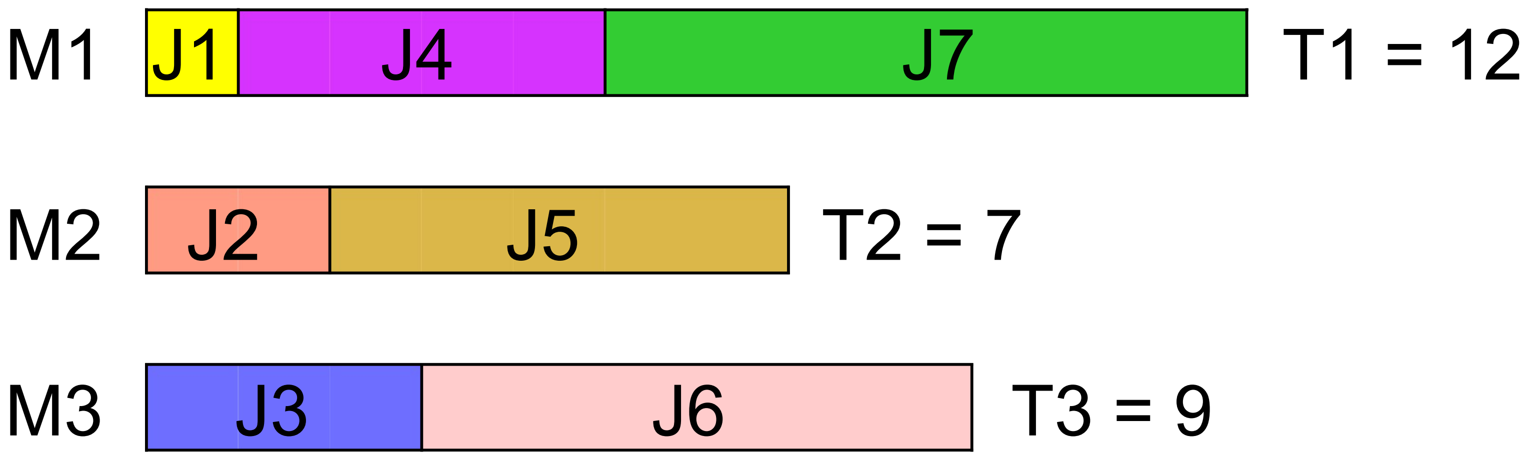
# Topic 1: Load Balancing Problem

**Input:** $m$ identical machines: M1, M2, ..., M$m$

$n$ jobs: J1, J2, ..., J$n$

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

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




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

**Q: What is the best assignment ?**

# Some General Settings of Load Balancing

## Some machines are more efficient than the others

**Example: M3 needs less processing times than the others.**

**Three Machines: M1, M2, M3**

**Ten Jobs: J1, J2, ..., J10**

**Processing time ($t_j$): 2, 4, 6, ..., 20 on M1 and M2**
**1, 2, 3, ..., 10 on M3**

## Some machines can process only a part of jobs.

**Example: M1 can process J1, J2, ..., J7**
**M2 can process J1, J2, ..., J8**
**M3 can process all jobs (J1, J2, ..., J10)**

# Some General Settings of Load Balancing

**Some machines are more efficient than the others**

**Example: M3 needs less processing times than the others.**

**Three Machines: M1, M2, M3**

**Ten Jobs: J1, J2, ...., J10**

**Processing time ($t_j$): 2, 4, 6, ..., 20 on M1 and M2**
**1, 2, 3, ..., 10 on M3**

**Some machines can process only a part of jobs.**

**Example: M1 can process J1, J2, ..., J7**
**M2 can process J1, J2, ..., J8**
**M3 can process all jobs (J1, J2, ..., J10)**

# Topic 6: Generalized Load Balancing Problem (Use of LP)

**Input:** Set of $m$ machines: $M$ = {Machine 1, ..., Machine $m$}
Set of $n$ jobs: $J$ = {Job 1, ..., Job $n$}
Processing time of each job: $t_j$ ($j$ = 1, 2, ..., $n$)
Subset of $M$ for each job: $M_j$ ($j$ = 1, 2, ..., $n$)

**Constraint:** Job $j$ should be assigned to a machine in $M_j$

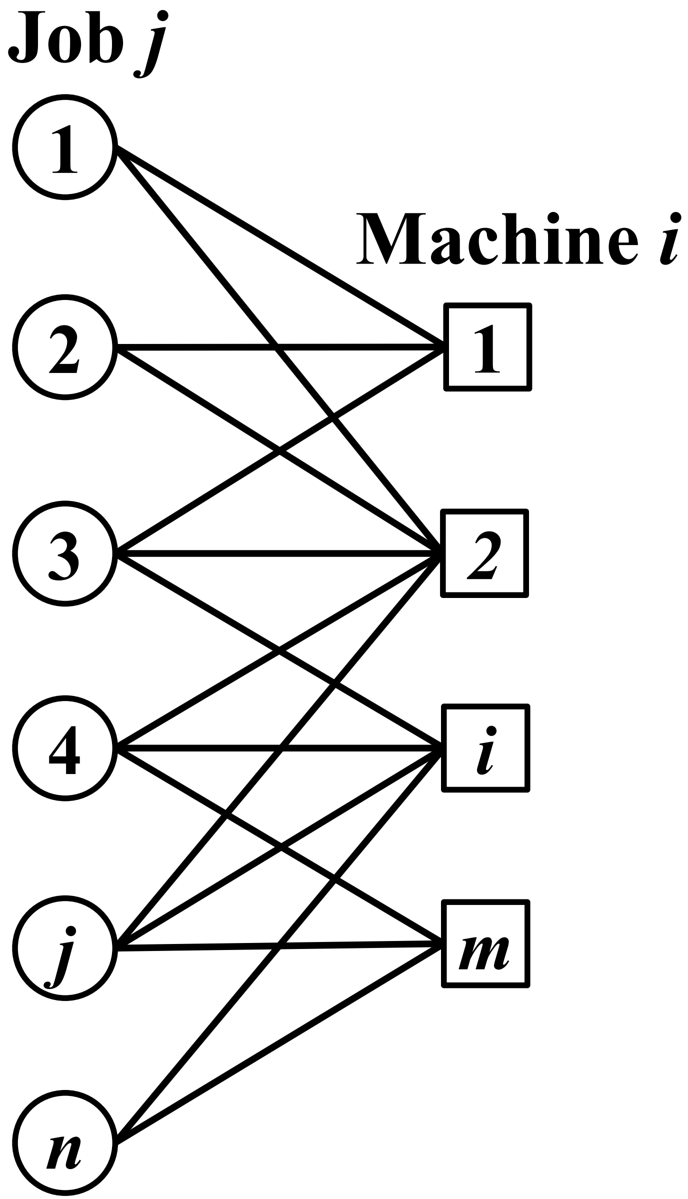
**Objective:** Minimization of the makespan

**Output:** Assignment of $n$ jobs to $m$ machines

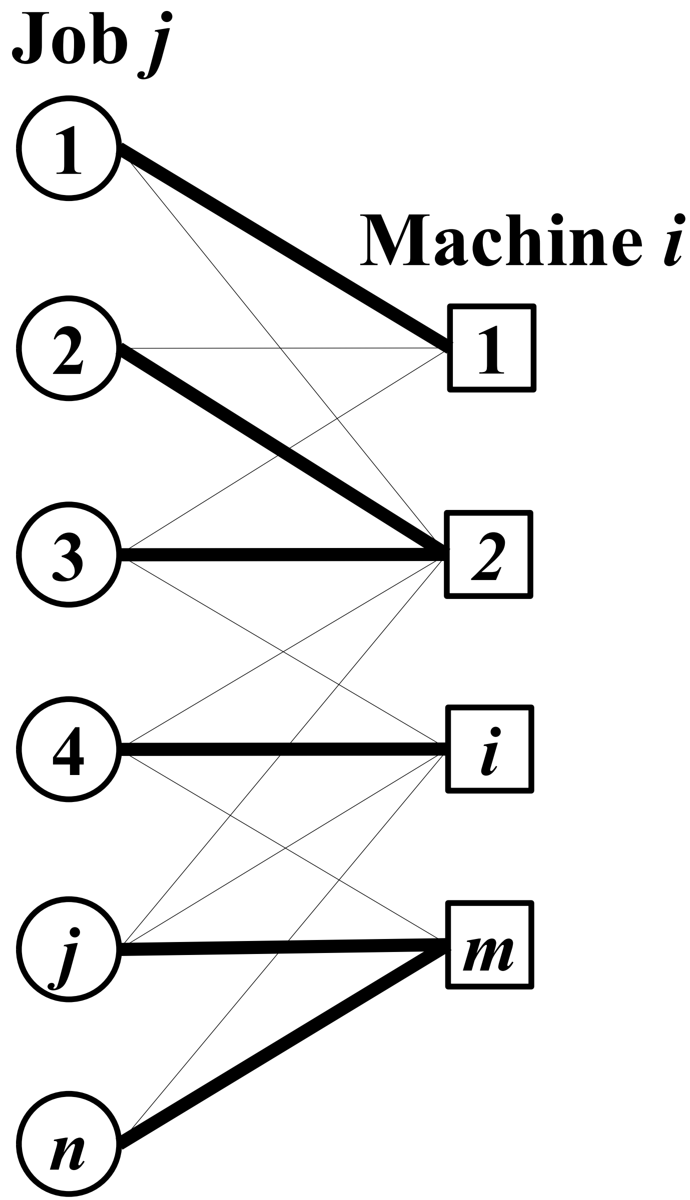
Let $J_i$ be the set of jobs assigned to machine $i$. The load of machine $i$ is $L_i = \sum_{j \in J_i} t_j$. The objective is to minimize $\mathrm{Max}_i L_i$.

***$L^*$*: Optimal value of the objective function** $(L^* \geq \mathrm{Max}_j \, t_j)$

## Qualified machines for each job




## Example of assignment



**Input:** Set of $m$ machines: $M$ = {Machine 1, ..., Machine $m$}

Set of $n$ jobs: $J$ = {Job 1, ..., Job $n$}

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

Subset of $M$ for each job: $M_j$ ($j$ = 1, 2, ..., $n$)

**Constraint:** Job $j$ should be assigned to a machine in $M_j$

**Objective:** Minimization of the makespan

**Output:** Assignment of $n$ jobs to $m$ machines

Let $J_i$ be the set of jobs assigned to machine $i$. The load of machine $i$ is $L_i = \sum_{j \in J_i} t_j$ . The objective is to minimize $\text{Max}_i L_i$ .

## Objective Function

$$\text{Minimize } \text{Max}\{L_i = \sum_{j \in J_i} t_j \mid i = 1, 2, \ldots, m\}$$

$$\text{Minimize } L \quad \text{subject to } \sum_{j \in J_i} t_j \le L \text{ for } i = 1, 2, \ldots, m$$

# Formulation using 0-1 variables $x_{ij}$

Minimize $L$

Each job should select one machine.

subject to $\sum_{i=1}^{m} x_{ij} = 1$ for all $j \in J$ (job $j$ in $J$)

The total processing time at each machine is equal to or smaller than $L$.

