

# Load Balancing Problem

**Input:**  $m$  identical machines:  $M1, M2, \dots, Mm$


$n$  jobs:  $J1, J2, \dots, Jn$

Processing time of each job:  $t_j$  ( $j = 1, 2, \dots, n$ )

Example: 3 machines and 7 jobs ( $t_j = 1, 2, 3, 4, 5, 6, 7$ )

M1  T1 = 12

M2  T2 = 7

M3  T3 = 9

Makespan  $T = \max \{T1, T2, T3\} = 12$

# Greedy Algorithm

Assign a job to the machine with the smallest load in an arbitrary order of jobs.

**Q: How good is this greedy algorithm?**

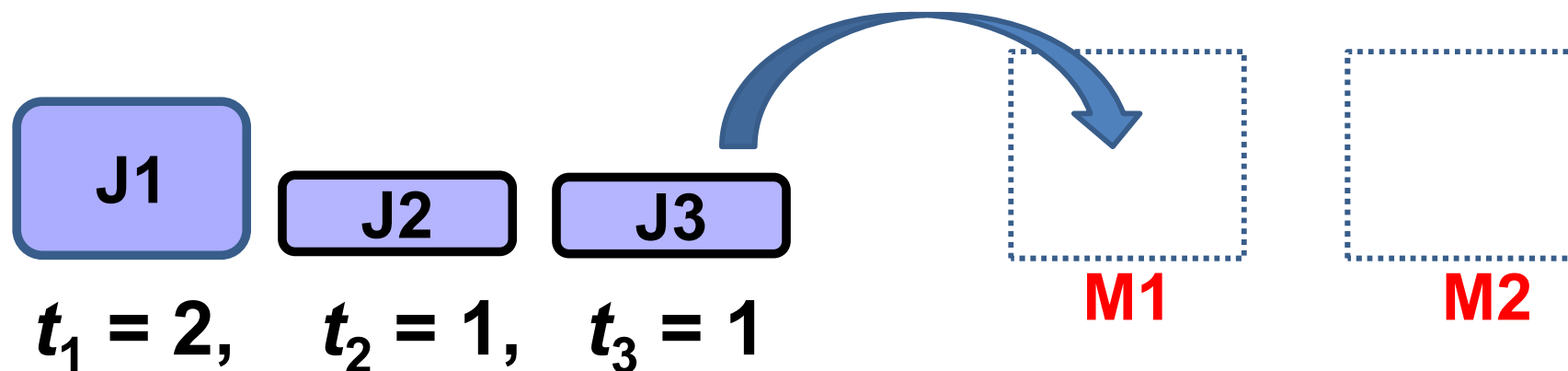
The obtained makespan  $T$  is not worse than  $2T^*$  where  $T^*$  is the optimal makespan (  $T \leq 2T^*$  ): **2-approximation**



“<” holds

## Simplest Example:

Two-Machine Three-Job Problem ( $m = 2, n = 3$ )

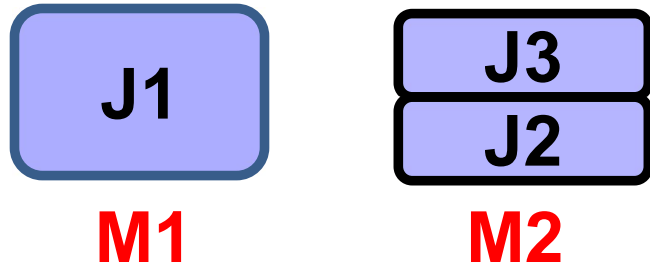


**Q1:** The best makespan obtained by the greedy algorithm?

**Q2:** The worst makespan obtained by the greedy algorithm?

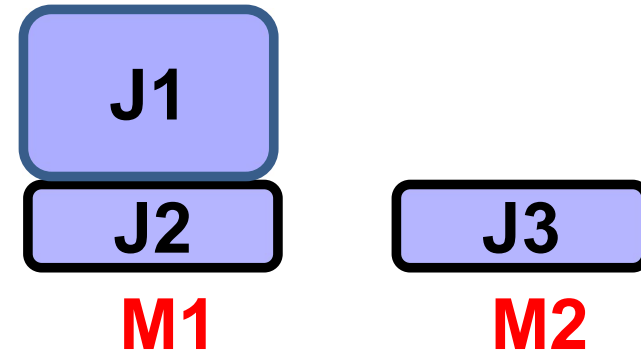
**Q3:** The average makespan by the greedy algorithm?

