Previous lectures: Classification of scheduling models $(\alpha|\beta|\gamma)$

- Machines

- Jobs

- Objective functions C_{max}, ΣC_j, $\Sigma w_j C_j$

  L_{max}, ΣT_j, $\Sigma w_j T_j$, ΣU_j, $\Sigma w_j U_j$

This lecture:

Single machine problems

Single machine models

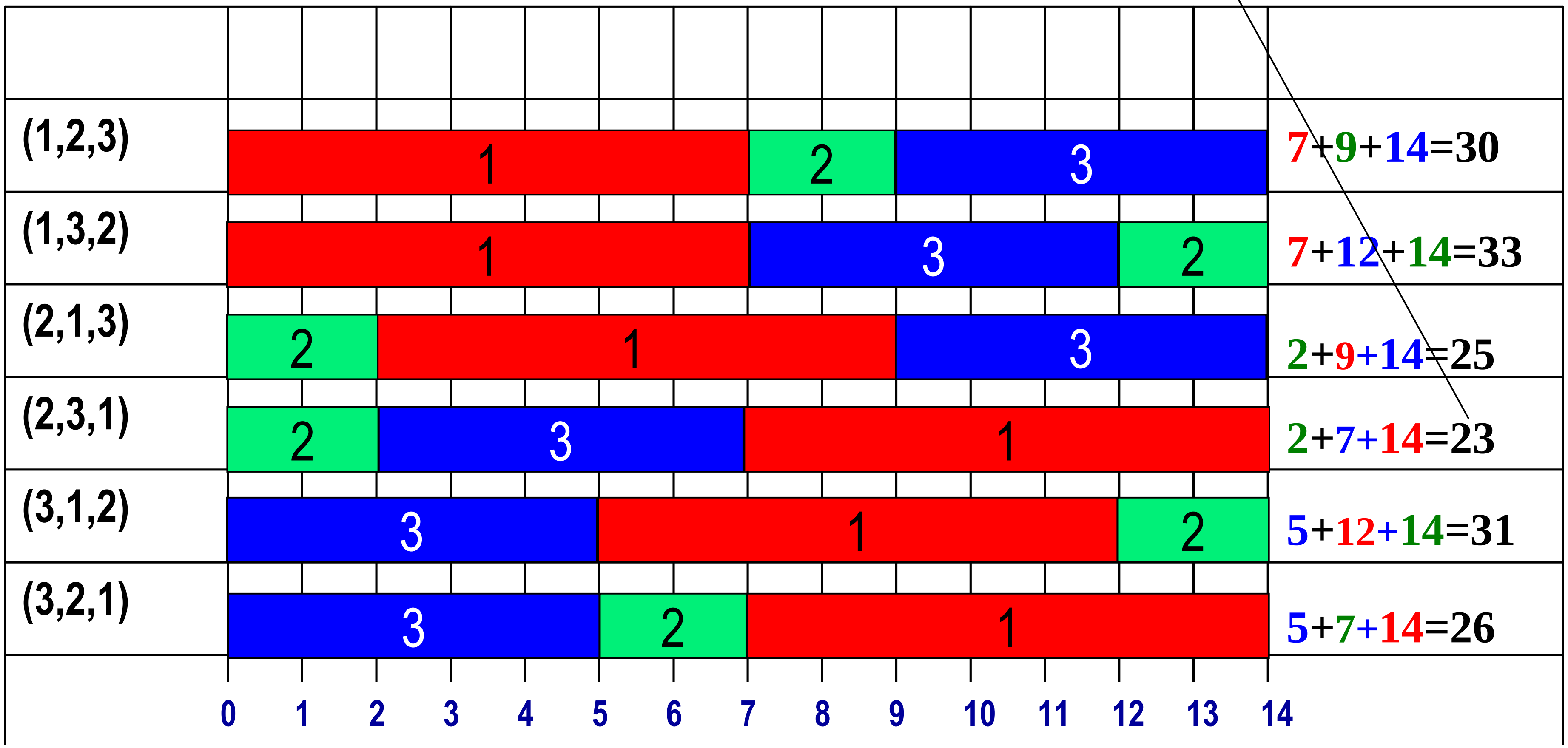
Why single machine models?

- arise in practice when there is one service point (production facility);
- special case of all other environments;
- multi-machine systems can often be decomposed into a number of single stage systems ;
- provide a basis for design of exact and approximation procedures for more complicated machine environments

$1||\sum C_j$

Job	p_j
1	7
2	2
3	5

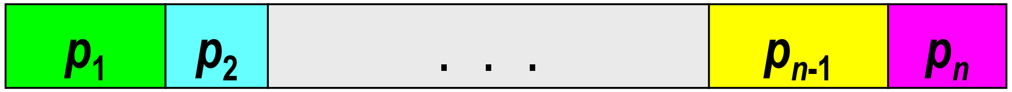
Best solution. How to find it?



$1 || \sum C_j$

Job	p_j
1	7
2	2
3	5

Take the jobs in any order **1, 2,…, *n***.



$$C_1 = p_1$$

$$C_2 = p_1 + p_2$$

$$\vdots$$

$$C_{n-1} = p_1 + p_2 + \dots + p_{n-1}$$

$$C_n = p_1 + p_2 + \dots + p_{n-1} + p_n$$

$$\sum C_j = np_1 + (n-1)p_2 + \dots + 2p_{n-1} + p_n$$

Thus, job 1 contributes np_1,

job 2 contributes $(n-1)p_2$,

and so on.

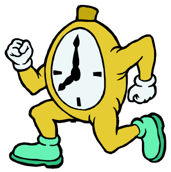
If we want to minimize $\sum C_j$,

we want p_1 to be the smallest,

p_2 the second smallest, etc.

$$1||\sum C_j$$

Job	p_j
1	7
2	2
3	5

- The problem is solved by ordering jobs in **SPT** order
- As any other sorting, this takes  **= $O(n \log n)$**

$1||\sum w_jC_j$

Consider $1||\,\Sigma w_jC_j$.

w_j - the importance of job j,

Σw_jC_j - indication of total holding, or inventory costs incurred by the schedule.

Consider first the special case of the problem above:

$1|\,p_j=1\,|\,\Sigma w_jC_j$.

$$1 \mid p_j = 1 \mid \sum w_j C_j$$

Job	p_j	w_j
1	1	7
2	1	2
3	1	5

Find an optimal job sequence ______________________________

and the optimal value of the objective function _______________

$1 | p_j = 1 | \sum w_j C_j$

Job	p_j	w_j
1	1	7
2	1	2
3	1	5

- In an optimal schedule the jobs have to be ordered in **decreasing** (non-increasing) order of their weights.

