# CIVIL-557
# Decision-Aid Methodologies in Transportation
## Lecture III
## Shortest Path Problems

Fabian Torres

Transport and Mobility Laboratory TRANSP-OR
École Polytechnique Fédérale de Lausanne EPFL

EPFL

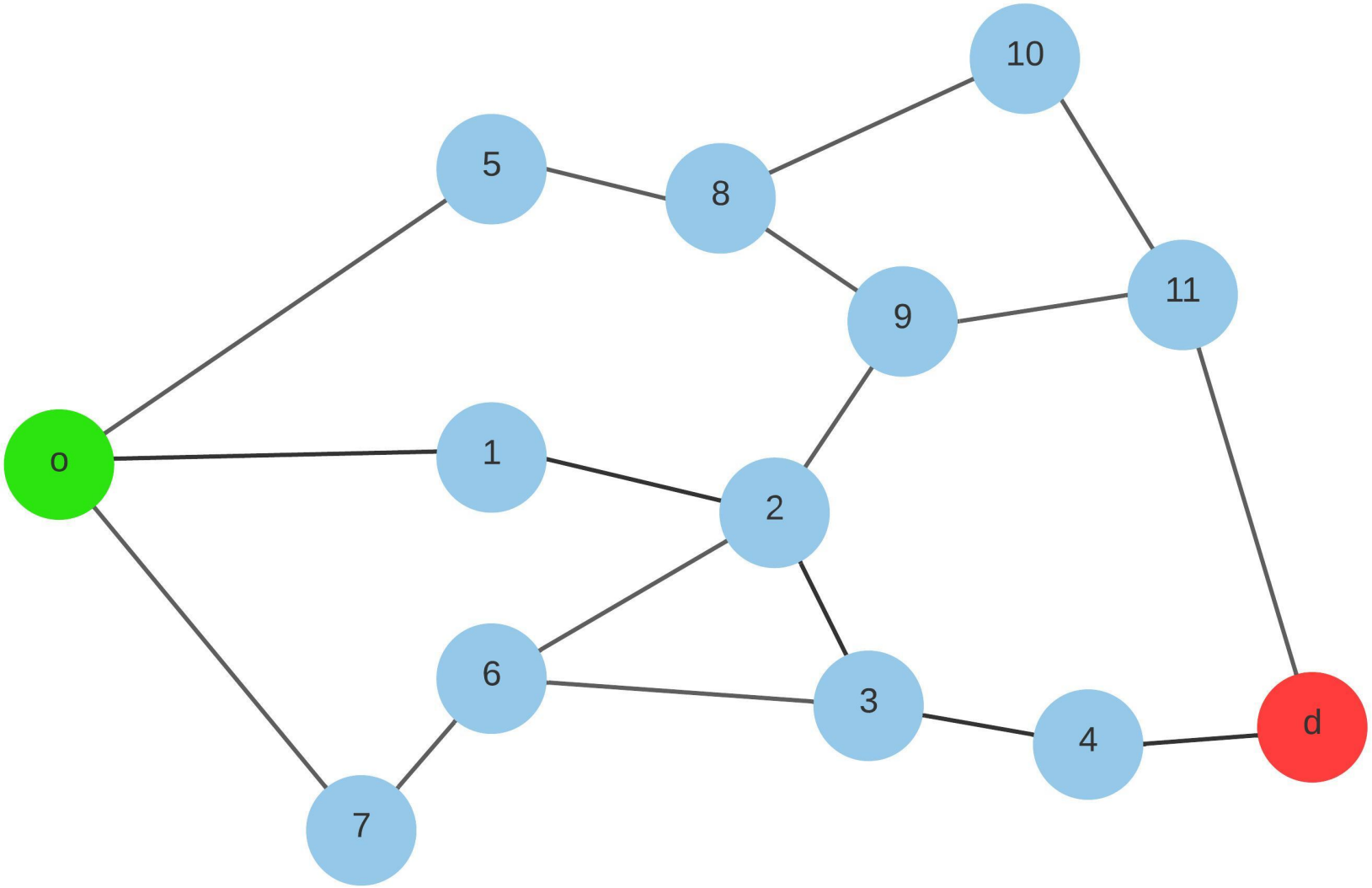
# Outline

# Outline

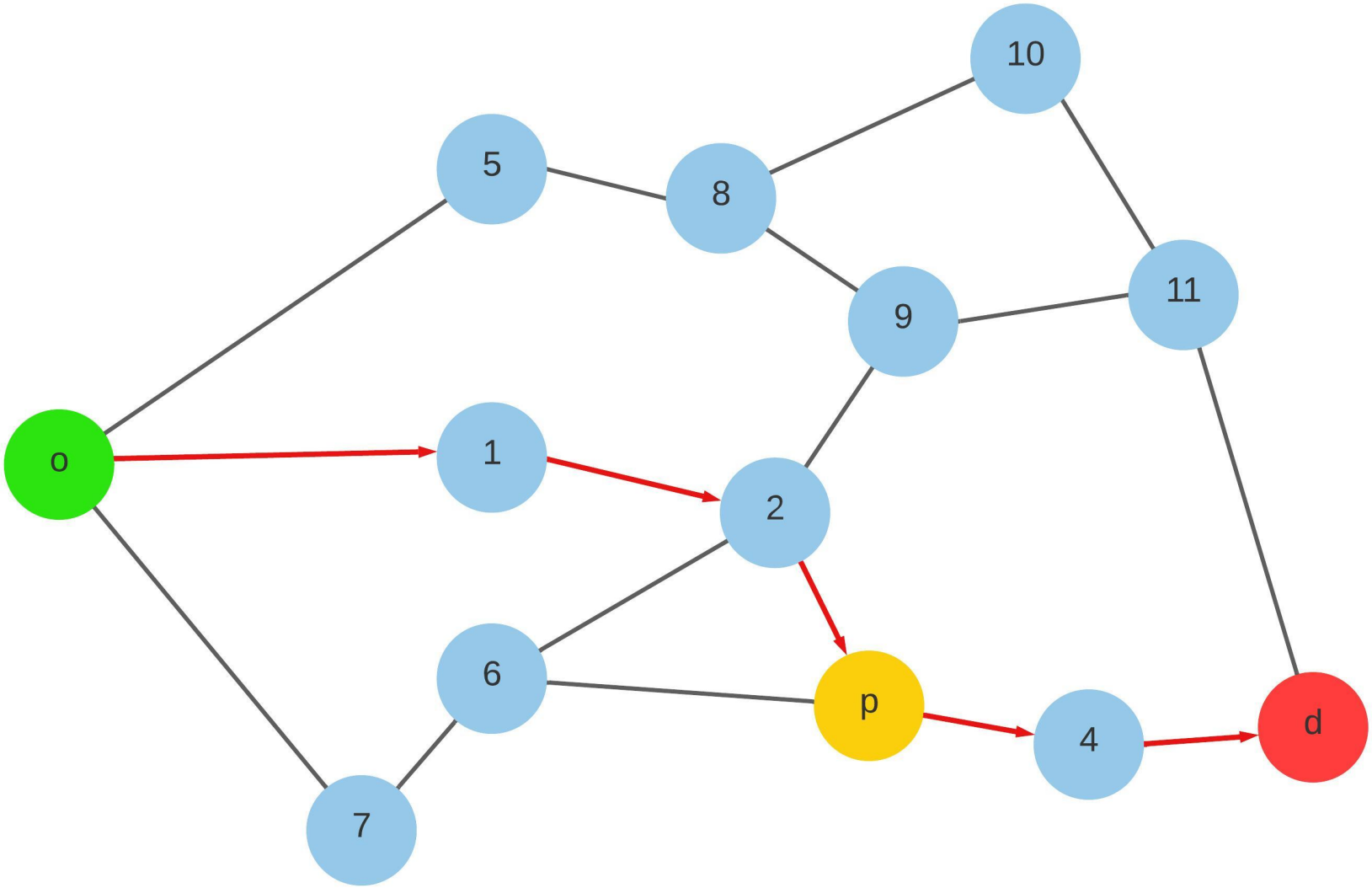


EPFL

# Shortest Path Problem (SPP)



# SPP: Principle of optimality



# SPP: Principle of optimality

Let $P = \{o, \ldots, p_1, \ldots, p_2, \ldots d\}$ be the shortest path that starts in the origin $o$ and finishes in the destination $d$. And let $p_1$ and $p_2$ be any intermediate nodes in the path $(o - d)$.