$\sum_{j=1}^{n} x_{ij} \cdot t_j \leq L$ for all $i \in M$ (machine $i$ in $M$)

$x_{ij} \in \{0, 1\}$ for all $j \in J,\ i \in M_j$

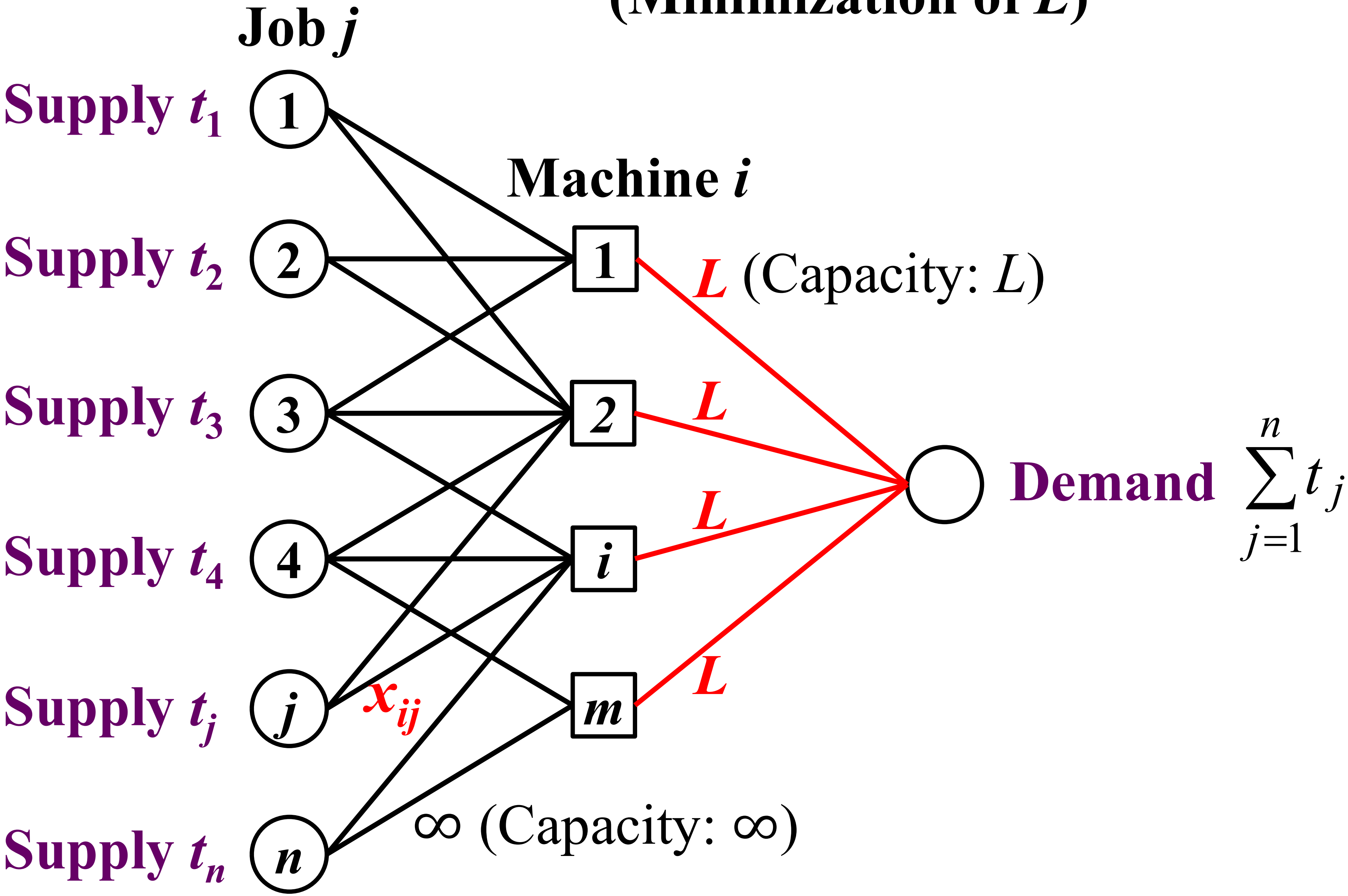
$x_{ij} = 0$ for all $j \in J,\ i \notin M_j$

$L$: variable for formulating the objective function

$x_{ij}$: 0-1 variable for denoting the selection of machine $i$ by job $j$

# Flow Formulation using real number variables $x_{ij}$

**(Minimization of $L$)**




$x_{ij}$ : Flow from job $j$ to machine $i$

($x_{ij}$ is not the 0-1 variable in the previous page)

# GL-IP: Generalized Load Balancing as an Integer Problem

Minimize $L$

Flow from each job $j$ is $t_j$.

subject to $$\sum_{i=1}^{m} x_{ij} = t_j \quad \text{for all } j \in J \quad (\text{job } j \text{ in } J)$$

Total flow to machine $i$.

$$\sum_{j=1}^{n} x_{ij} \leq L \quad \text{for all } i \in M \quad (\text{machine } i \text{ in } M)$$

$$x_{ij} \in \{0, t_j\} \quad \text{for all } j \in J,\ i \in M_j$$

$$x_{ij} = 0 \qquad \text{for all } j \in J,\ i \notin M_j$$

$L$: variable for formulating the objective function

$x_{ij}$: variable for denoting the flow from job $j$ to machine $i$

**$L^*$: Optimal value of the objective function**

# GL-LP: Linear Programming Relaxation of GL-IP

Minimize $L$

subject to $\sum_{i=1}^{m} x_{ij} = t_j$ for all $j \in J$ (job $j$ in $J$)

$\sum_{j=1}^{n} x_{ij} \le L$ for all $i \in M$ (machine $i$ in $M$)

$\boxed{x_{ij} \in \{0, t_j\} \Rightarrow x_{ij} \ge 0}$ for all $j \in J,\ i \in M_j$

$x_{ij} = 0$ for all $j \in J,\ i \notin M_j$

$$L^* \ge L^*_{LP}$$

$L^*_{LP}$ : Optimal value of the relaxation problem

$L^*$: Optimal value of the original problem

**Question:** Why $x_{ij} \ge 0$ instead of $0 \le x_{ij} \le t_j$ ?

$$\boxed{x_{ij} \in \{0, t_j\} \Rightarrow x_{ij} \geq 0}$$

If the solution of the relaxation problem satisfies

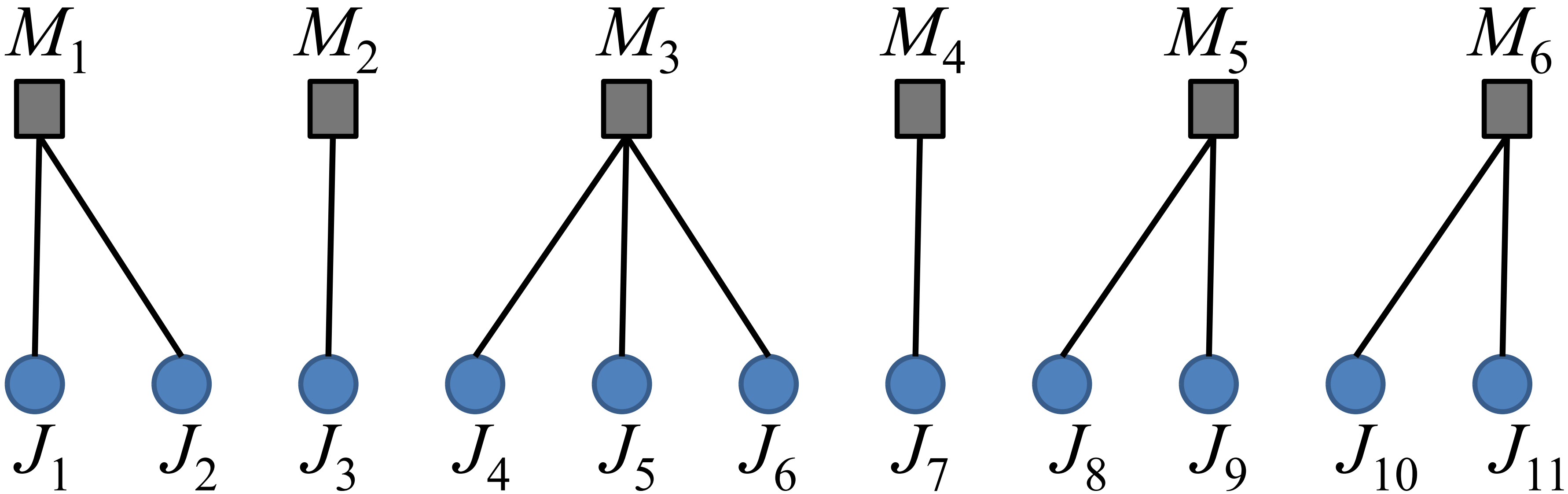
$$x_{ij} \in \{0, t_j\} \text{ for all } j \in J,\ i \in M_j$$

the solution is the optimal solution of the original problem.

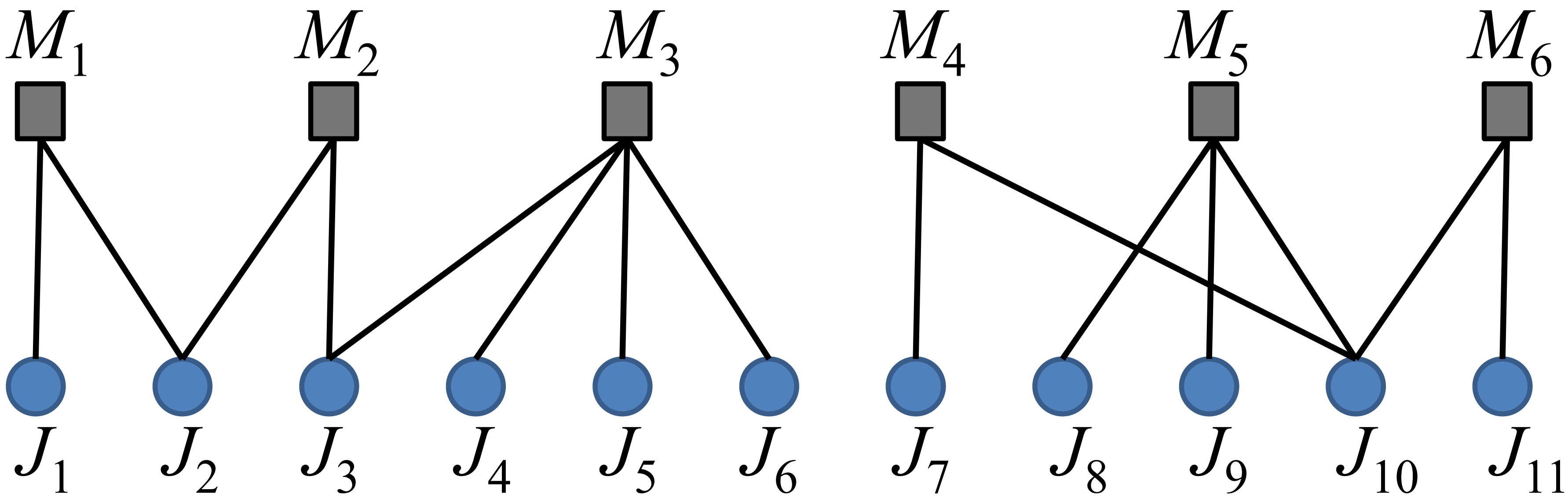
However, in general, this condition is not satisfied. That is, $t_j$ is distributed over multiple machines (e.g., $t_j = 10$ and $x_{1j} = 2.1$, $x_{2j} = 3.8$, $x_{3j} = 4.1$), which is not a feasible solution.