## Optimal Solution ( $T^* = 2$ )



$T = 2$  by Job Order: J1, J2, J3

## Greedy Solution ( $T = 3$ )



$T = 3$  by Job Order: J2, J3, J1

## How to evaluate the greedy algorithm on this example?

(1) Use of the obtained makespan by an arbitrary order of  $n$  jobs:  $T$ .

Evaluation:  $T/T^* = 1$  or  $1.5$

(2) Use of the makespan by the worst order of  $n$  jobs:  $T_{\text{Worst}}$

Evaluation:  $T_{\text{Worst}}/T^* = 1.5$  (Theoretical upper bound is 2)

(3) Use of the average makespan over all orders of  $n$  jobs:  $T_{\text{Average}}$

$$T_{\text{Average}} = (2 + 2 + 2 + 2 + 3 + 3)/6 = 7/3 = 2.333...$$

$$\text{Evaluation: } T_{\text{Average}}/T^* = 2.333.../2 = 1.166...$$

$T_{\text{Average}}$  is usually much better than  $T_{\text{Worst}}$ .

# Homework Exercise

## Exercise 4-1:

Create a simple example which has a large value of  $T_{\text{Average}}/T^*$  for the greedy algorithm. Create another example to maximize  $T_{\text{Average}}/T^*$  for the greedy algorithm.

# Sorted Greedy Algorithm

Assign a job to the machine with the smallest load in a descending order of jobs (the longest is the first).

**Example:** 3 machines and 5 jobs ( $m = 3, n = 5$ )

$$t_1 = 2, t_2 = 4, t_3 = 5, t_4 = 6, t_5 = 7$$

**Question 1:** Calculate the makespan obtained by the sorted greedy algorithm.

**Question 2:** Calculate the worst makespan obtained by the simple greedy algorithm.

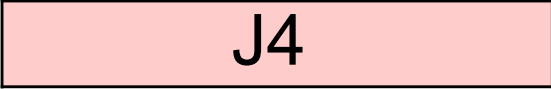

# Sorted Greedy Algorithm

Assign a job to the machine with the smallest load in a descending order of jobs (the longest is the first).

Example: 3 machines and 5 jobs ( $m = 3, n = 5$ )

$$t_1 = 2, t_2 = 4, t_3 = 5, t_4 = 6, t_5 = 7$$

M1  T1 = 7

M2   T2 = 8


M3   T3 = 9 ( $T = 9$ )

Theoretical Analysis: **3/2-Approximation Algorithm**

# Explanation: 3/2-Approximation Algorithm

**Case 1:  $T$  is determined by a machine with a single job.**

M1   $T1 = 10$  ( $T = 10$ )

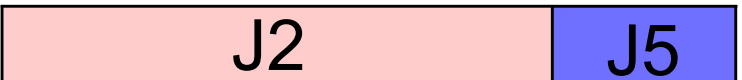
M2   $T2 = 8$

M3   $T3 = 9$

In this case, it is clear that  $T = T^*$  (i.e., the optimal solution is obtained).

**Case 2:  $T$  is determined by a machine with at least two jobs.**

M1   $T1 = 7$



M2   $T2 = 8$

M3   $T3 = 9$  ( $T = 9$ )