**Proposition:**

The subpaths $(o - p_1), (p_1 - p_2)$ and $(p_2 - d)$ are shortest paths from ($o$ to $p_1$), ($p_1$ to $p_2$) and ($p_2$ to $d$) respectively.

**Proof:**

Suppose there was a shorter path that could replace any of these subpaths. Then, we could replace this smaller cost section into path P and we would find a shorter path from ($o$ to $d$). Which contradicts that P was the shortest path since we found a shorter path than P. □

EPFL

# SPP: Principle of optimality

If we knew the shortest path to every node connected to $p$, we could extend these paths to node $p$ and determine which is the shortest at node $p$:

- $D(i)$ is the shortest path distance from the origin to $i$.
- $N(p)$ is the set of nodes connected to the node $(p)$.
- $c_{ip}$ is the cost of going from node $i$ to $p$ directly.

$$D(p) = \min_{i \in N(p)} \{D(i) + c_{ip}\}$$

For acyclic directed graphs, this leads to an algorithm that is polynomial time $O(|A|)$ where $A$ is the set of arcs.

EPFL

# SPP: Principle of optimality

For a general directed graph with non-negative costs we need an ordering: **Where do we begin?**

Consider a set of stages $k \in \{0, 1, \ldots, |N|\}$, where k is the number of nodes visited in a path.

$$D_k(j) = \min_{i \in N(j)} \{D_{k-1}(i) + c_{ij}\}$$

We start by setting $D_0(o) = 0$ and then find the shortest paths for all nodes connected to the origin.

**For all** $k \in 1, \ldots, |N|$ :

**For all** $j \in N$ :

$$D_k(j) = min_{i \in N(j)}\{D_{k-1}(i) + c_{ij,\infty}\}$$

EPFL

# Label


Brand Information
Legal Description
Product Description
Marketing Communication
Coca-Cola
20 FL OZ (1.25 PT) 591 mL
240 CALORIES PER BOTTLE
Nutrition Facts
Amount/Serving
Total Fat 0g
Sodium 75mg
Total Carb. 65g
Protein 0g
Calories 240
please recycle
NO REFILL
Share a Coke with
Danielle
#ShareaCoke
mycoke rewards .com
0 490440 3
Company Information
Identification Marks

# Label

- A label stores information about a path. Any information can be stored, e.g., cost, time, capacity, etc.
- Let $\mathcal{L}_k = (i, D_k(i), L_{k-1})$ be a label with information about the current node (i.e., $i$), the cost of the path to node $i$ (i.e., $D_k(i)$) and the previous label from the path (i.e., $L_{k-1}$).
- Let $\mathcal{M}(k, i)$ be a bucket that stores labels that are currently in node $i$ after visiting $k$ nodes.