**Question: How to obtain a feasible solution from the obtained solution of the relaxation problem?**

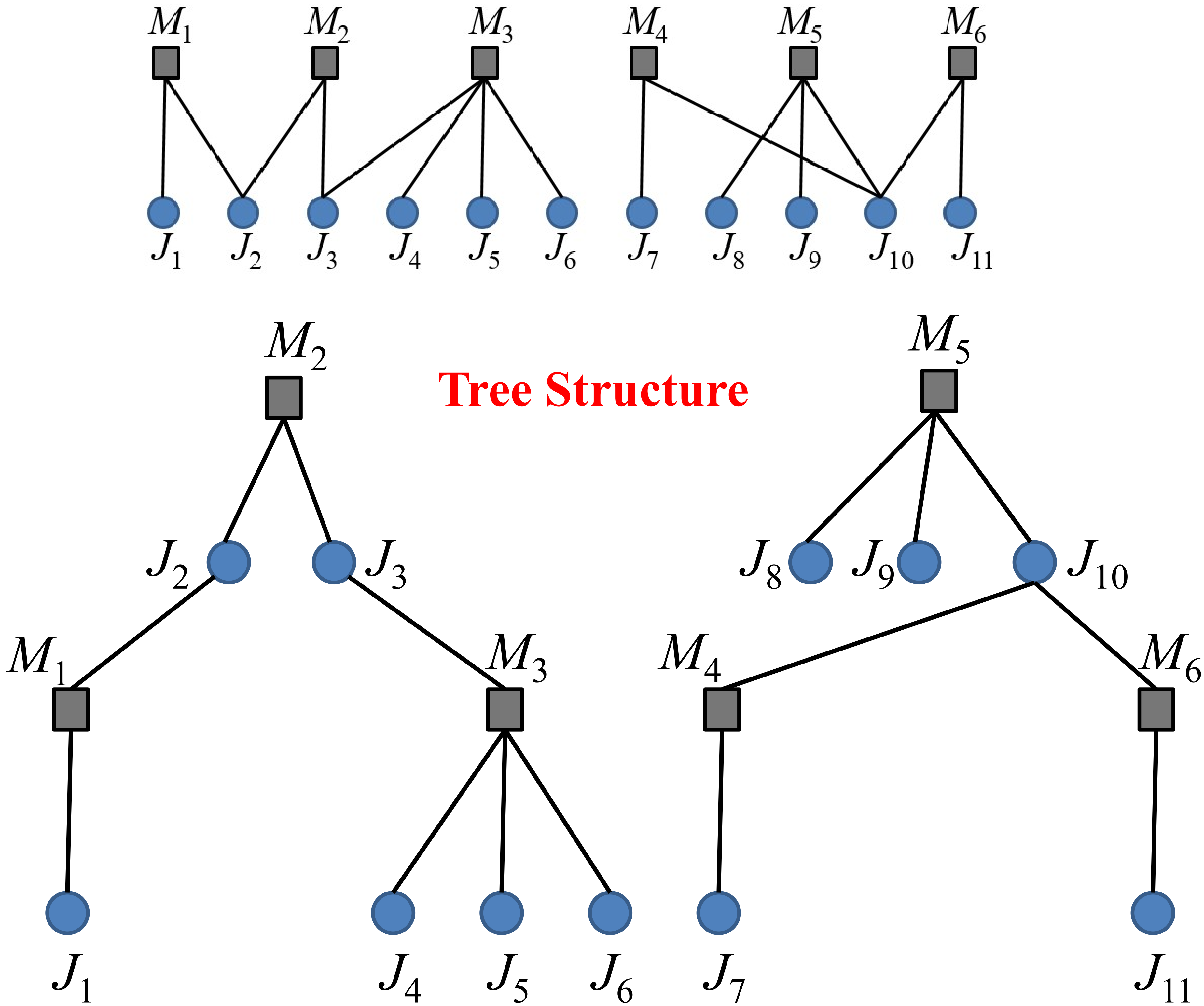
Construct a graph $G(x) = (V(x), E(x))$ where $V(x) = M \cup J$ and $(i, j) \in E(x)$ if and only if $x_{ij} > 0$. If the graph has no cycles (i.e., if the graph is acyclic), each of its connected components has a tree structure.



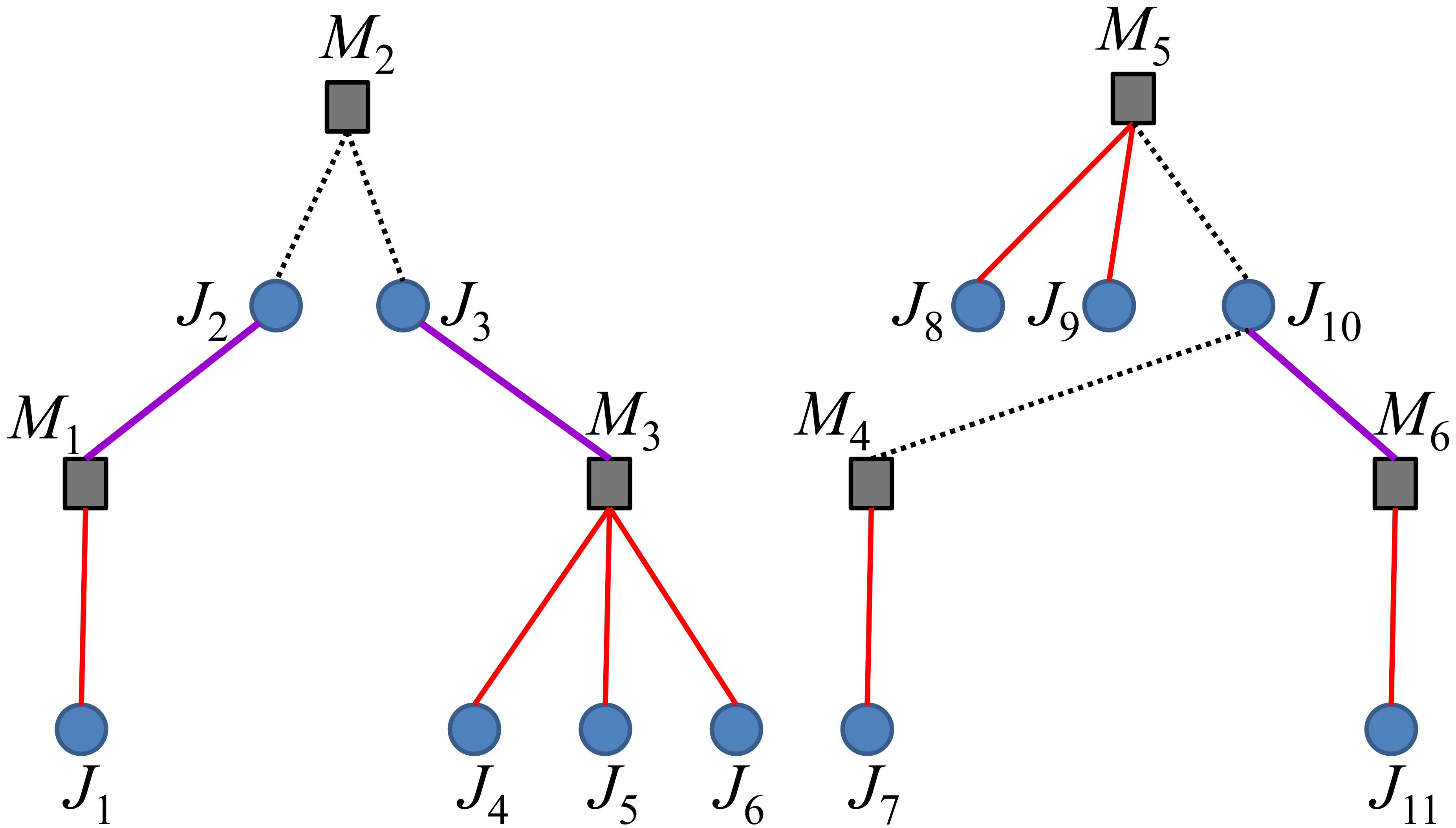

Feasible Solution




Obtained graph from the relaxation problem (no cycles)



$M_1$ $M_2$ $M_3$ $M_4$ $M_5$ $M_6$
$J_1$ $J_2$ $J_3$ $J_4$ $J_5$ $J_6$ $J_7$ $J_8$ $J_9$ $J_{10}$ $J_{11}$
Tree Structure
$M_2$
$J_2$ $J_3$
$M_1$ $M_3$
$J_1$ $J_4$ $J_5$ $J_6$
$M_5$
$J_8$ $J_9$ $J_{10}$
$M_4$ $M_6$
$J_7$ $J_{11}$



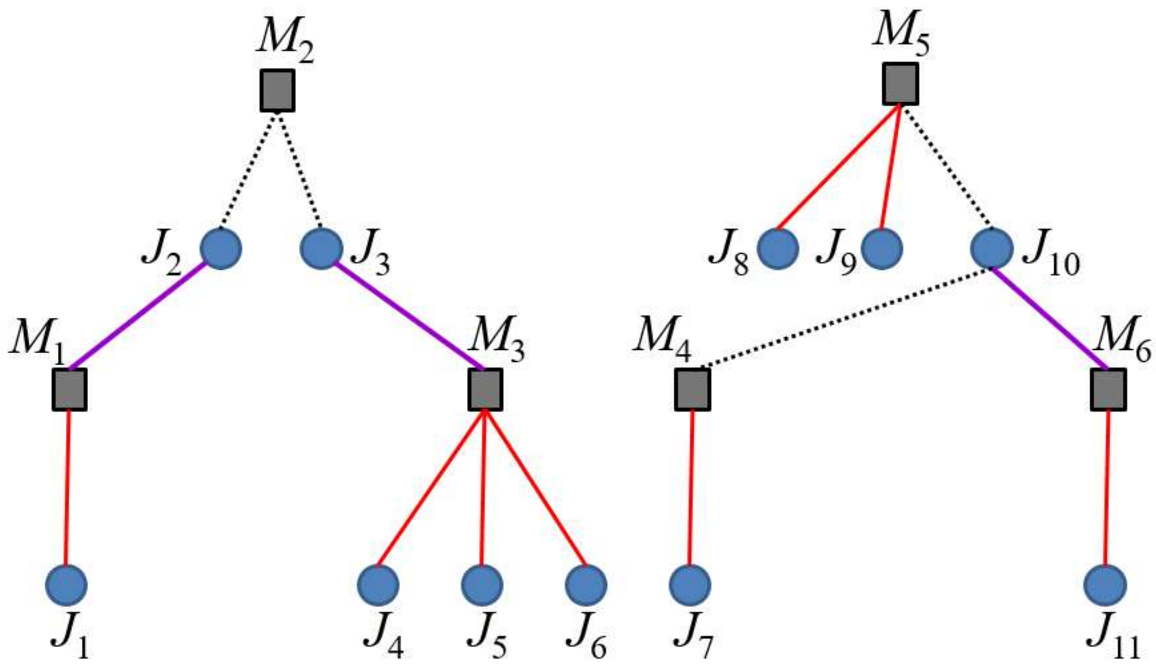

## Job Assignment

(1) Each leaf is assigned to its parent. (——)

(2) Each intermediate job node is assigned to an arbitrary child. (——)