- As any other sorting, this takes  **= $O(n \log n)$**

$$1 \mid \cancel{p_j = 1} \mid \sum w_j C_j$$

$$1 \mid \mid \sum w_j C_j$$

- If $\boldsymbol{w_1} = \boldsymbol{w_2} = \ldots = \boldsymbol{w_n}$, then "<u>smaller</u>" jobs go first
- If $\boldsymbol{p_1} = \boldsymbol{p_2} = \ldots = \boldsymbol{p_n}$, then "<u>more expensive</u>" jobs go first

2. WSPT (Smith's rule)

Consider now $1||\ \Sigma w_j C_j$ with arbitrary p_j.

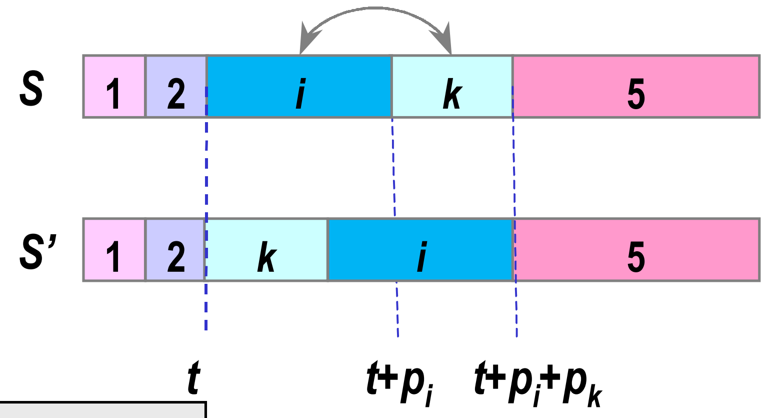
Weighted Shortest Processing Time *first* (WSPT) rule:

the jobs are sequenced in nondecreasing order of p_j / w_j.

W.E. Smith: Various optimizers for single-stage production.

Naval Research Logistics Quarterly 3, 59-66, 1956

2. WSPT (Smith's rule)



Theorem 1. For $1||\sum w_j C_j$ the WSPT rule is optimal.

Proof. **Adjacent pairwise interchange:**

Suppose a schedule ***S***, which is not WSPT, is optimal.

In this schedule there must be at least two adjacent jobs ***i*** and ***k*** such that $\dfrac{p_i}{w_i} > \dfrac{p_k}{w_k}$

We assume that ***i*** precedes ***k*** and ***i*** starts at time ***t.***

Swapping jobs ***i*** and ***k*** leads to a schedule ***S'*** such that

$$F(S) = \Sigma_{i \neq j,k} w_i C_i + w_j C_j + w_k C_k = \Sigma_{i \neq j,k} w_i C_i + w_j(t + p_j) + w_k(t + p_j + p_k)$$

$$F(S') = \Sigma_{i \neq j,k} w_i C_i + w_j C_j' + w_k C_k' = \Sigma_{ii \neq j,k} w_i C_i + w_k(t + p_k) + w_j(t + p_j + p_k)$$