## Case 2: $T$ is determined by a machine with at least two jobs.

M1   $T_1 = 7$

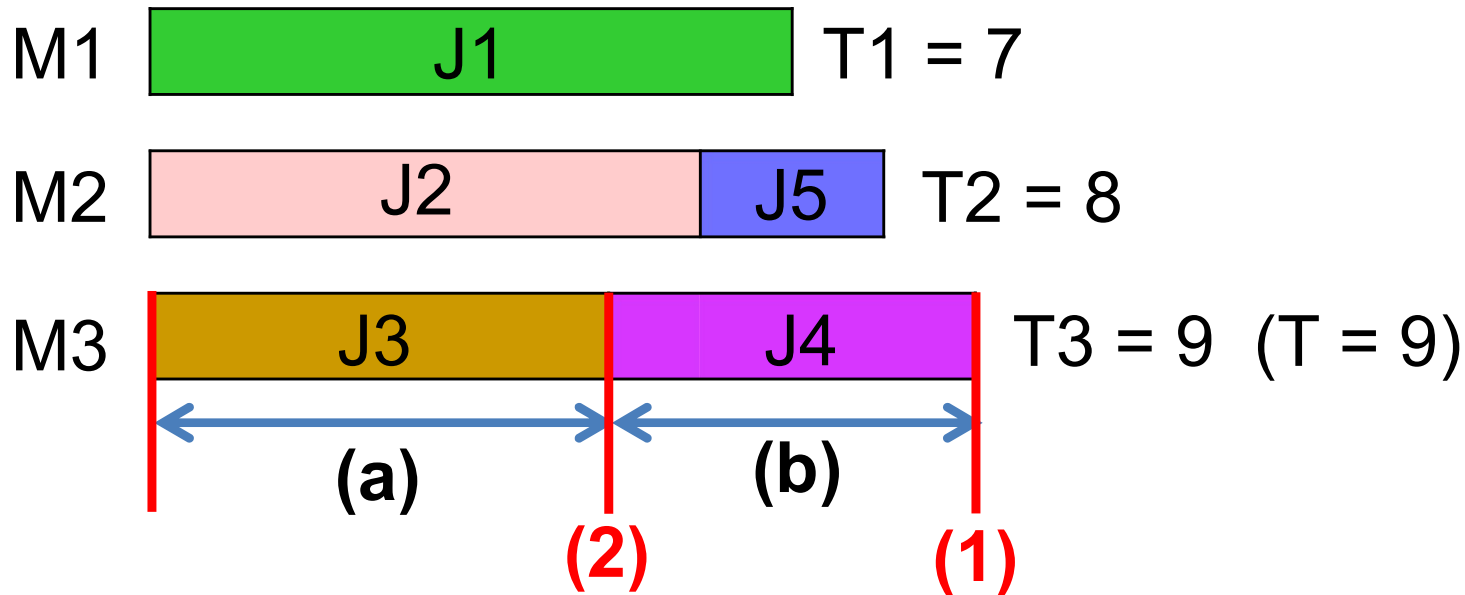
M2    $T_2 = 8$

M3    $T_3 = 9 \quad (T = 9)$

In this case, it is clear that  $n$  (the number of jobs) is larger than  $m$  (the number of machines). That is, we have at least  $m+1$  jobs. This means that one machine has at least two jobs among the first  $(m+1)$  jobs. The smallest processing time among the first  $(m+1)$  jobs is the processing time of the  $(m+1)$ th job:  $t_{m+1}$ . The second smallest processing time among them is the processing time of the  $m$ -th job:  $t_m$ . Since at least one machine (say, machine  $j$ ) has two jobs among the first  $(m+1)$  jobs, we have the following relation:

$$T^* \geq T_j \geq t_m + t_{m+1} \geq 2t_{m+1} \quad \Rightarrow \quad t_{m+1} \leq T^* / 2$$

## Case 2: $T$ is determined by a machine with at least two jobs.



(1) is the makespan obtained by the sorted greedy algorithm.

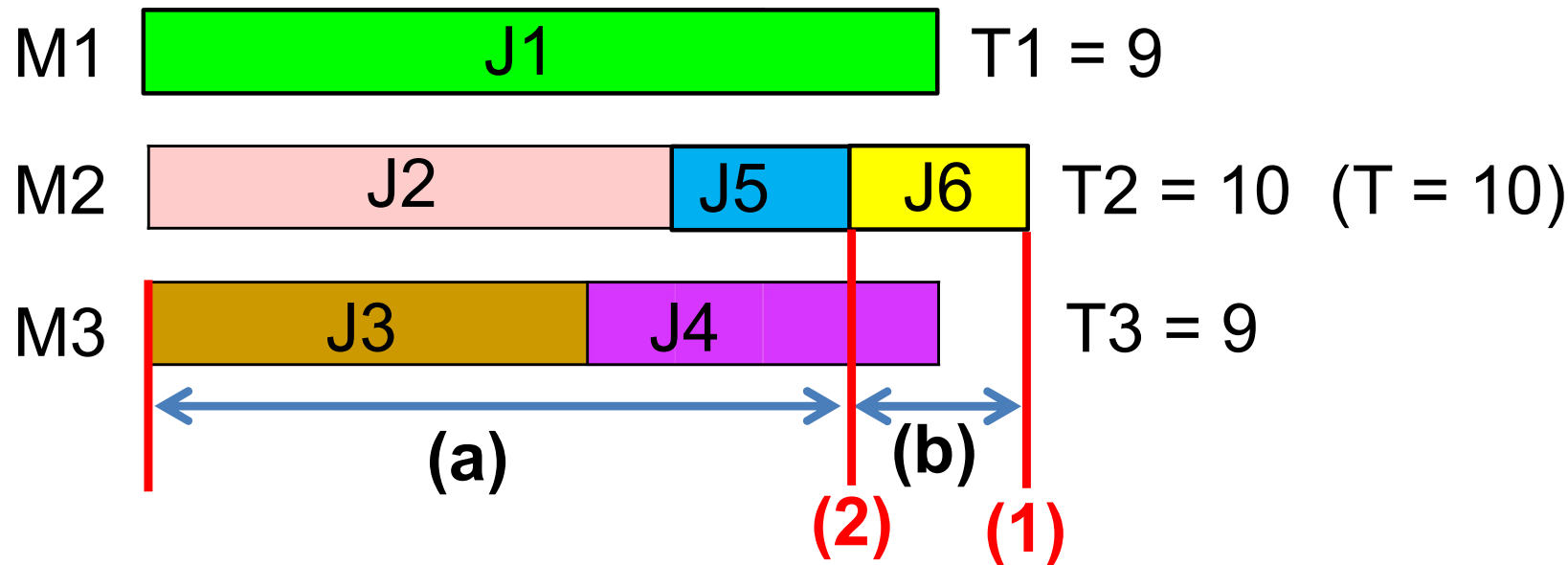
The makespan is determined by M3 in this figure.