# Upper Bound: $2L^*$

**2-approximation Algorithm**

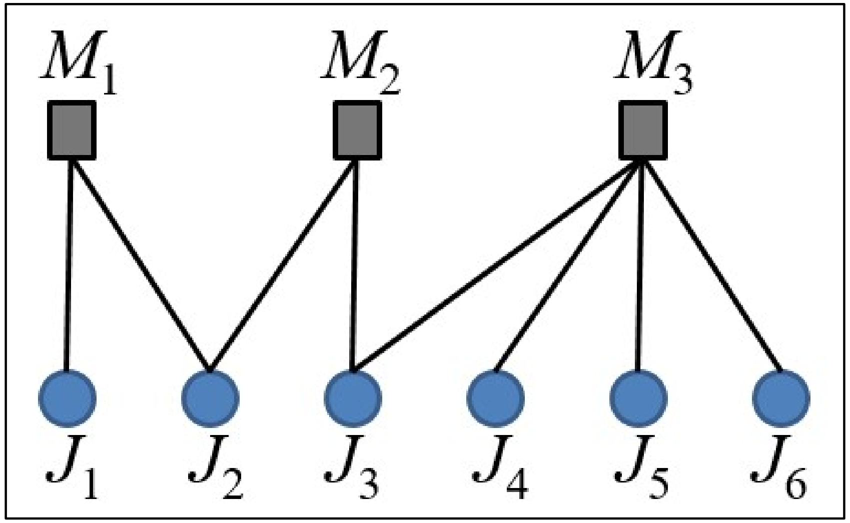



**(1) Leaf job assignment to its parent machine (——)**

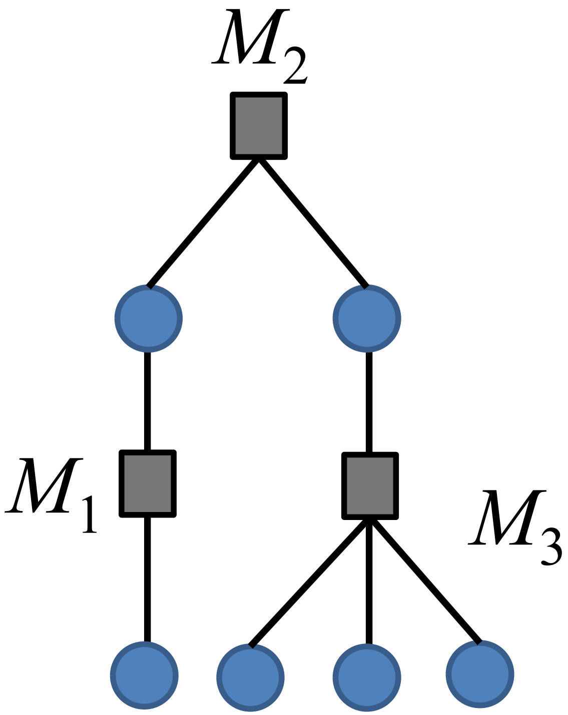
Since each job has a single edge, $x_{ij} = t_j$ holds for each job assignment. For those jobs assigned by this procedure, the following relation holds for each machine $i$: $\sum_j t_j = \sum_j x_{ij} \le L \le L^*$

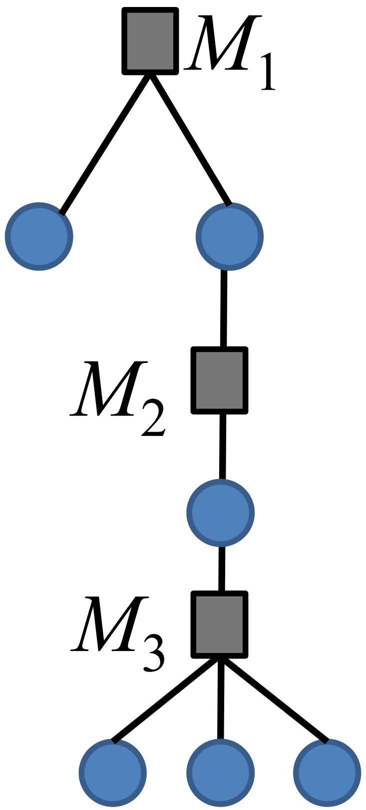
**(2) Intermediate job assignment to its child machine (—— )**

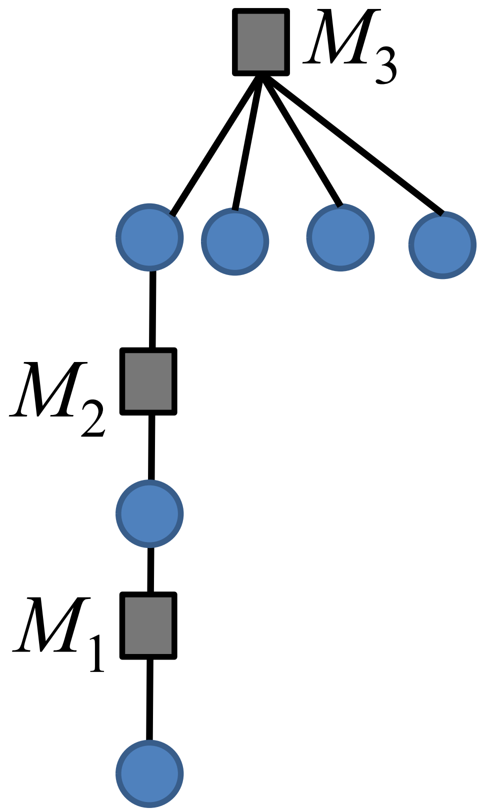
Since only a single job is assigned to some machines, the increase by this procedure is less than or equal to $\mathrm{Max}_j\, t_j \le L^*$