$$F(S') - F(S) = w_j p_k - w_k p_j < 0$$

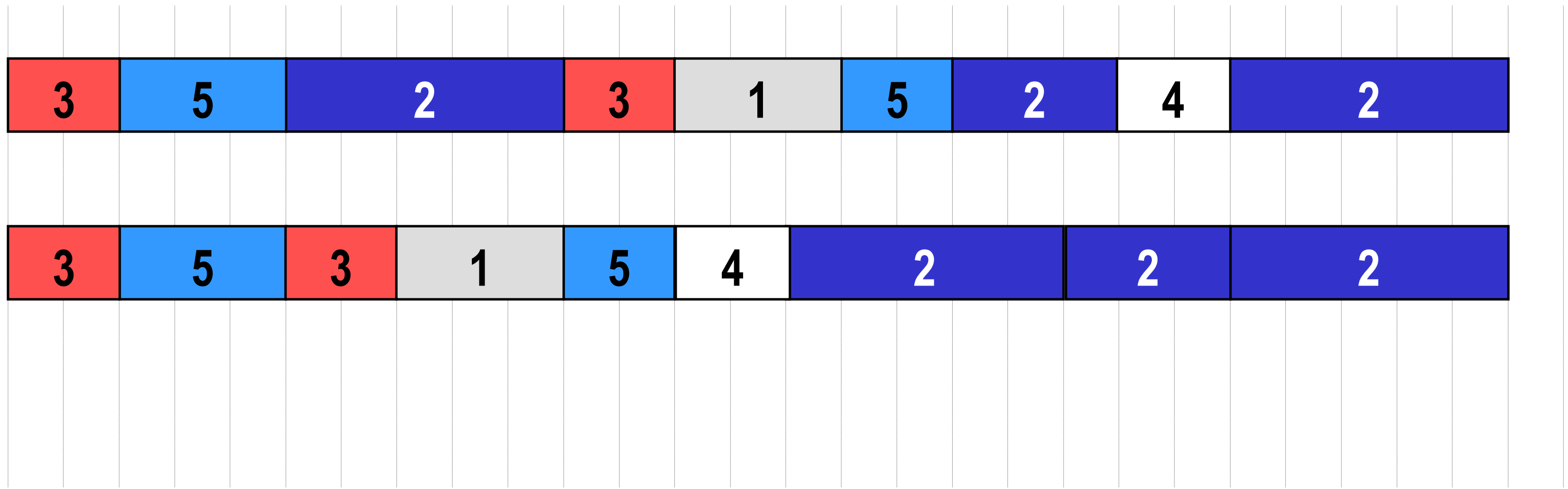
■

3. Preemption: $1|Pmtn|\sum C_j$

$p_1 = 3$; $p_2 = 13$; $p_3 = 4$; $p_4 = 2$; $p_5 = 5$

3. Preemption: $1|Pmtn|\Sigma C_j$

$p_1 = 3$; $p_2 = 13$; $p_3 = 4$; $p_4 = 2$; $p_5 = 5$



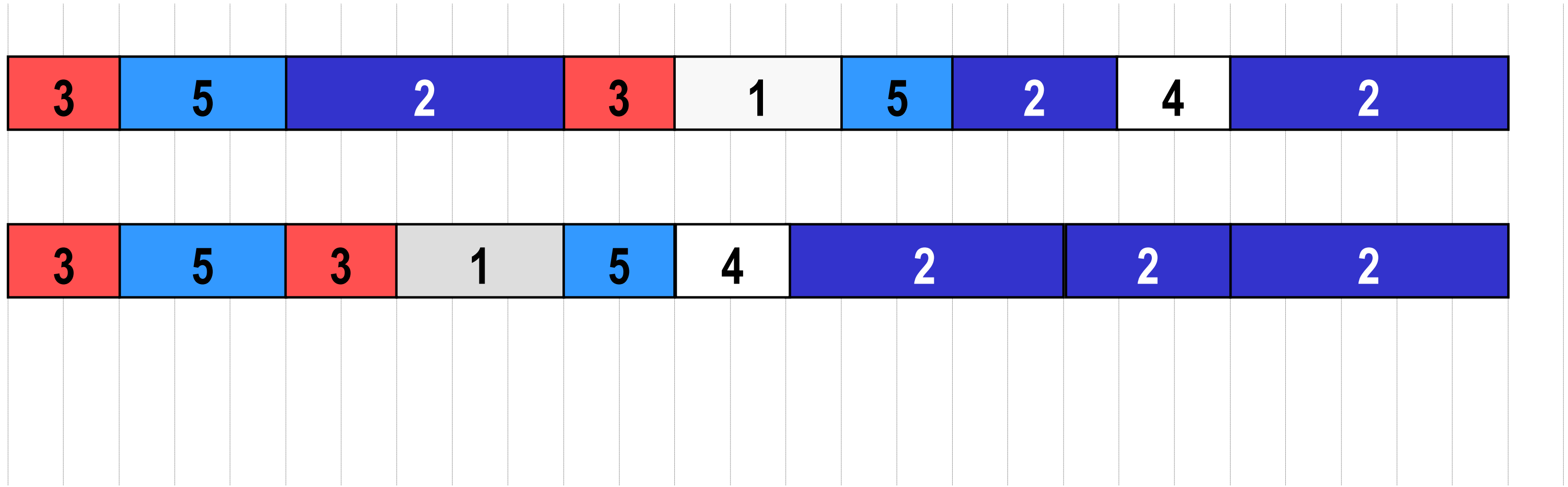
Suppose job j is processed with preemption.

Modify the schedule by moving all parts of job j just before its last part and processing job j contiguously.

Those jobs that are processed before the last part of job j are shifted to the left.

3. Preemption: $1|Pmtn|\,\Sigma C_j$

$p_1 = 3$; $p_2 = 13$; $p_3 = 4$; $p_4 = 2$; $p_5 = 5$



For problem $1|Pmtn|\boldsymbol{F(C_1,\ldots,C_n)}$ there exists an optimal schedule without preemption.

McNaugton: Scheduling with deadlines and loss functions.

Management Science 6, 1-12, 1959

4. SRPT-rule for problem $\mathbf{1|}\ \boldsymbol{r_j}$ **, Pmtn|** $\boldsymbol{\Sigma C_j}$

Consider $\mathbf{1|}\ \boldsymbol{r_j}$ **, Pmtn** | $\Sigma \boldsymbol{C_j}$.

Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.

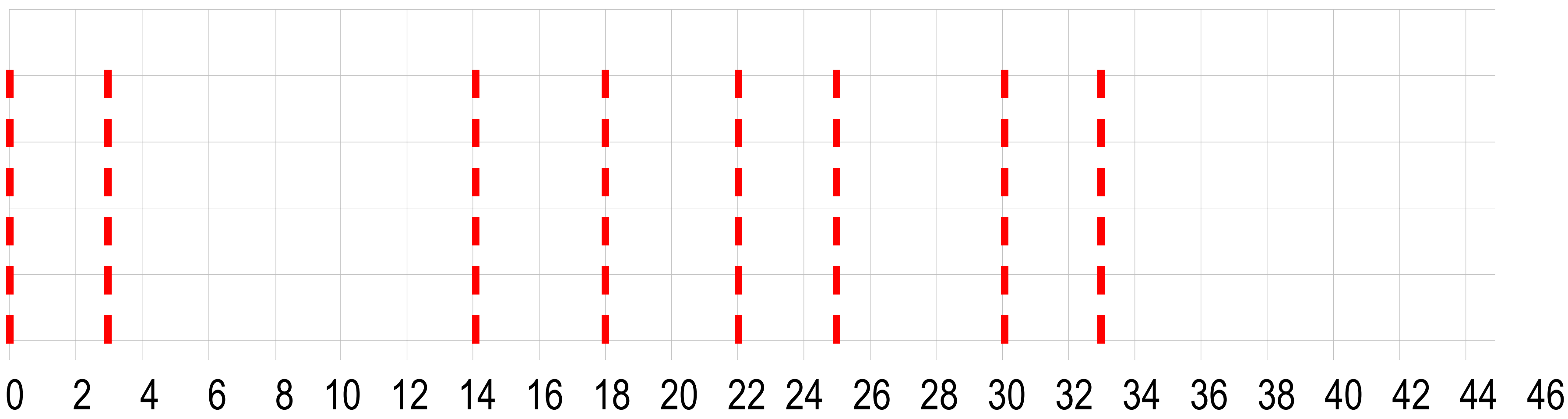
K.R. Baker: Introduction to Sequencing and Scheduling.

John Wiley & Sons, 1974

4. SRPT-rule for problem **1| r_j , Pmtn| ΣC_j**

	r_j	p_j
J_1	0	13
J_2	3	6
J_3	14	8
J_4	18	3
J_5	22	5
J_6	25	3
J_7	30	4
J_8	33	1

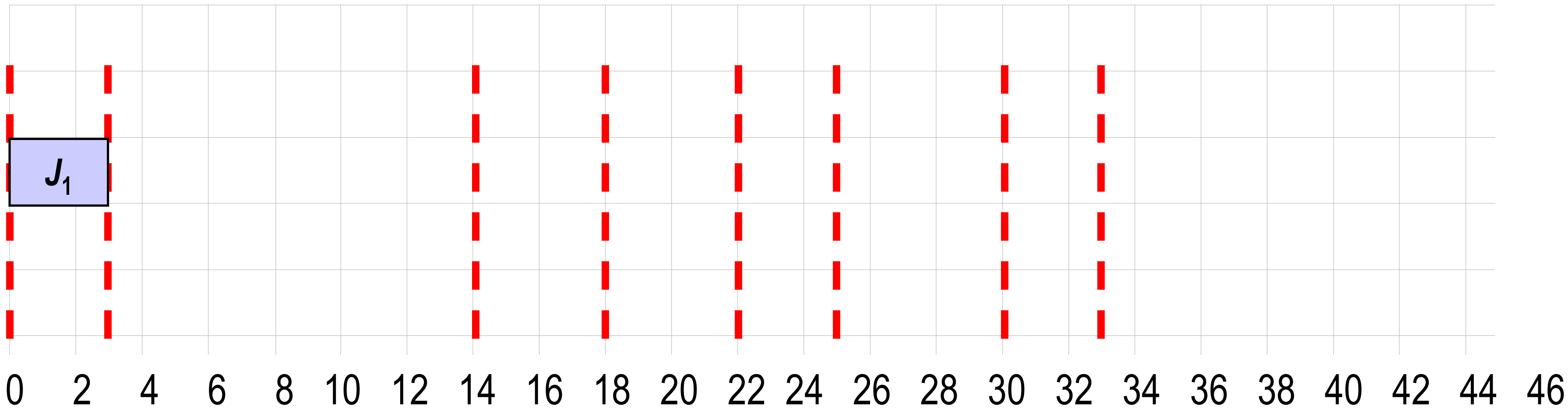
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem **1| r_j , Pmtn| ΣC_j**

	r_j	p_j	Remaining
J_1	0	13	10
J_2	3	6	
J_3	14	8	
J_4	18	3	
J_5	22	5	
J_6	25	3	
J_7	30	4	
J_8	33	1	