(2) is the smallest load when the last job of this machine (i.e., J4) is assigned. Thus, (a) is smaller than the average load over all machines.

J4 is the  $(m+1)$ th or later job.

$$(a) < \frac{1}{m} \sum_{j=1}^n t_j \leq T^*$$

$$(b) \leq t_{m+1} \leq T^* / 2$$



$$(a) < \frac{1}{m} \sum_{j=1}^n t_j \leq T^* \quad (b) \leq t_{m+1} \leq T^*/2$$

**Case 1:** When the makespan  $T$  is the processing time of a single job,  $T = T^*$ .

**Case 2:** When the makespan  $T$  is the sum of the processing times of at least two jobs,  $T = (a) + (b) < 3T^*/2$ .

Thus, the sorted greedy algorithm is a  $3/2$ -approximation algorithm.

## Another Explanation (Slightly Different Proof):

M1  T1 = 11 (T = 11)

M2  T2 = 10

M3  T3 = 9

After the first  $m$  jobs are assigned (i.e., when each machine has a single job), the load of each machine  $T_j$  satisfies the following relation:

$$T_j \leq T^*, \quad j = 1, 2, \dots, m$$

Then, after the  $(m+1)$ th job is assigned to the machine with the current minimum load ( $< T^*$ ), the following relation holds (from the relation between  $T^*$  and  $t_{m+1}$ :  $t_{m+1} \leq T^* / 2$ ):

$$T_j \leq T^* + T^* / 2, \quad j = 1, 2, \dots, m$$

This relation always holds when a new job is added since the current minimum load is smaller than  $T^*$  and the new job has a shorter processing time than  $t_{m+1}$  (or the same as  $t_{m+1}$ ).

## Sorted Greedy Algorithm

Assign a job to the machine with the smallest load in a descending order of jobs (the longest is the first).

**Q: How good is this sorted greedy algorithm?**

The obtained makespan  $T$  is not worse than  $(3/2)T^*$  ( $T \leq (3/2)T^*$ ). **3/2-approximation algorithm.**

**Q: How tight is this upper bound?**

The sorted greedy algorithm is a 4/3-approximation algorithm.  
R. L. Graham. Bounds for multiprocessing timing anomalies. *SIAM J. Applied Mathematics* 17 (1969), 263–269.

**The sorted greedy is a 4/3-approximation algorithm.**

LPT-List-Scheduling( $m, n, t_1, t_2, \dots, t_n$ ) {

Sort jobs so that  $t_1 \geq t_2 \geq \dots \geq t_n$

for  $i = 1$  to  $m$  {

$L_i \leftarrow 0$   $\leftarrow$  load on machine  $i$

$J(i) \leftarrow \phi$   $\leftarrow$  jobs assigned to machine  $i$

}

for  $j = 1$  to  $n$  {

$i = \operatorname{argmin}_k L_k$   $\leftarrow$  machine  $i$  has smallest load

$J(i) \leftarrow J(i) \cup \{j\}$   $\leftarrow$  assign job  $j$  to machine  $i$