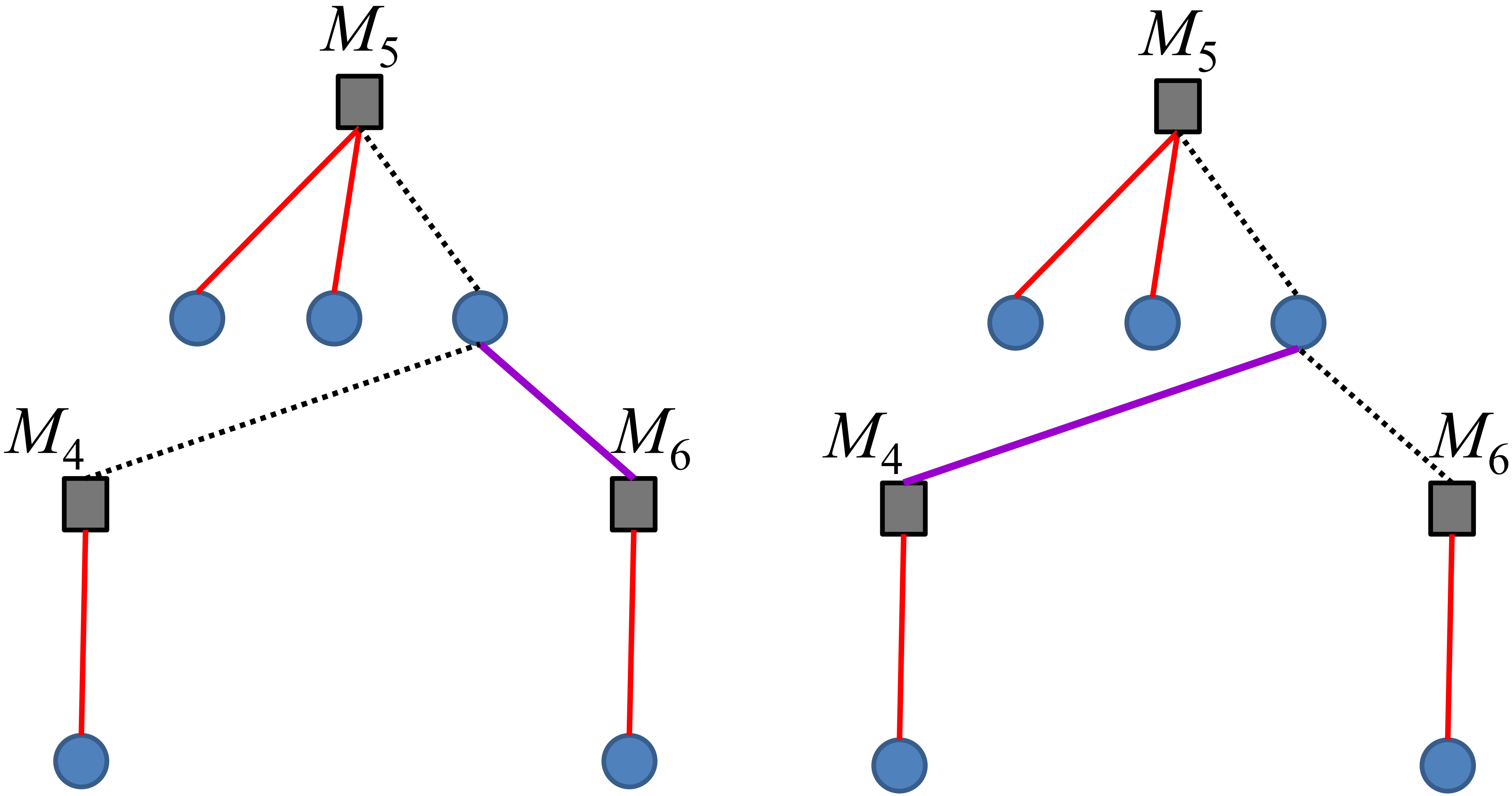

**Q1: Choice of a tree structure**
**Which is a good choice?**



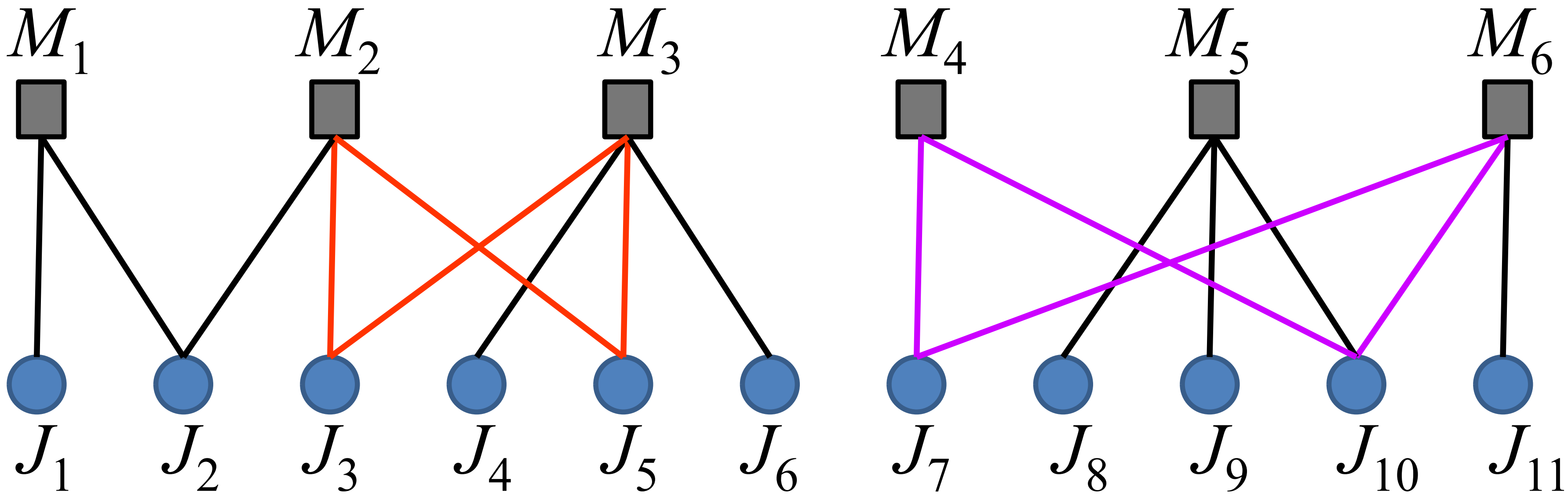



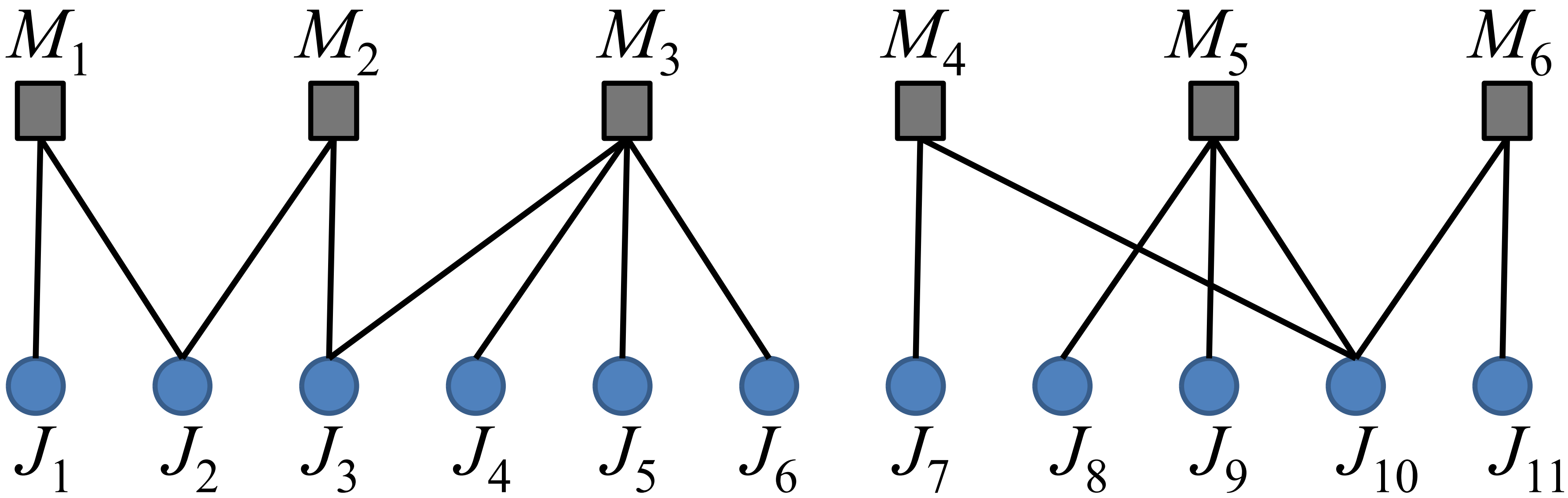



# Q2: Choice of a child node (which is better?)



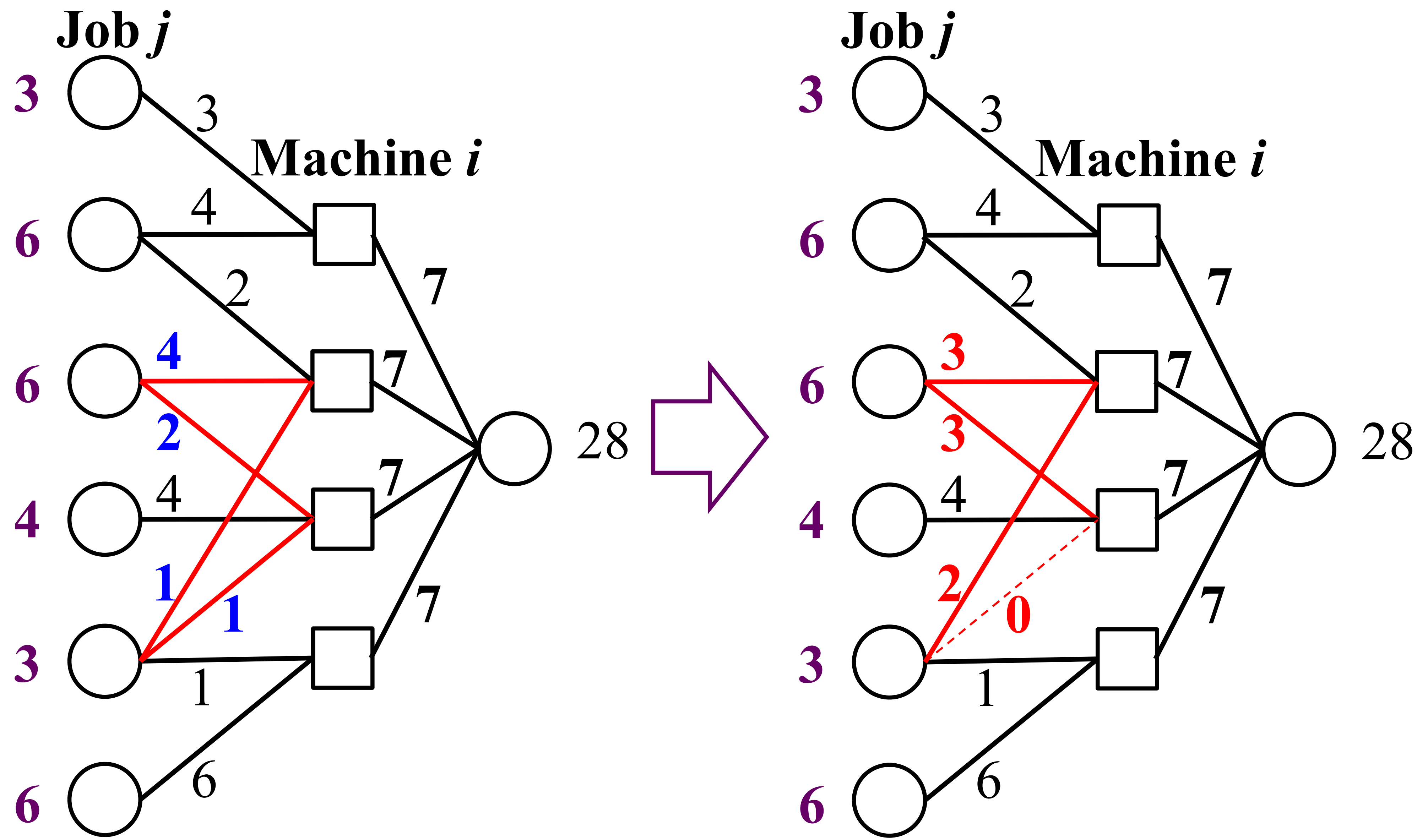
When the obtained graph from the solution of the relaxation problem has cycles, a graph with no cycles can be created without changing its objective value (makespan: $L$).



# How to eliminate cycles from the obtained graph?

Choose a cycle. Change the flow of each edge along the cycle without changing the total input to each node and the total output from each node (i.e., without changing the objective value *L*).



**Exercise 12-1**:

Create two examples of the generalized load balancing problem where a graph with no cycle is obtained from one example and a graph with a cycle (or two cycles) is obtained from the other example. Then explain the entire procedure of the LP-based algorithm using the created examples.

## Exercise 12-1: Use of LP

Create two examples of the generalized load balancing problem where a graph with no cycle is obtained from one example and a graph with a cycle (or two cycles) is obtained from the other example. Then explain the entire procedure of the LP-based algorithm using the created examples.

**One example:** Create an example. Apply **an LP package** to the created example. Generate a graph using the obtained LP solution. The generated graph should have no cycle.

**The other example:** Create an example. Apply **an LP package** to the created example. Generate a graph using the obtained LP solution. The graph should have one or more cycles.

## Exercise 12-1: Use of LP

Create two examples of the generalized load balancing problem where a graph with no cycle is obtained from one example and a graph with a cycle (or two cycles) is obtained from the other example. Then explain the entire procedure of the LP-based algorithm using the created examples.

**One example:** Create an example. Apply **an LP package** to the created example. Generate a graph using the obtained LP solution. The generated graph should have no cycle.

**The other example:** Create an example. Apply **an LP package** to the created example. Generate a graph using the obtained LP solution. The graph should have one or more cycles.

## LP Problem:

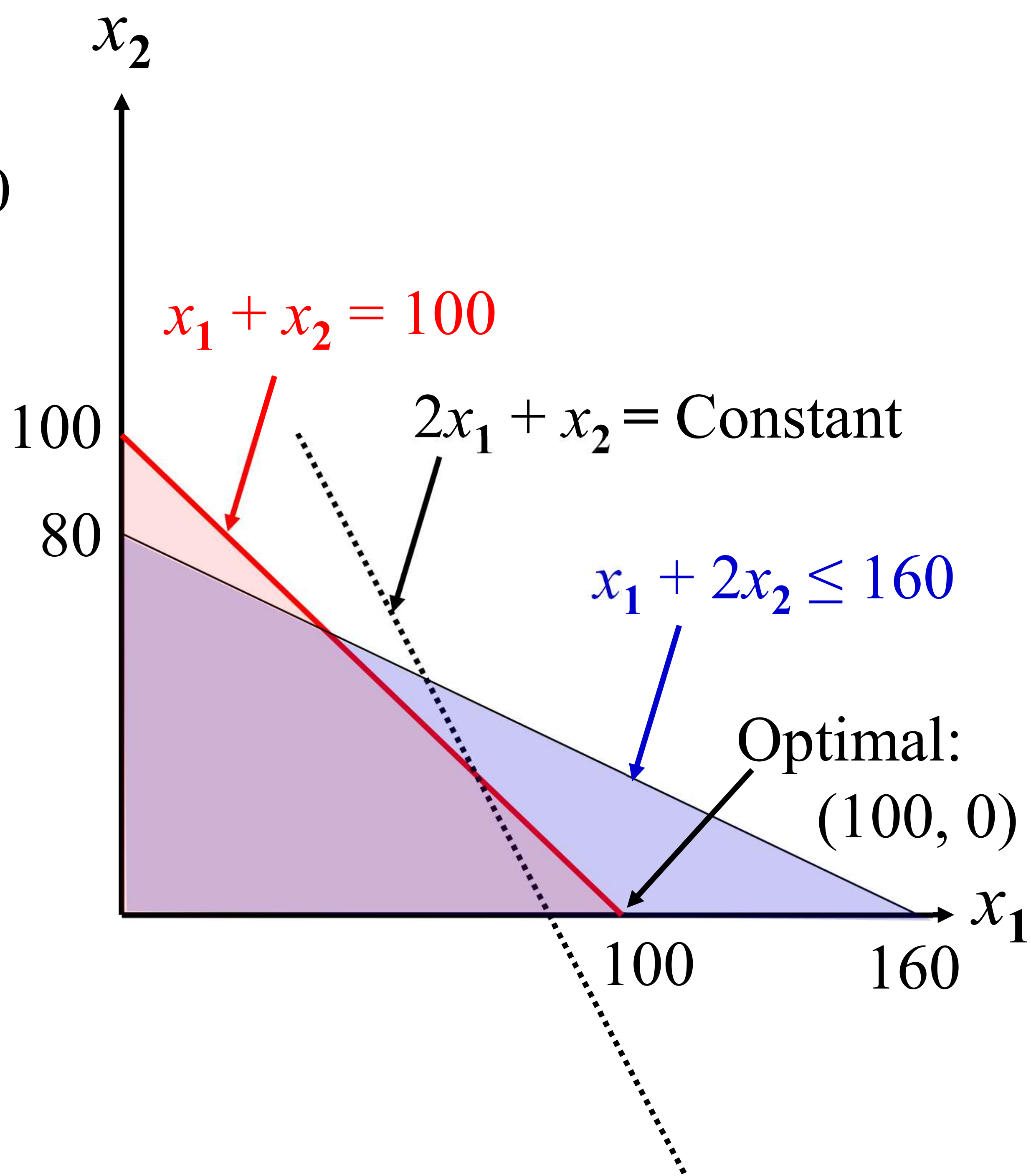
Maximize $2x_1 + x_2$

Subject to $x_1 + x_2 = 100$

$x_1 + 2x_2 \leq 160$

$x_1 \geq 0$

$x_2 \geq 0$



## LP Problem:

Minimize $L$

Subject to $x_1 + x_2 = 100$

$$x_1 + 2x_2 \le L$$

$$2x_1 + x_2 \le L$$

$$x_1 \ge 0$$

$$x_2 \ge 0$$

**LP Problem:**

Minimize $L$

Subject to
$$x_1 + x_2 = 100$$
$$x_1 + 2x_2 \le L$$
$$2x_1 + x_2 \le L$$
$$x_1 \ge 0$$
$$x_2 \ge 0$$