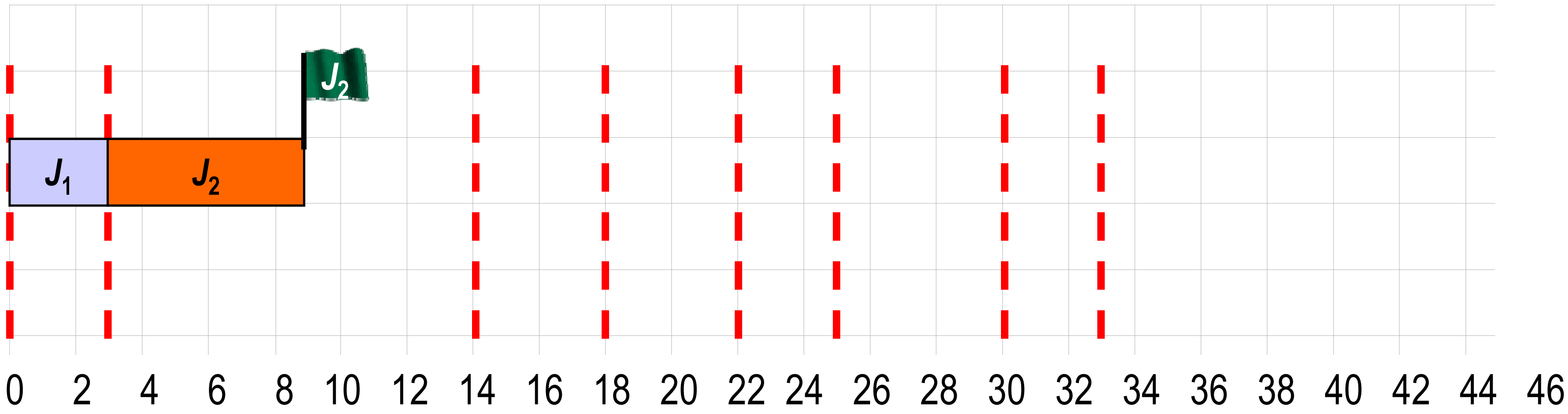
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem $1| r_j , \mathbf{Pmtn}| \Sigma C_j$

	r_j	p_j	Remaining
J_1	0	13	10
J_2	3	6	0
J_3	14	8	
J_4	18	3	
J_5	22	5	
J_6	25	3	
J_7	30	4	
J_8	33	1	

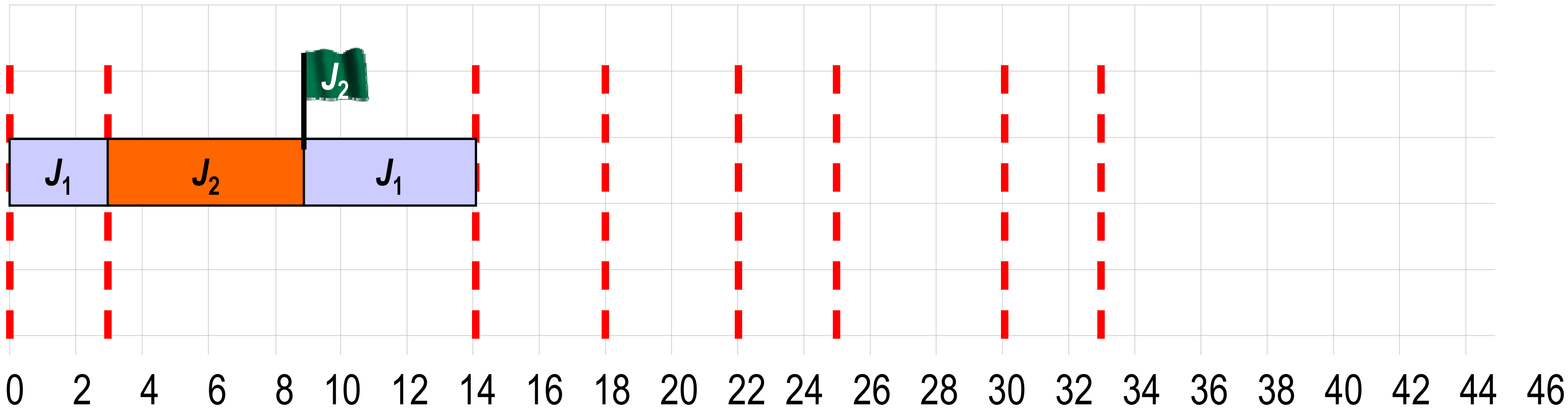
***Shortest Remaining Processing Time** first (SRPT) rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem **1| r_j , Pmtn| ΣC_j**

	r_j	p_j	Remaining
J_1	0	13	**10 5**
J_2	3	6	**0**
J_3	14	8	
J_4	18	3	
J_5	22	5	
J_6	25	3	
J_7	30	4	
J_8	33	1	

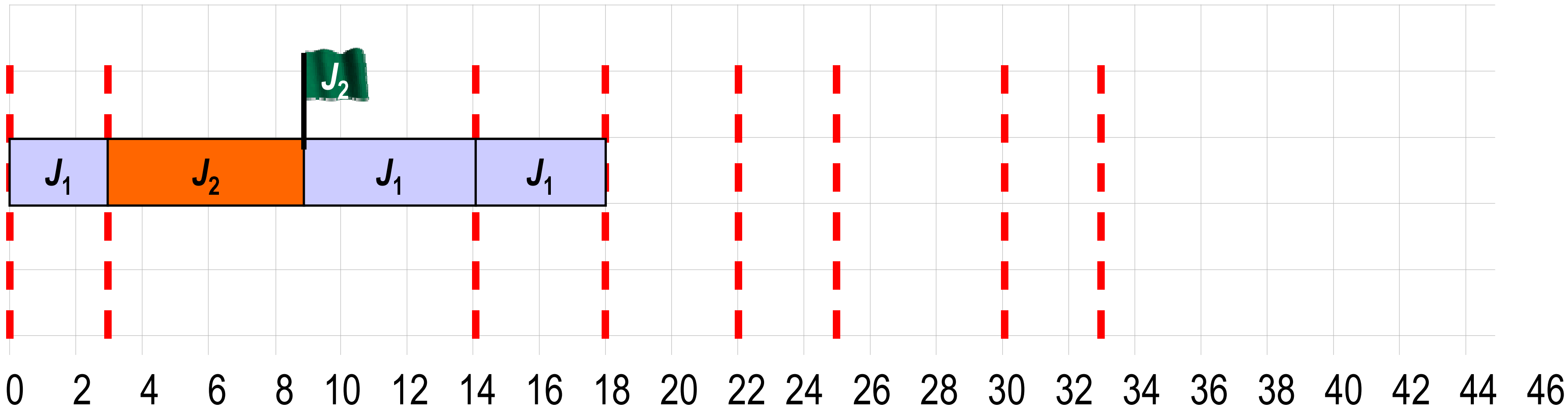
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem 1| r_j , Pmtn| ΣC_j

	r_j	p_j	Remaining
J_1	0	13	10 5 1
J_2	3	6	0
J_3	14	8	
J_4	18	3	
J_5	22	5	
J_6	25	3	
J_7	30	4	
J_8	33	1	