**For simplicity:** $\mathcal{M}(k, i) = \{\mathcal{L} = (i, c, L)\}$, where $c = D_k(i)$ and $L = L_{k-1}$

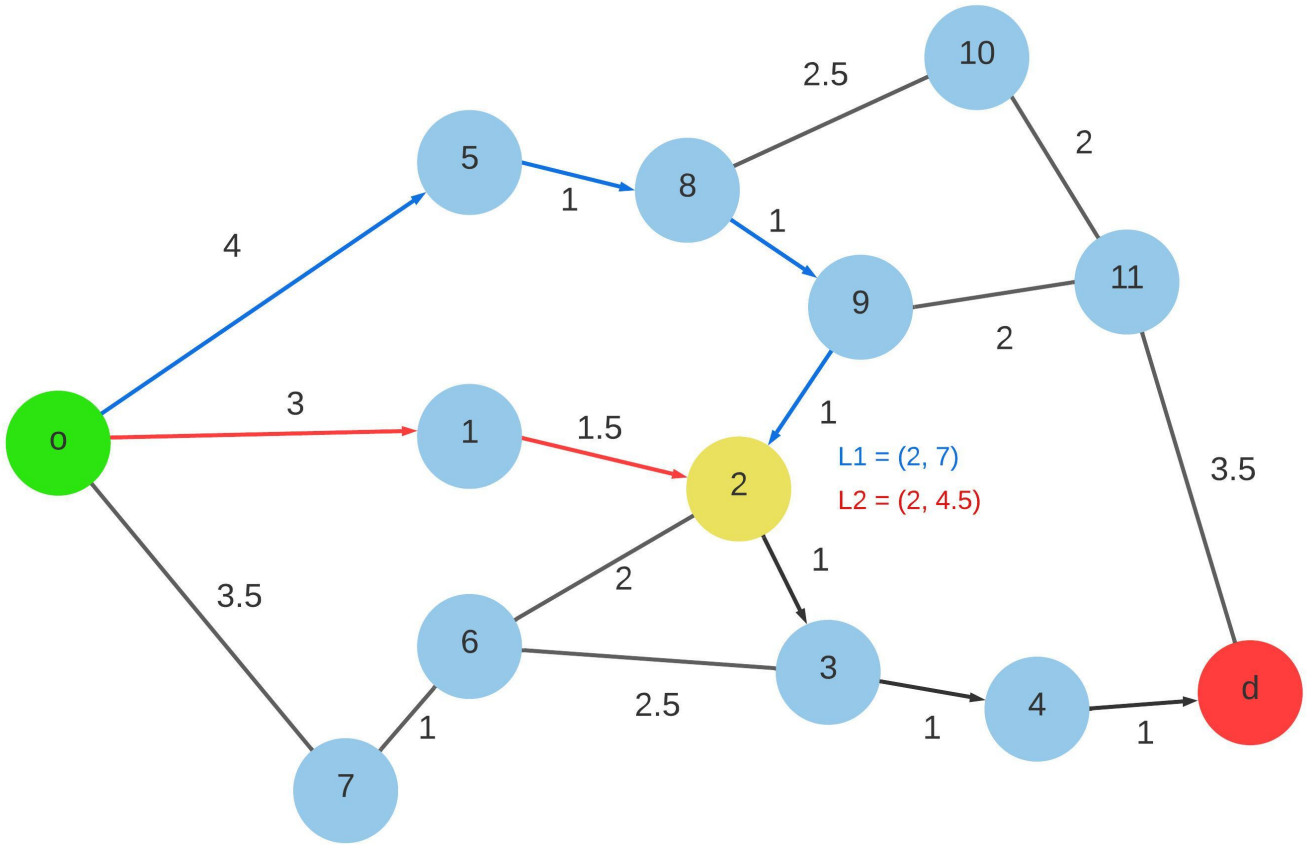
EPFL

# Label extension

- Extending a label $\mathcal{L}_{extend} = (i^e, c^e, L)$ is to extend the path from the current node, i.e., $i^e$ to another adjacent node in the graph, e.g., $j^n$.
- Extending a label creates a new label $\mathcal{L}_{new}$.
- The label is created like this $\mathcal{L}_{new} = (j^n, c^e + c_{i^e j^n}, \mathcal{L}_{extend})$.

EPFL

# Dominance



# Dominance

If two labels are in the same node we can use **the Principle of optimality** to decide which is better and eliminate the one that has a larger cost.

**Dominance rules:**
Label $\mathcal{L}_1 = (i_1, c_1, L_1)$ dominates label $\mathcal{L}_2 = (i_2, c_2, L_2)$ if the following is true:

1. $i_1 = i_2$
2. $c_1 \leq c_2$

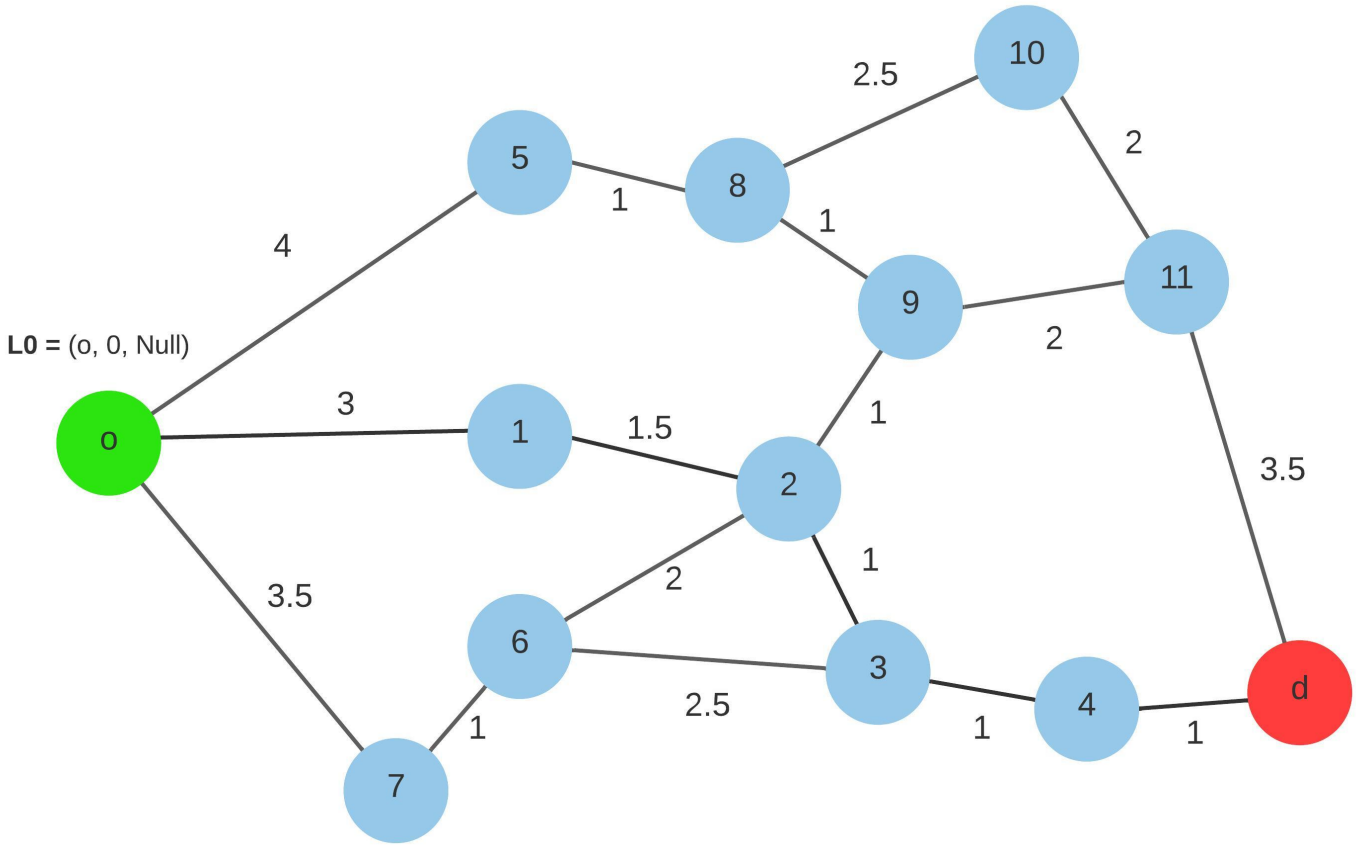
EPFL

# Labeling Algorithm

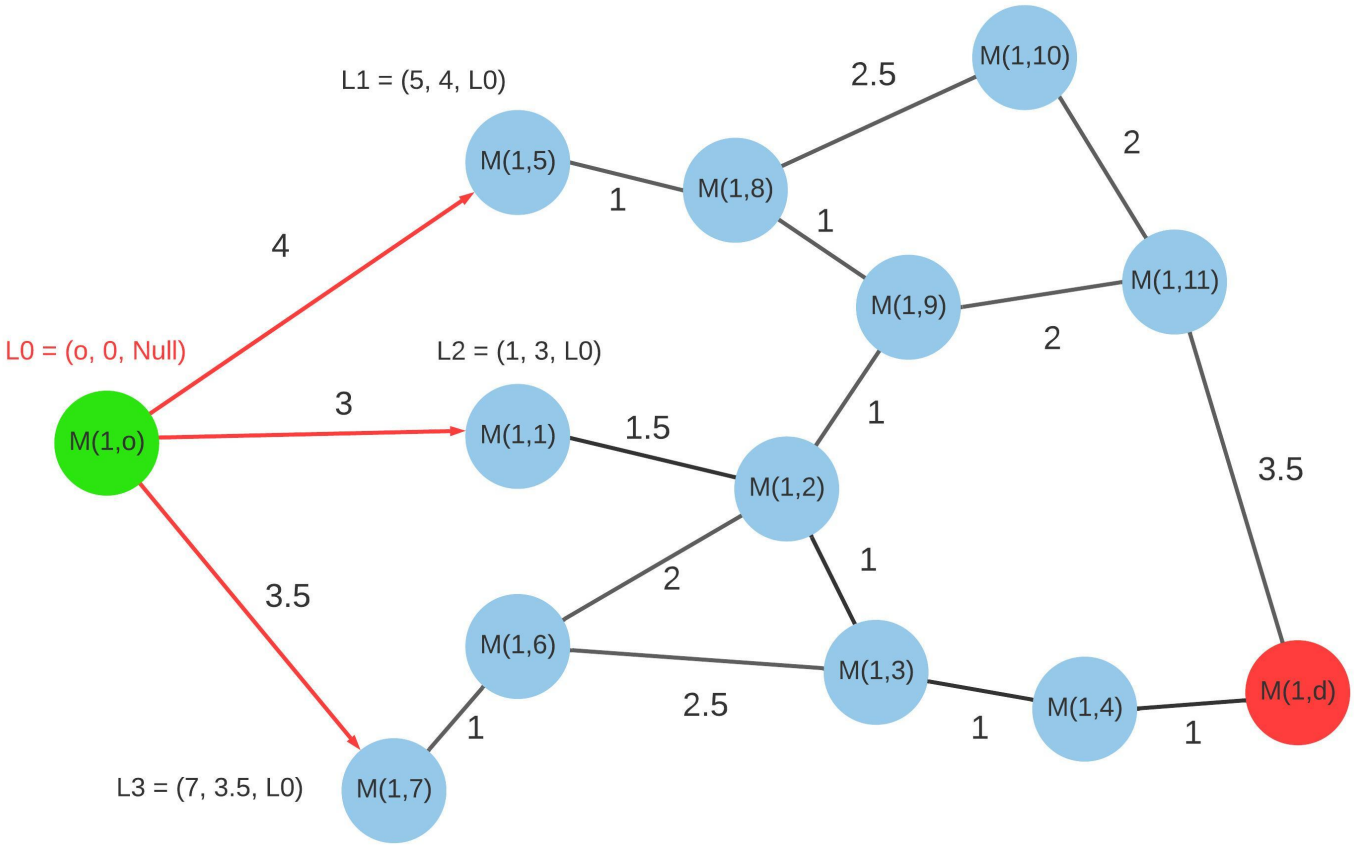
**Output:** Shortest path

$\mathcal{M}(k,i) \leftarrow \emptyset, \quad \forall i \in N, k \in \{0,\ldots,|N|\}$;
$\mathcal{M}(0,o) \leftarrow \{(o,0,\emptyset)\}$;
**for** $k = 1$ **to** $|N|$ **do**
  **for** $i \in N$ **do**
    **for** $j \in N(i)$ **do**
      **for** $l \in \mathcal{M}(k-1,i)$ **do**
        **if** *feasible(i,j)* **then**
          Create new label:
          $g \leftarrow (j, f_c, l)$;
          *dominated* $\leftarrow$ *false*;
          *Dominance* $\leftarrow$ rules;
          **for** $h \in \mathcal{M}(k,j)$ **do**
            Check dominance:
            **if** *Dominance*$(h,g)$ **then**
              *dominated* $\leftarrow$ *true*;
              **Break**;
            **else if** *Dominance*$(g,h)$ **then**
              Delete dominated label h:
              $\mathcal{M}(k,j) \setminus \{h\}$;
            **end**
          **end**
          **if** *not dominated* **then**
            $\mathcal{M}(k,j) \leftarrow \mathcal{M}(k,j) \cup \{g\}$;
          **end**
        **end**
      **end**
    **end**
  **end**
**end**
**return** Cheapest label from $\mathcal{M}$