**LP Problem:**

Minimize $x_3$

Subject to
$$x_1 + x_2 = 100$$
$$x_1 + 2x_2 - x_3 \le 0$$
$$2x_1 + x_2 - x_3 \le 0$$
$$x_1 \ge 0$$
$$x_2 \ge 0$$

**LP Problem:**

Minimize $L$

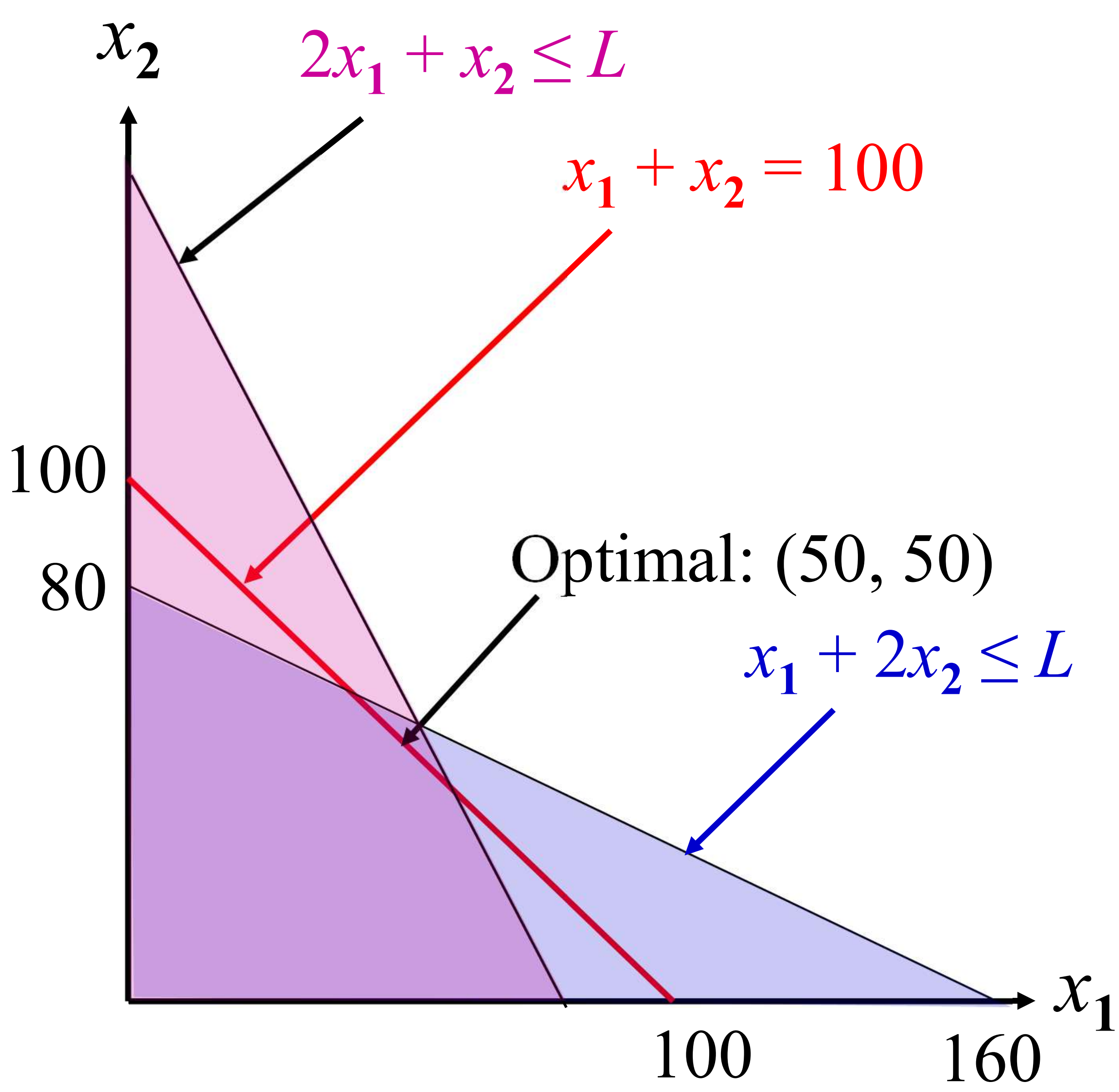
Subject to $x_1 + x_2 = 100$

$x_1 + 2x_2 \leq L$

$2x_1 + x_2 \leq L$

$x_1 \geq 0$

$x_2 \geq 0$



**LP Problem:**

Minimize $L$

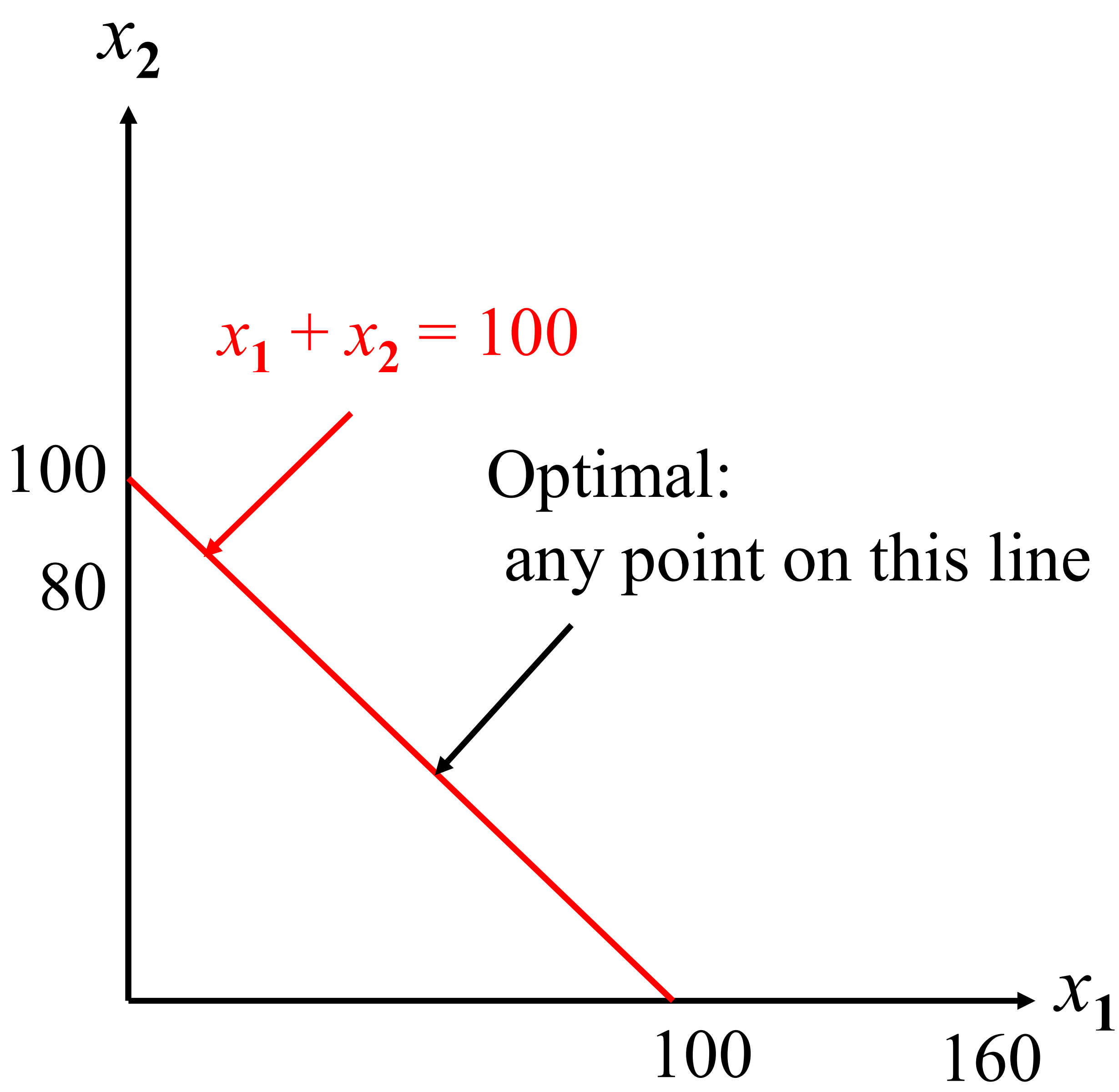
Subject to $x_1 + x_2 = 100$

$2x_1 + 2x_2 \le L$

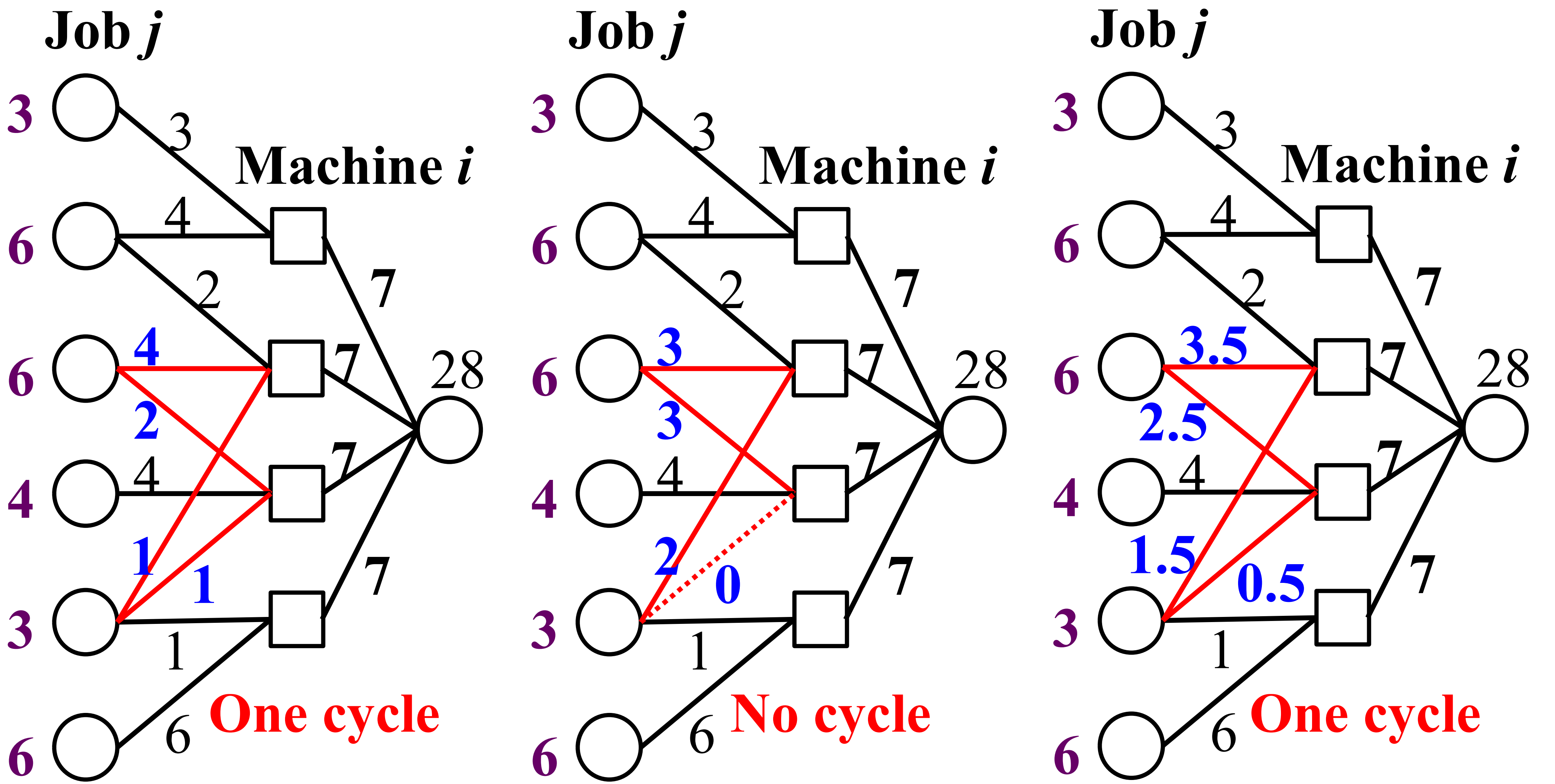
$2x_1 + 2x_2 \le L$