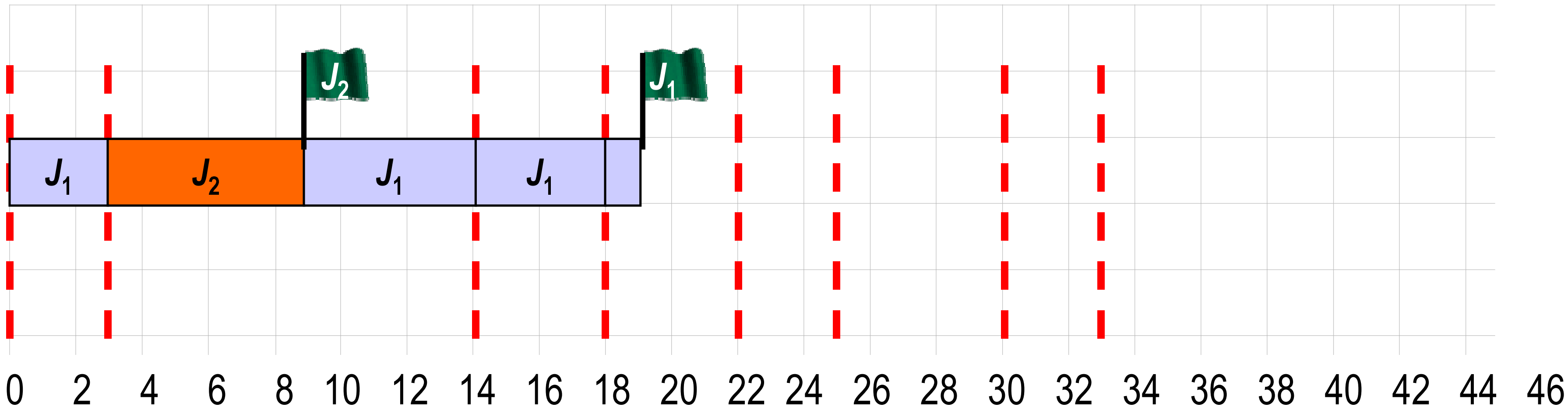
***Shortest Remaining Processing Time** first (SRPT) rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem **1| r_j , Pmtn| ΣC_j**

	r_j	p_j	Remaining
J_1	0	13	10 5 1 0
J_2	3	6	0
J_3	14	8	
J_4	18	3	
J_5	22	5	
J_6	25	3	
J_7	30	4	
J_8	33	1	

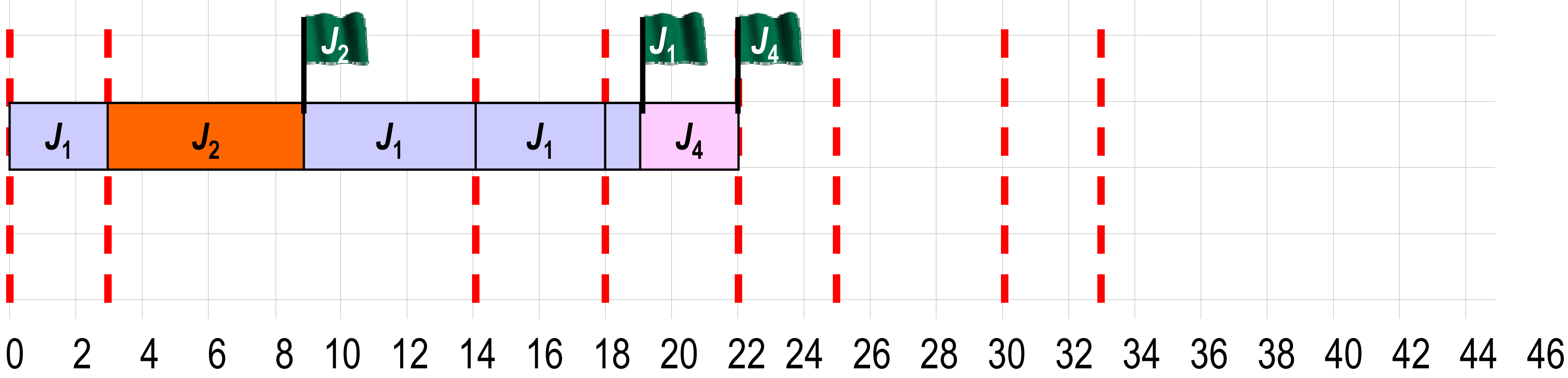
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem $1| r_j, \text{Pmtn}| \Sigma C_j$

	r_j	p_j	Remaining			
J_1	0	13	10	5	1	0
J_2	3	6	0			
J_3	14	8				
J_4	18	3	0			
J_5	22	5				
J_6	25	3				
J_7	30	4				
J_8	33	1				

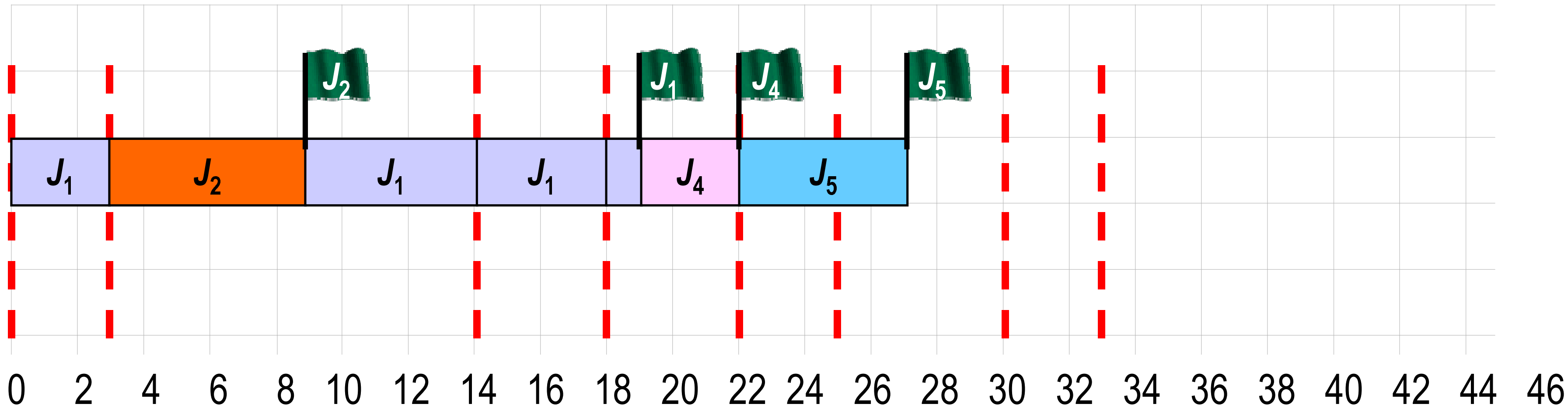
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem **1| r_j , Pmtn| ΣC_j**

	r_j	p_j	Remaining
J_1	0	13	10 5 1 0
J_2	3	6	0
J_3	14	8	
J_4	18	3	0
J_5	22	5	0
J_6	25	3	
J_7	30	4	
J_8	33	1	