$L_i \leftarrow L_i + t_j$   $\leftarrow$  update load of machine  $i$

}

return  $J(1), \dots, J(m)$

}

**procedure** SORTED-BALANCE

Sort jobs in descending order of processing time (so  $t_1 \geq t_2 \geq \dots \geq t_n$ )

**for**  $j = 1, \dots, n$  **do**

Let  $M_i =$  machine that achieves  $\min_k T_k$

Assign job  $j$  to machine  $M_i$

Set  $A(i) \leftarrow A(i) \cup \{j\}$

Set  $T_i \leftarrow T_i + t_j$

**end for**

**end procedure**

## Homework Exercise

### Exercise 4-1:

Create a simple example which has a large value of  $T_{\text{Average}}/T^*$  for the greedy algorithm. Create another example to maximize  $T_{\text{Average}}/T^*$  for the greedy algorithm.

### Exercise 4-2:

Create a simple example to demonstrate the usefulness of the sorted greedy algorithm. Create another example to maximize  $T_{\text{Average}}/T_{\text{Sorted}}$  where  $T_{\text{Average}}$  is the average makespan by the greedy algorithm and  $T_{\text{Sorted}}$  is the makespan obtained by the sorted greedy algorithm. The value of  $T_{\text{Average}}/T_{\text{Sorted}}$  can be viewed as the usefulness of the sorted greedy algorithm. If this value is close to 1, the performance of the sorted greedy algorithm is almost the same as the average performance of the greedy algorithm (i.e., the sorted greedy algorithm is not useful in comparison with the greedy algorithm).



# General Settings of Load Balancing

Each machine has a different processing time of each job (each machine is good at processing some jobs) .

**Example:**

**Three Machines (M1, M2, M3) and 15 Jobs (J1, J2, ..., J15)**

**Processing time ( $t_{ij}$ ):**

	J1	J2	J3	J4	J5	J6	J7	J8	J9	J10	J11	J12	J13	J14	J15
M1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M2	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
M3	3	6	9	12	15	18	21	24	27	30	15	15	15	15	15

## Exercise 4-3

Design an algorithm to solve this example, and explain your algorithm using this example. Then, show your algorithm as a general algorithm where the processing time of job  $j$  on machine  $i$  is given by  $t_{ij}$  ( $i = 1, 2, \dots, m; j = 1, 2, \dots, n$ ).

# General Settings of Load Balancing

**Some machines can process only a part (i.e., subset) of jobs.**

**Example: M1 can process J1, J2, ..., J7**

**M2 can process J1, J2, ..., J8**

**M3 can process all jobs (J1, J2, ..., J10)**

**Processing time ( $t_{ij}$ ): 2, 4, 6, 8, 10, 12, 14 on M1**

**2, 4, 6, 8, 10, 12, 14, 16 on M2**

**1, 2, 3, 4, 5, 6, 7, 8, 9, 10 on M3**

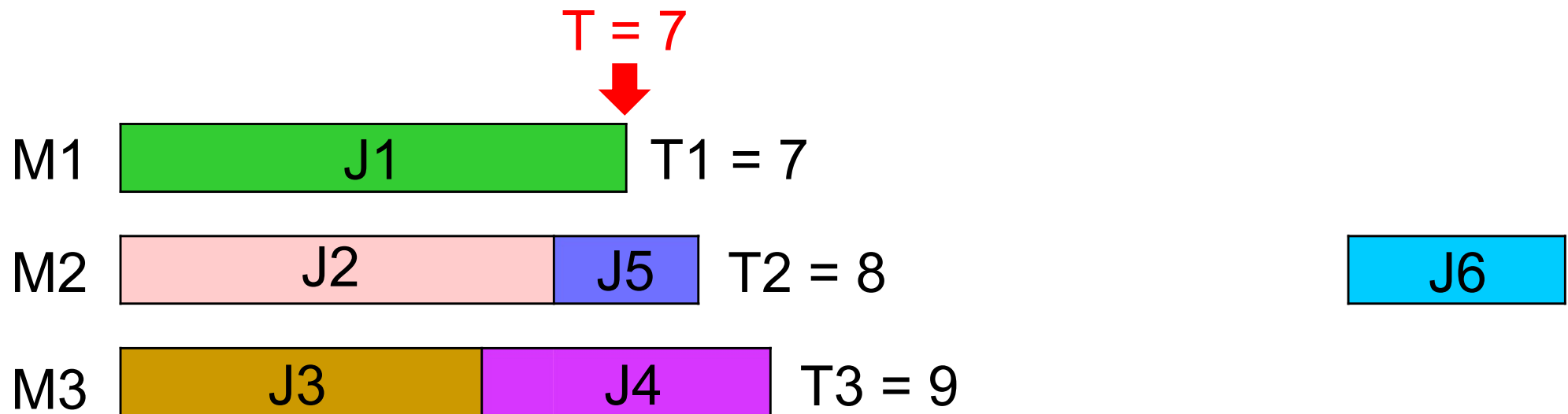
# General Settings of Load Balancing

Each job appears at different time.