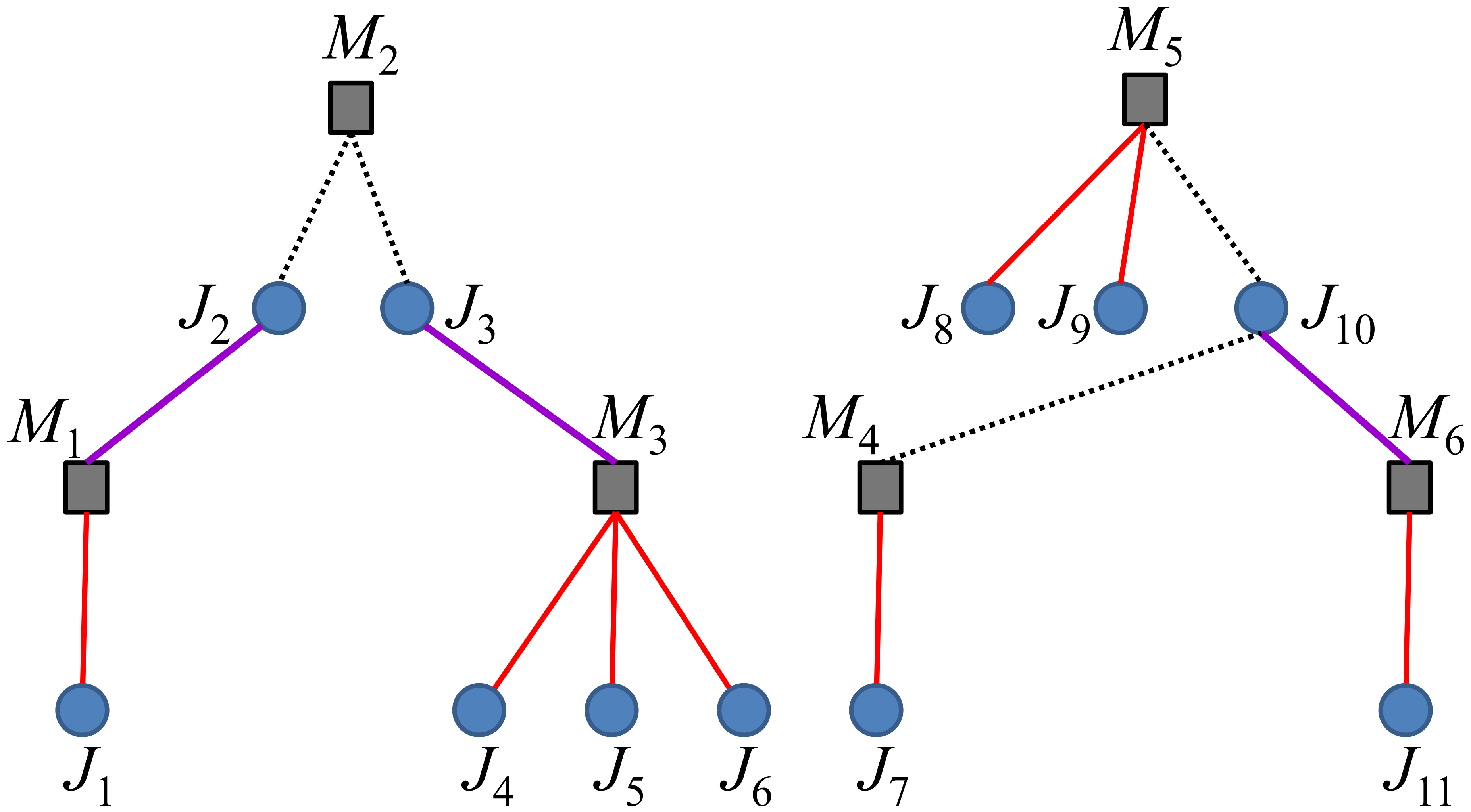
$x_1 \ge 0$

$x_2 \ge 0$



The following are examples of the optimal solutions of the LP formulation. In Exercise 11-1, a graph should be generated using the LP solution (which should be actually obtained from your LP package). That is, the point of Exercise 11-1 is to examine which graph is actually obtained by your LP package.



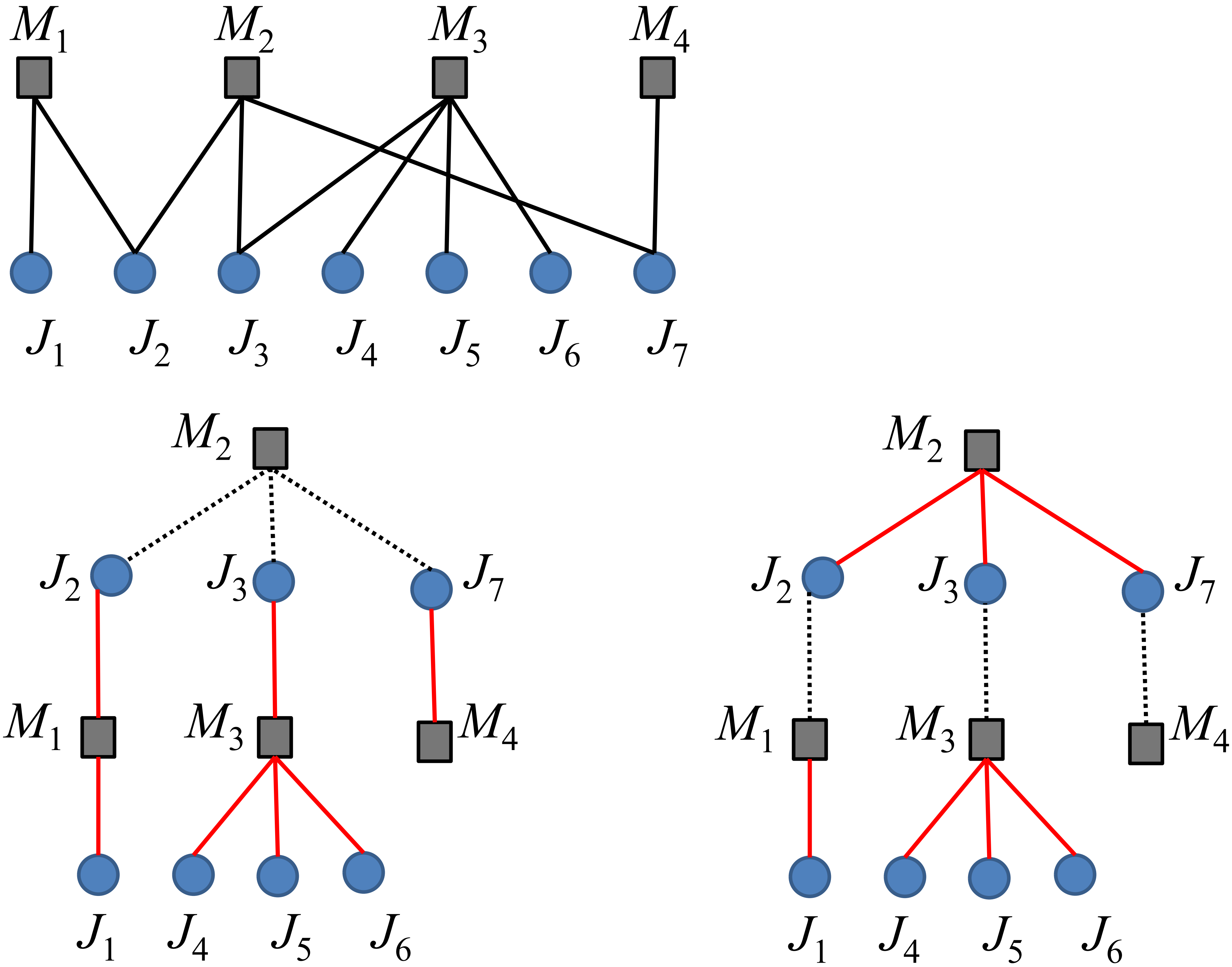



**Job Assignment**

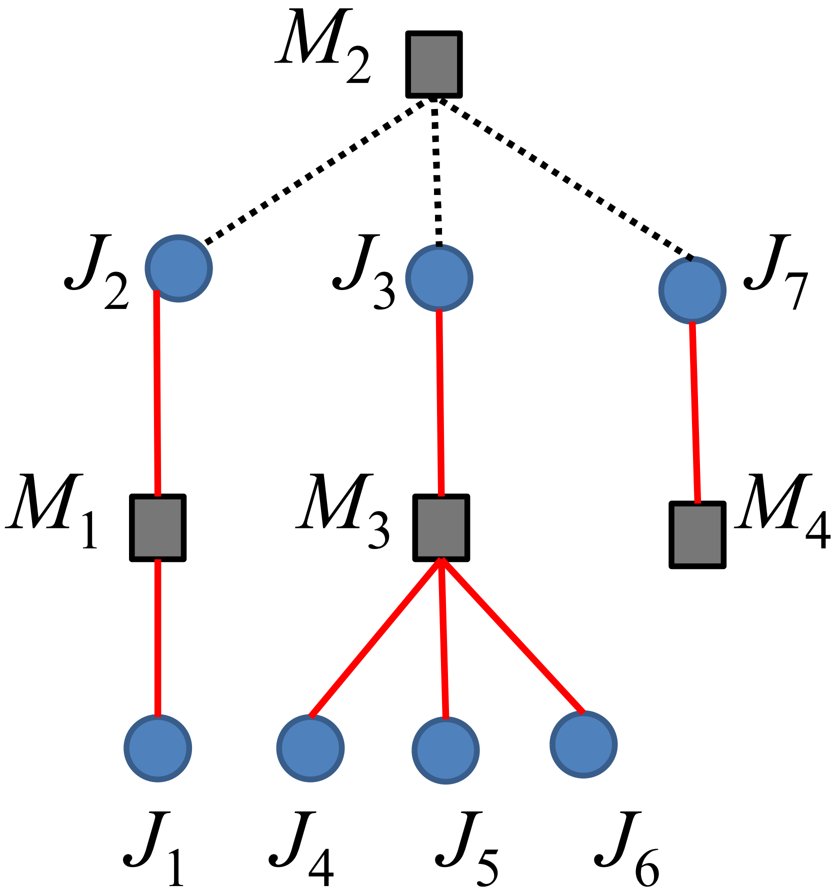
(1) Each leaf is assigned to its parent. (—)