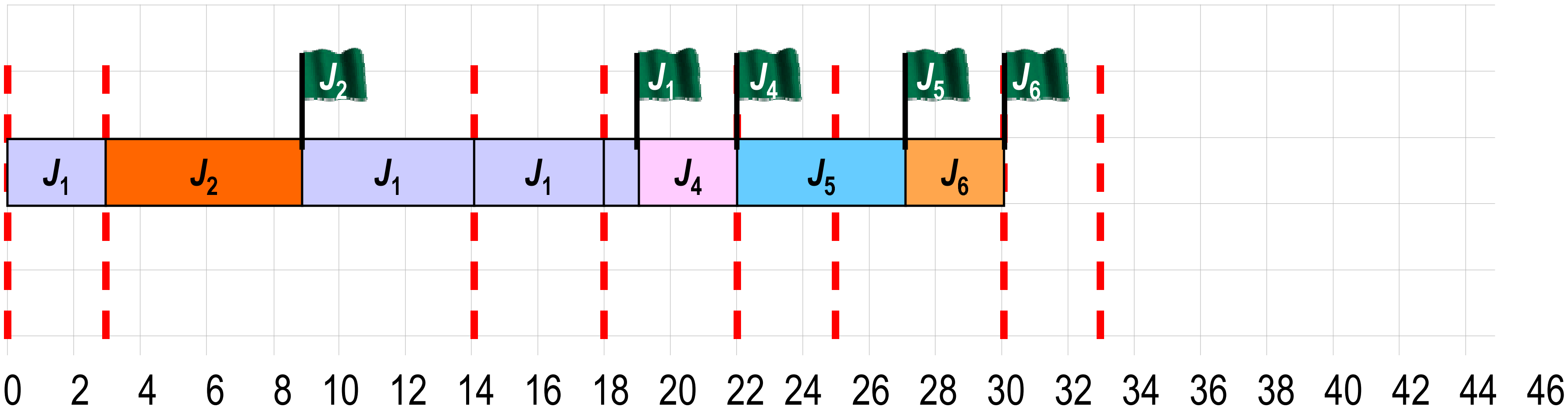
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem **1| r_j , Pmtn| ΣC_j**

	r_j	p_j	Remaining			
J_1	0	13	10	5	1	0
J_2	3	6	0			
J_3	14	8				
J_4	18	3	0			
J_5	22	5	0			
J_6	25	3	0			
J_7	30	4				
J_8	33	1				

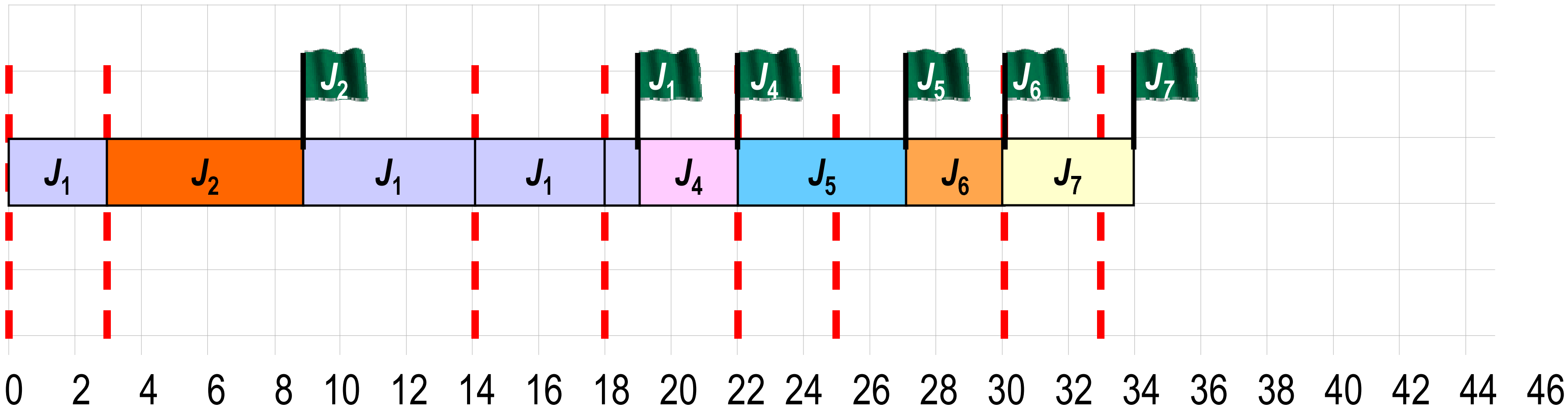
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem **1| r_j , Pmtn| ΣC_j**

	r_j	p_j	Remaining
J_1	0	13	10 5 1 0
J_2	3	6	0
J_3	14	8	
J_4	18	3	0
J_5	22	5	0
J_6	25	3	0
J_7	30	4	0
J_8	33	1	

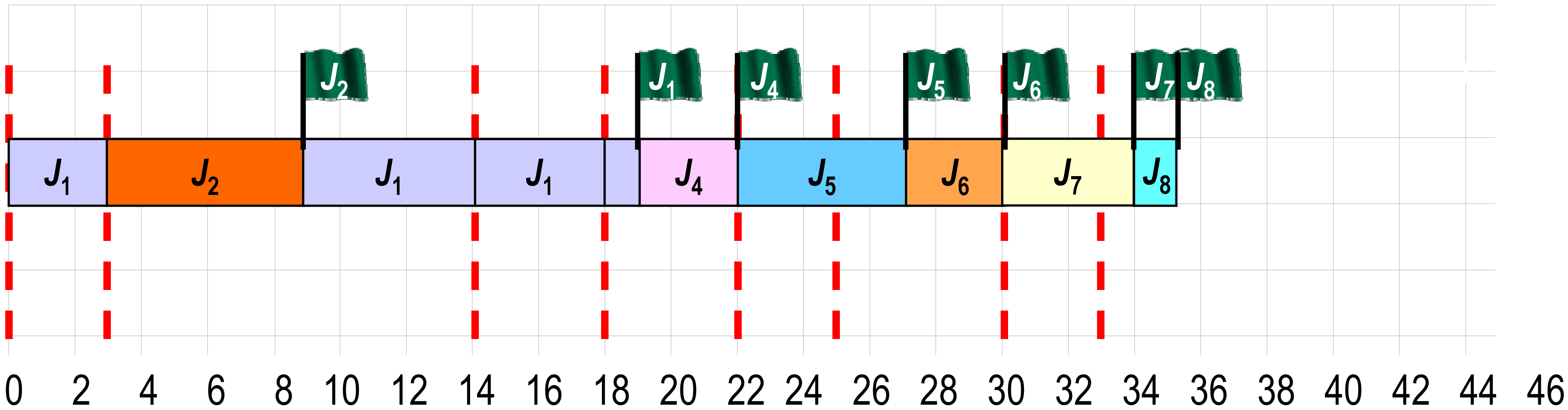
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.