**Example 1: Job 6 with  $t_6 = 3$  appears at time 7.**

(We assume that all the other jobs appear at time 0).

$$t_1 = 7, \quad t_2 = 6, \quad t_3 = 5, \quad t_4 = 4, \quad t_5 = 2, \quad t_6 = 3$$



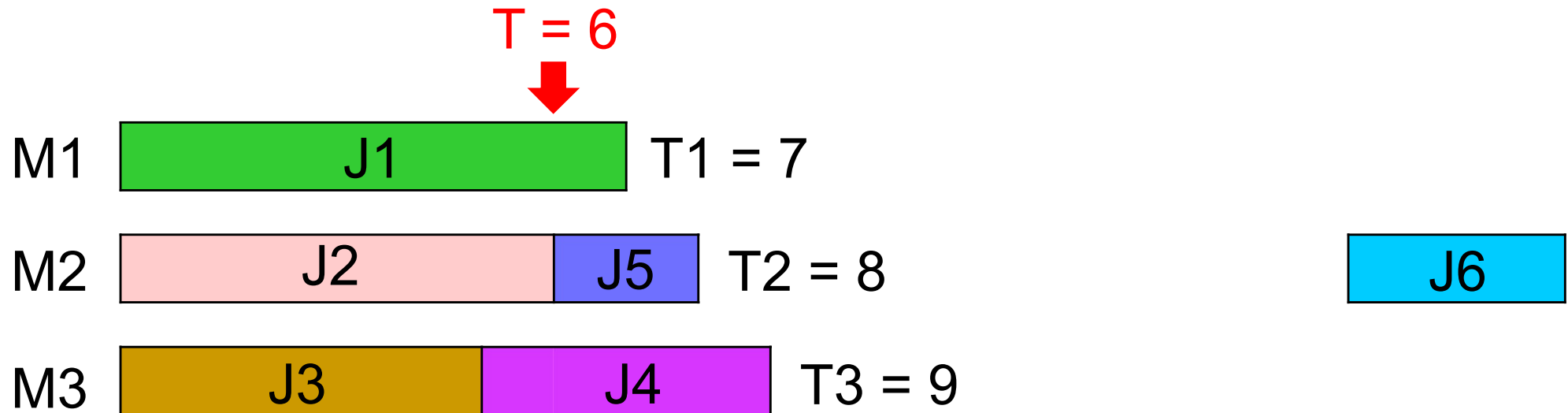
**We need to make a decision at time 7.**

# General Settings of Load Balancing

Each job appears at different time.

**Example 2: Job 6 with  $t_6 = 3$  appears at time 6.**

$$t_1 = 7, t_2 = 6, t_3 = 5, t_4 = 4, t_5 = 2, t_6 = 3$$



**We need to make a decision at time 6.**

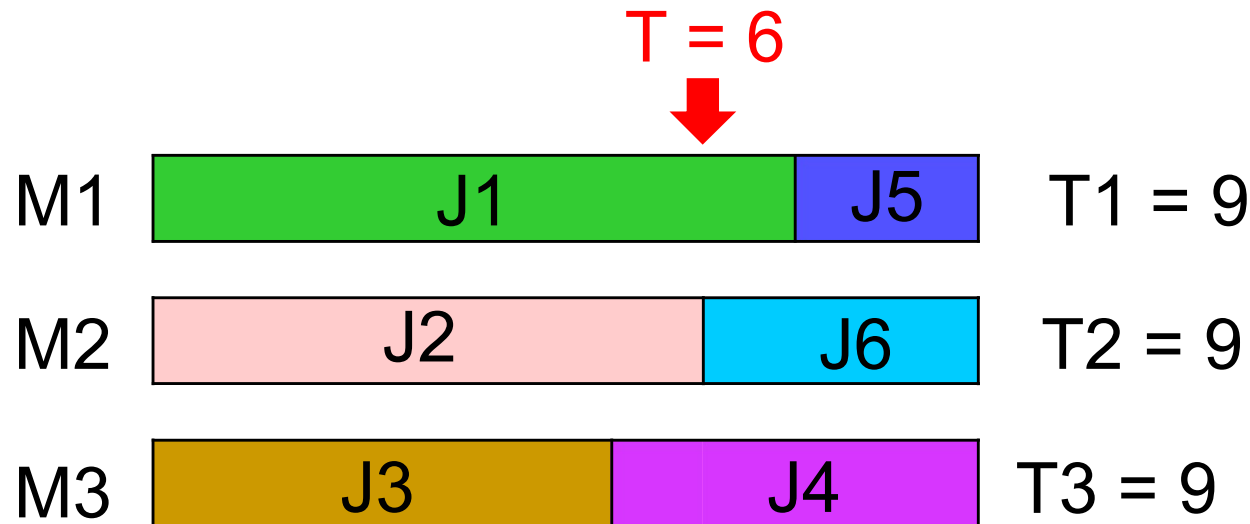
**We cannot change J4 but can change J5 with some cost.**