**Why ?**

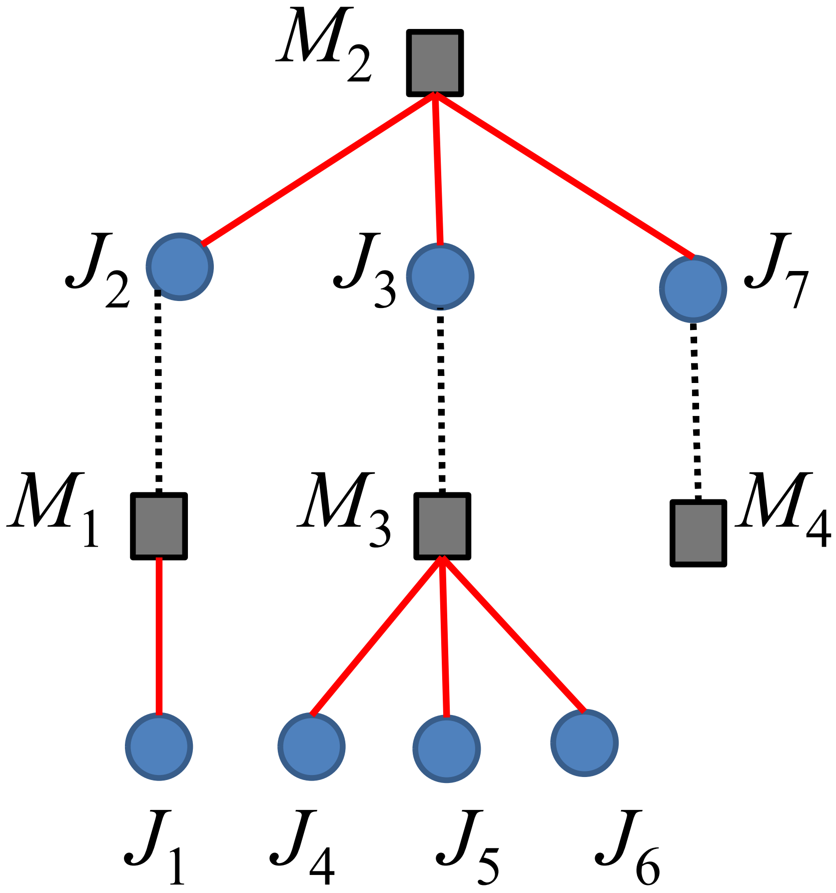
(2) Each intermediate job node is assigned to an arbitrary child.
(—)



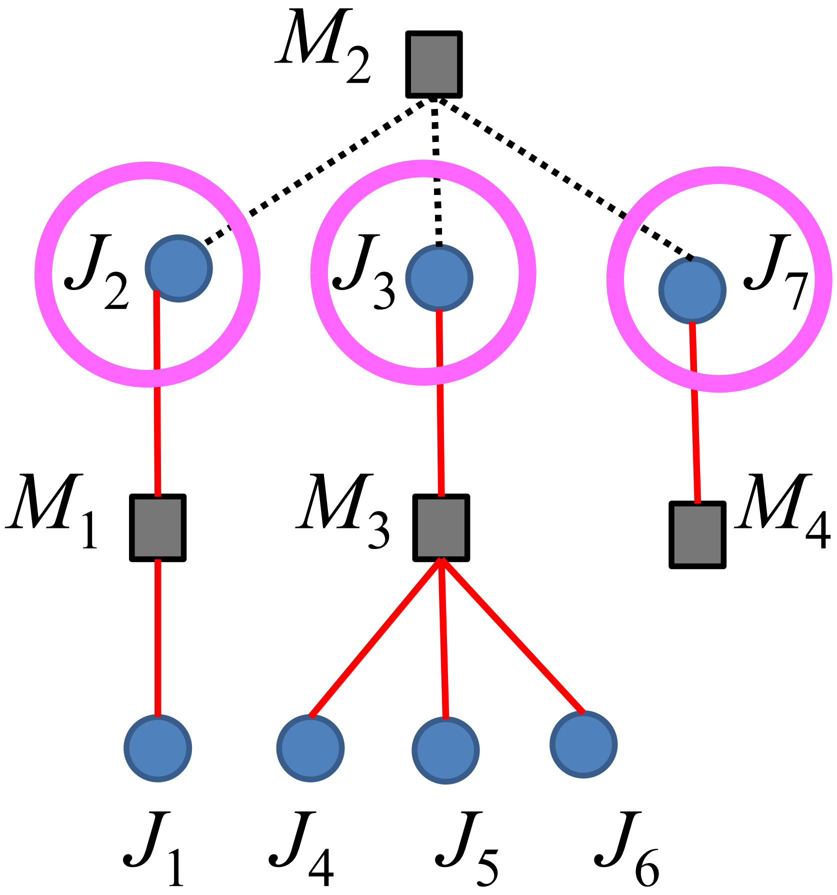
$M_1$
$M_2$
$M_3$
$M_4$
$J_1$
$J_2$
$J_3$
$J_4$
$J_5$
$J_6$
$J_7$
$M_2$
$J_2$
$J_3$
$J_7$
$M_1$
$M_3$
$M_4$
$J_1$
$J_4$
$J_5$
$J_6$
$M_2$
$J_2$
$J_3$
$J_7$
$M_1$
$M_3$
$M_4$
$J_1$
$J_4$
$J_5$
$J_6$



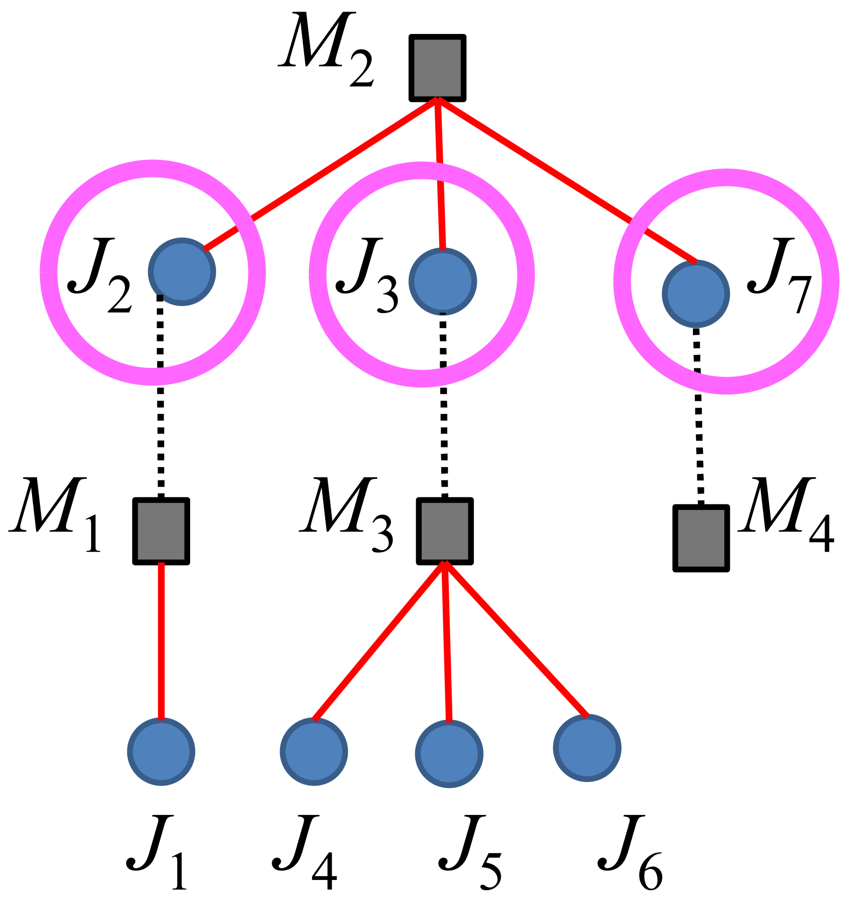

**2-Approximation**




**Not 2-Approximation**



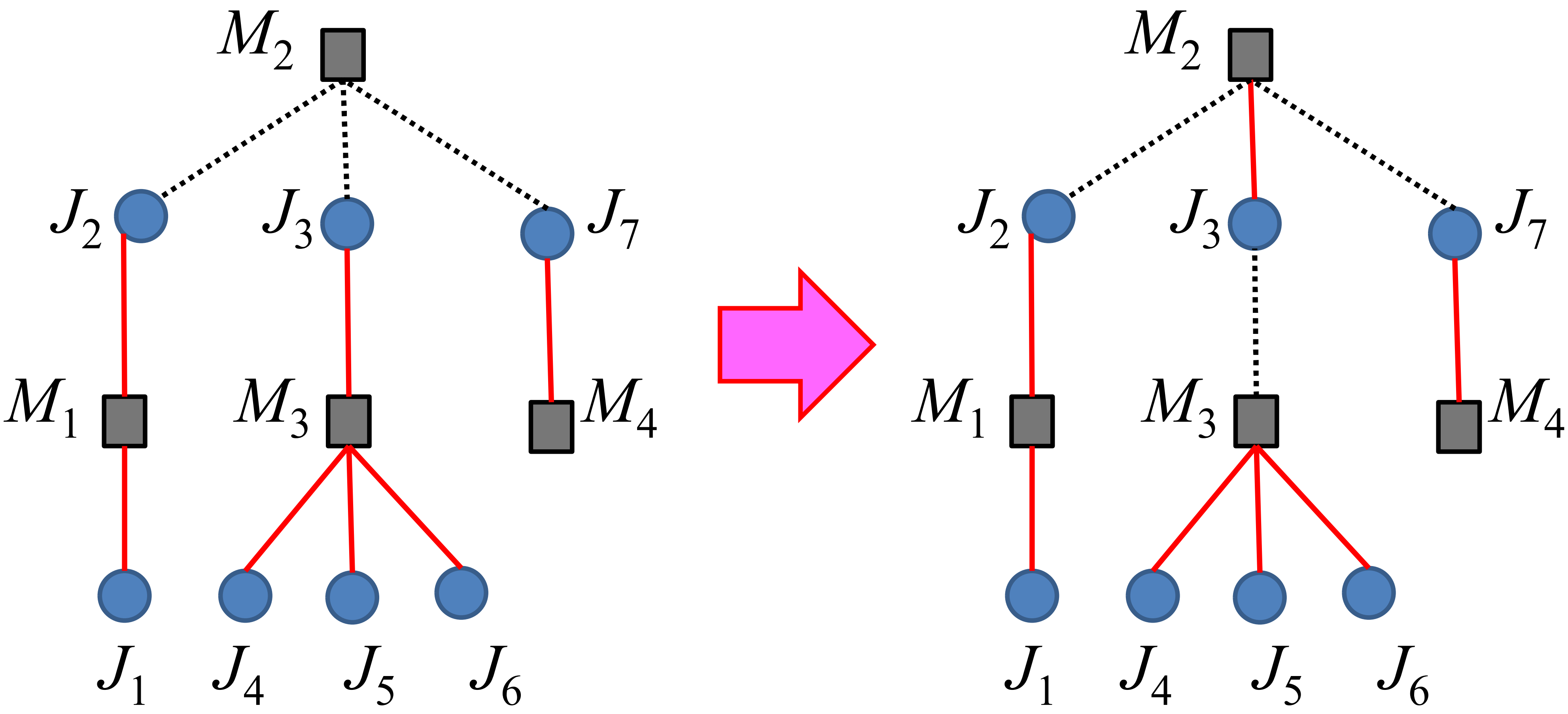

**2-Approximation**




**Not 2-Approximation**

When $t_2$, $t_3$, $t_7$ are very large and $t_1$, $t_4$, $t_5$, $t_6$ are very small, the optimal value $L^*$ is close to $\max\{t_2, t_3, t_7\}$. In the right figure, $L = t_2 + t_3 + t_7$ can be larger than $2L^*$ (which can be close to $3L^*$).

If the parent node has no job, it may be a good idea to assign one job to the parent.




**2-Approximation** **2-Approximation**