4. SRPT-rule for problem $1|\ r_j\ ,\ Pmtn|\ \Sigma C_j$

	r_j	p_j	Remaining
J_1	0	13	10 5 1 0
J_2	3	6	0
J_3	14	8	
J_4	18	3	0
J_5	22	5	0
J_6	25	3	0
J_7	30	4	0
J_8	33	1	0

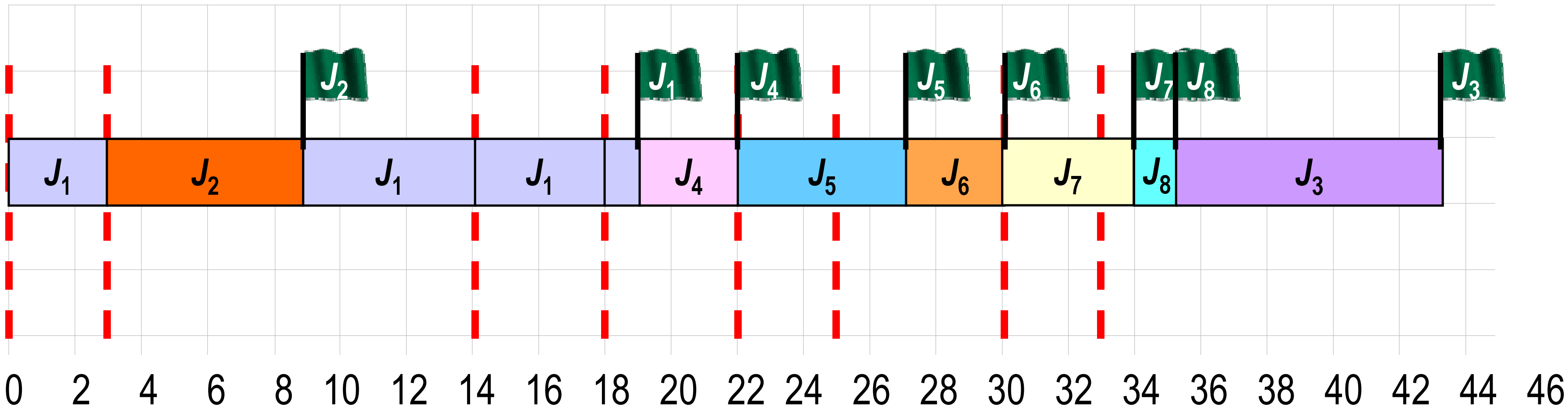
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.