# General Settings of Load Balancing

Each job appears at different time.

**Example 2: Job 6 with  $t_6 = 3$  appears at time 6.**

$$t_1 = 7, t_2 = 6, t_3 = 5, t_4 = 4, t_5 = 2, t_6 = 3$$



**We need to make a decision at time 6.**

**We cannot change J4 but can change J5 with some cost.**

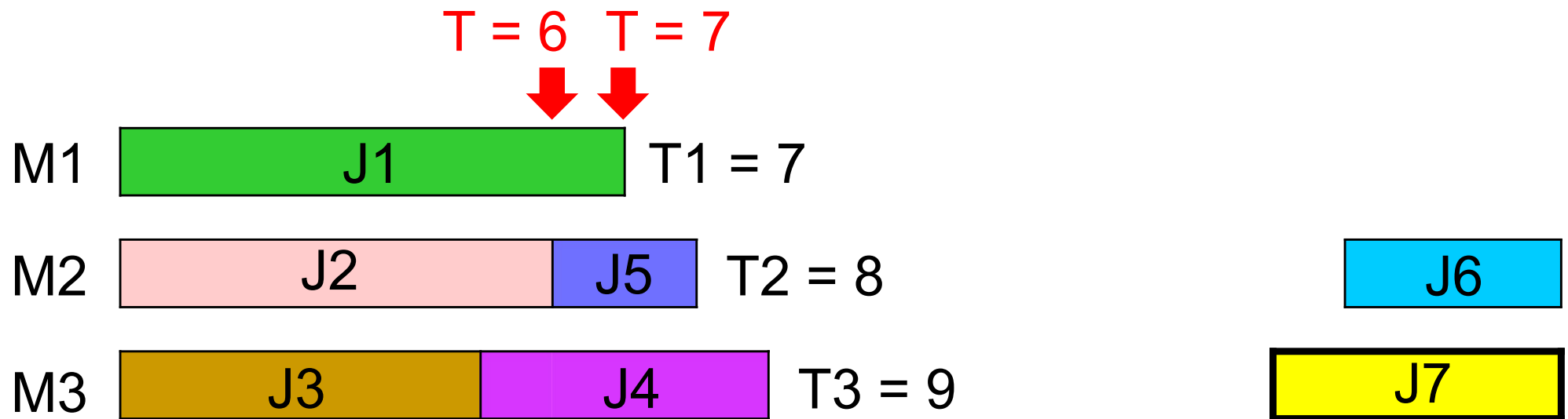
# General Settings of Load Balancing

Each job appears at different time.

