5	2
d_j	4	4	5	6	6	7	7	9
C_j	1	2	5	8				
C_j	1	2	*	5	8			
C_j	1	2	*	*	5	9		
C_j	1	2	*	*	5	*	10	
C_j	1	2	*	*	5	*	*	7

$\therefore \sum U_j = 4, E = \{3,7,6,5\}, L = \{8,2,1,4\}$.The optimal schedule is: s=(3,7,6,5,8,2,1,4).