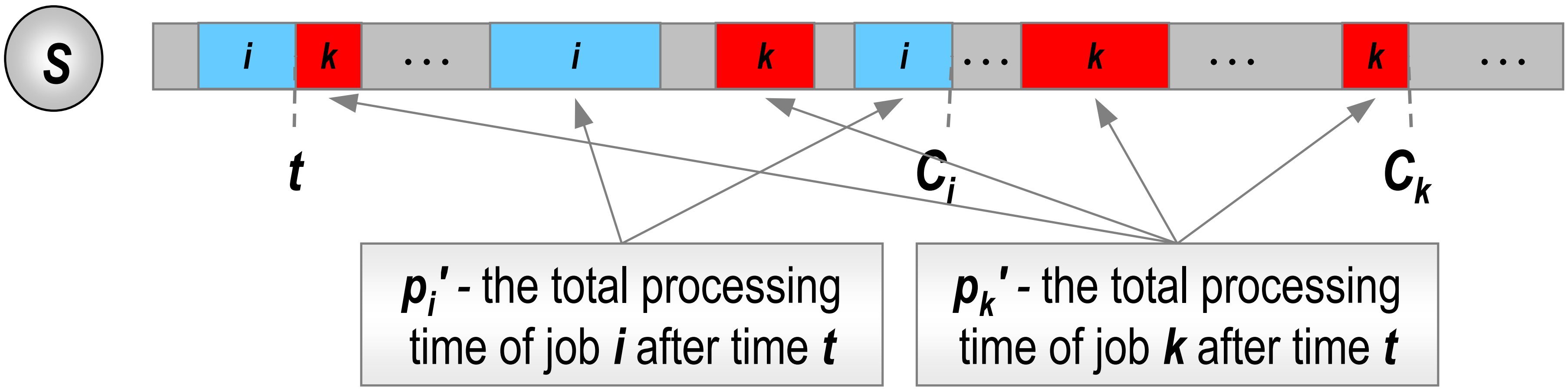


4. SRPT-rule for problem **1| r_j , Pmtn| ΣC_j**

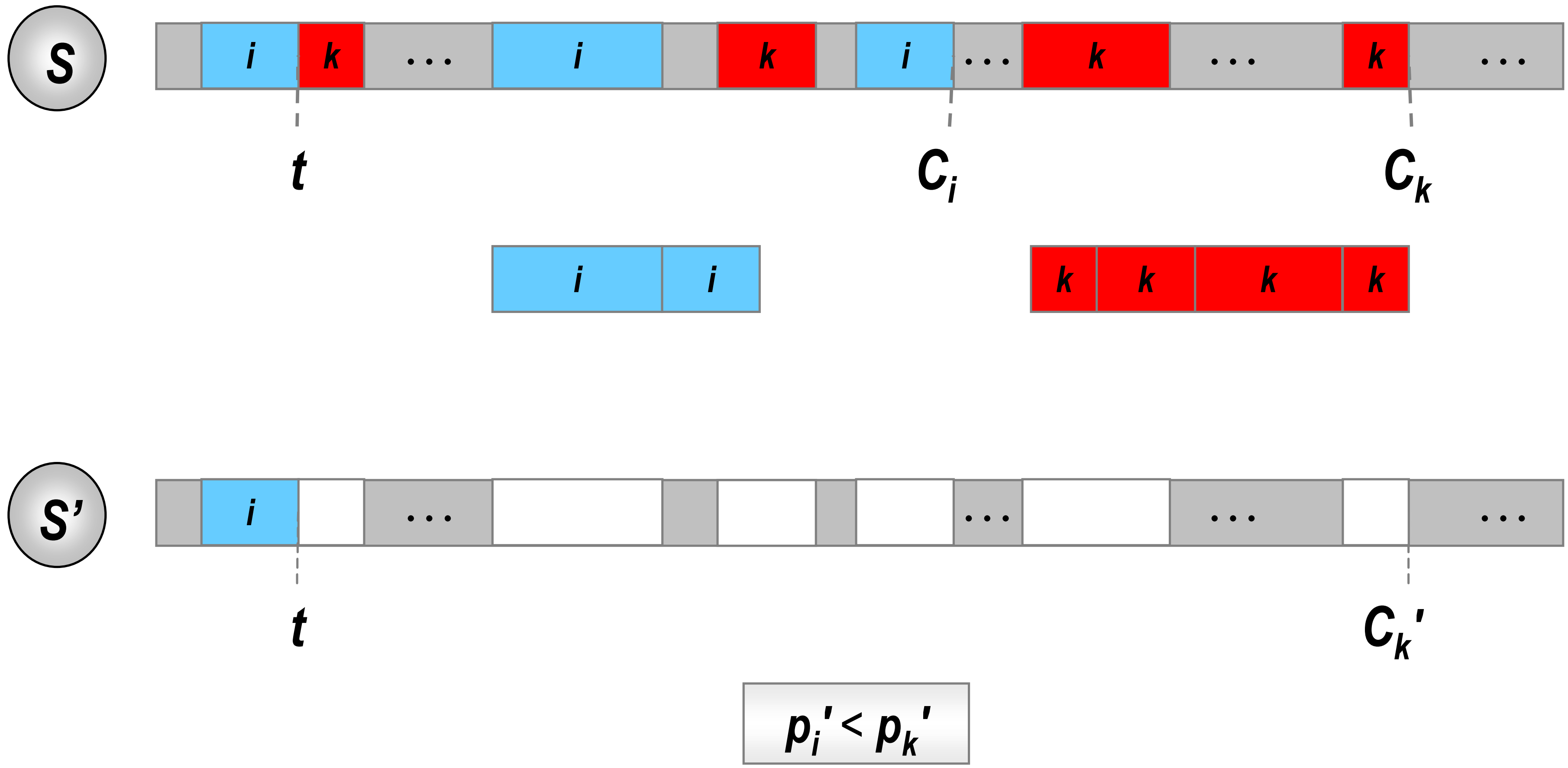
	r_j	p_j	Remaining			
J_1	0	13	**10**	**5**	**1**	**0**
J_2	3	6	**0**			
J_3	14	8	**0**			
J_4	18	3	**0**			
J_5	22	5	**0**			
J_6	25	3	**0**			
J_7	30	4	**0**			
J_8	33	1	**0**			

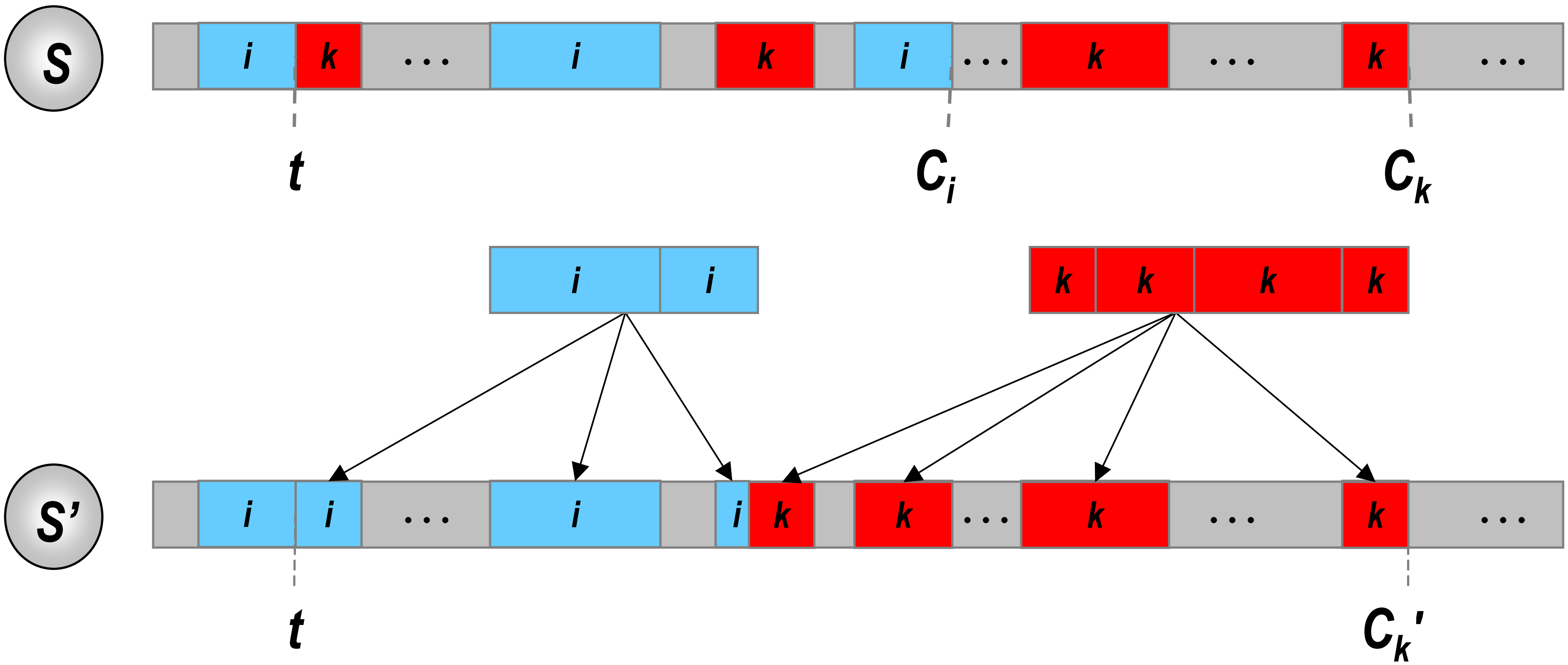
Shortest Remaining Processing Time *first* (*SRPT*) *rule:*
each time that a job is completed, or at the next release date, the job to be processed next has the smallest remaining processing time among the available jobs.





Theorem 2. For $1|\, r_j\, ,\, \mathbf{Pmtn}\, |\, \Sigma C_j$ the SRPT rule is optimal.


S
i
k
. . .
i
k
i
. . .
k
. . .
k
. . .
t
C_i
C_k
i
i
k
k
k
k
S'
i
. . .
. . .
. . .
. . .
t
C_k'
$p_i' < p_k'$



$p_i' < p_k'$

We have obtained a 'better' schedule ***S'***:

$$C_i' < C_i$$
$$C_k' = C_k$$
$$\Sigma C_j' - \Sigma C_j = (C_i' + C_k') - (C_i + C_k) < 0$$