**Example 3: Job 6 with  $t_6 = 3$  appears at time 6.**

**Job 7 with  $t_7 = 4$  appears at time 7.**

$$t_1 = 7, t_2 = 6, t_3 = 5, t_4 = 4, t_5 = 2, t_6 = 3, t_7 = 4.$$



**We need to make a decision at time 6 and at time 7.**

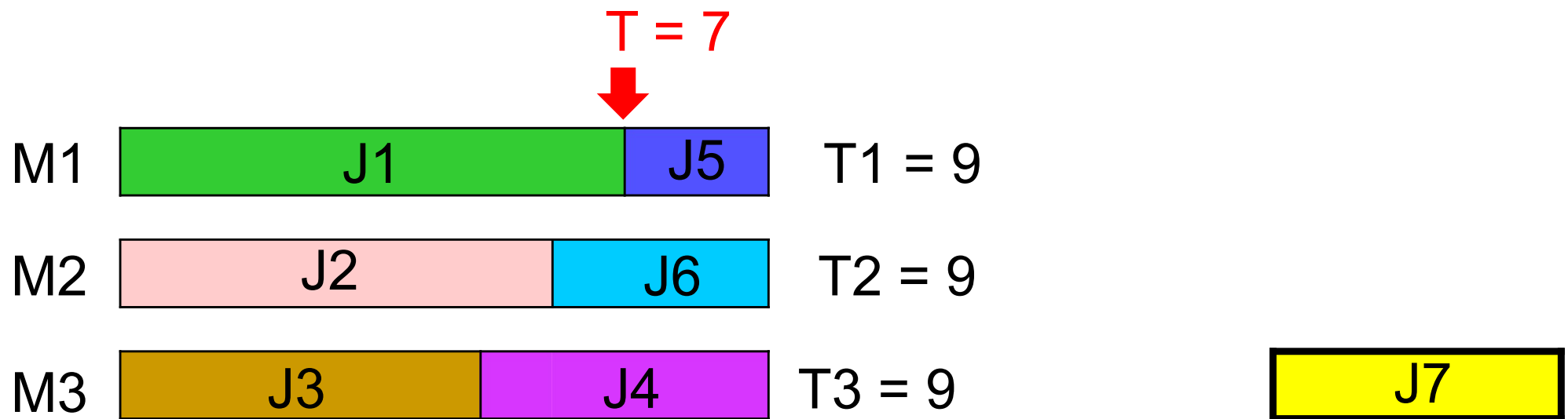
# General Settings of Load Balancing

Each job appears at different time.

**Example 3: Job 6 with  $t_6 = 3$  appears at time 6.**

**Job 7 with  $t_7 = 4$  appears at time 7.**

$t_1 = 7, t_2 = 6, t_3 = 5, t_4 = 4, t_5 = 2, t_6 = 3, t_7 = 4.$



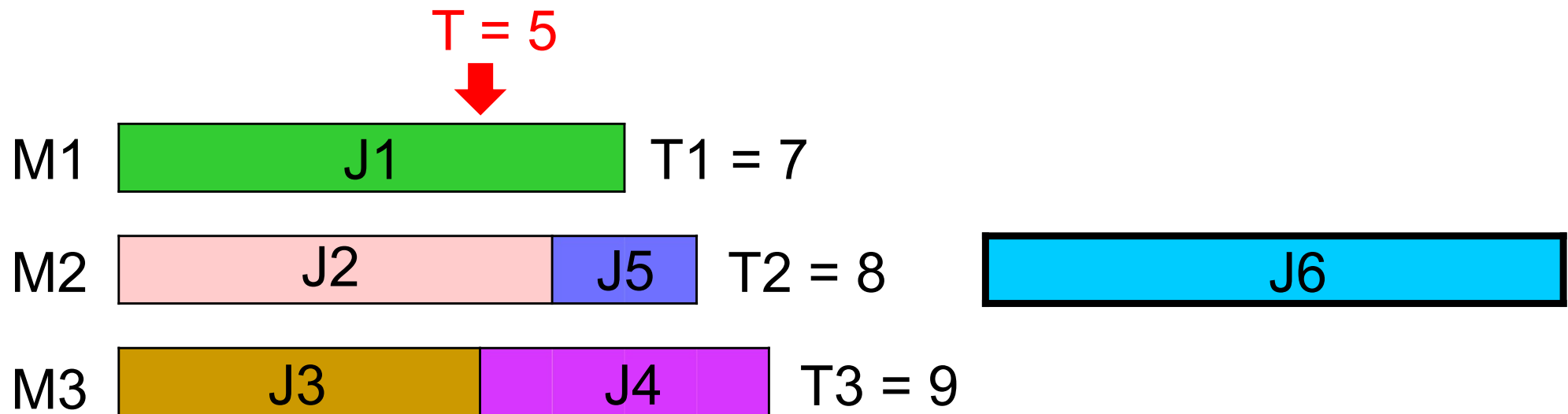
**We need to make a decision at time 6 and at time 7.**

# General Settings of Load Balancing

Each job appears at different time.

**Example 4: Job 6 with  $t_6 = 8$  appears at time 5.**

$$t_1 = 7, t_2 = 6, t_3 = 5, t_4 = 4, t_5 = 2, t_6 = 8$$



**We need to make a decision at time 5.**

**Q. In general, what is a good strategy of load balancing ?**