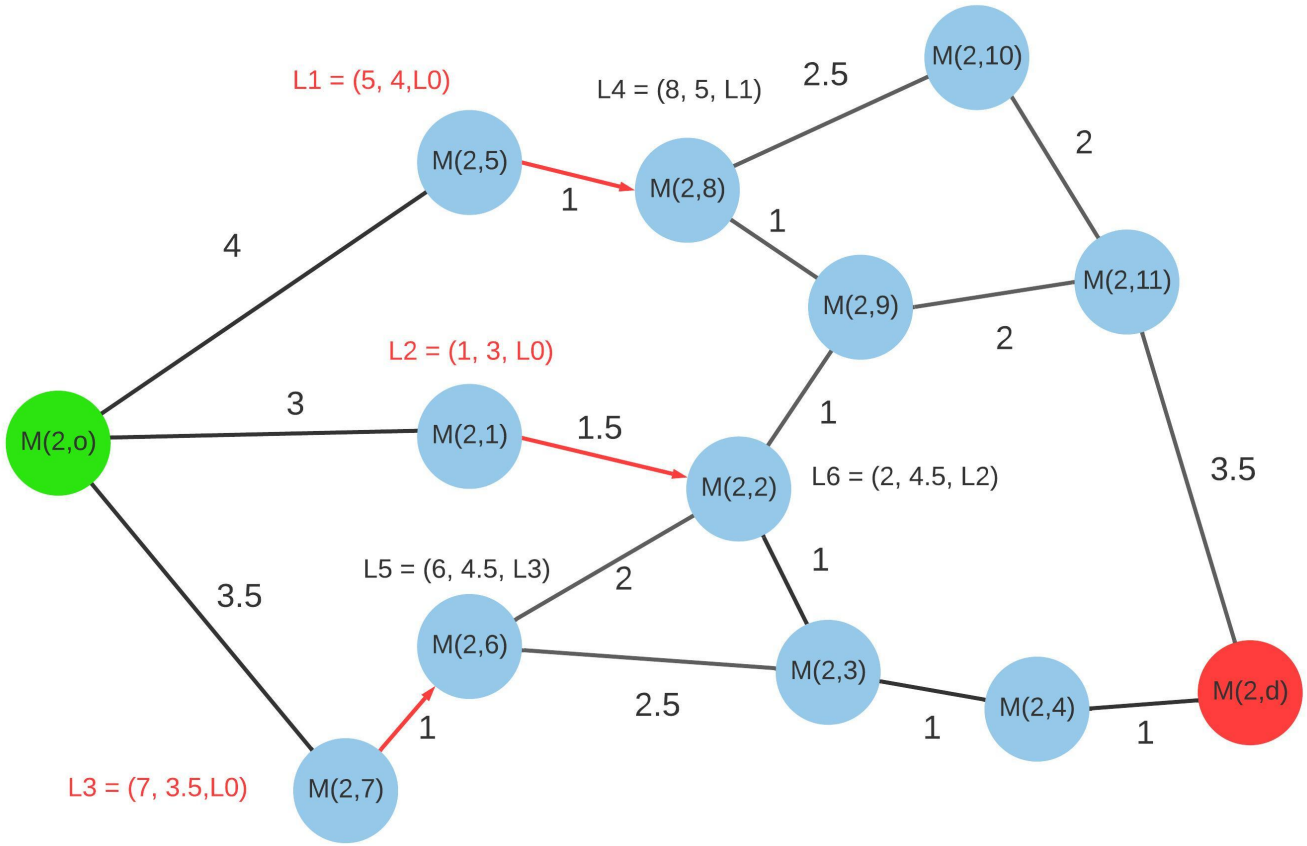
# Example: stage 0 (k = 0)



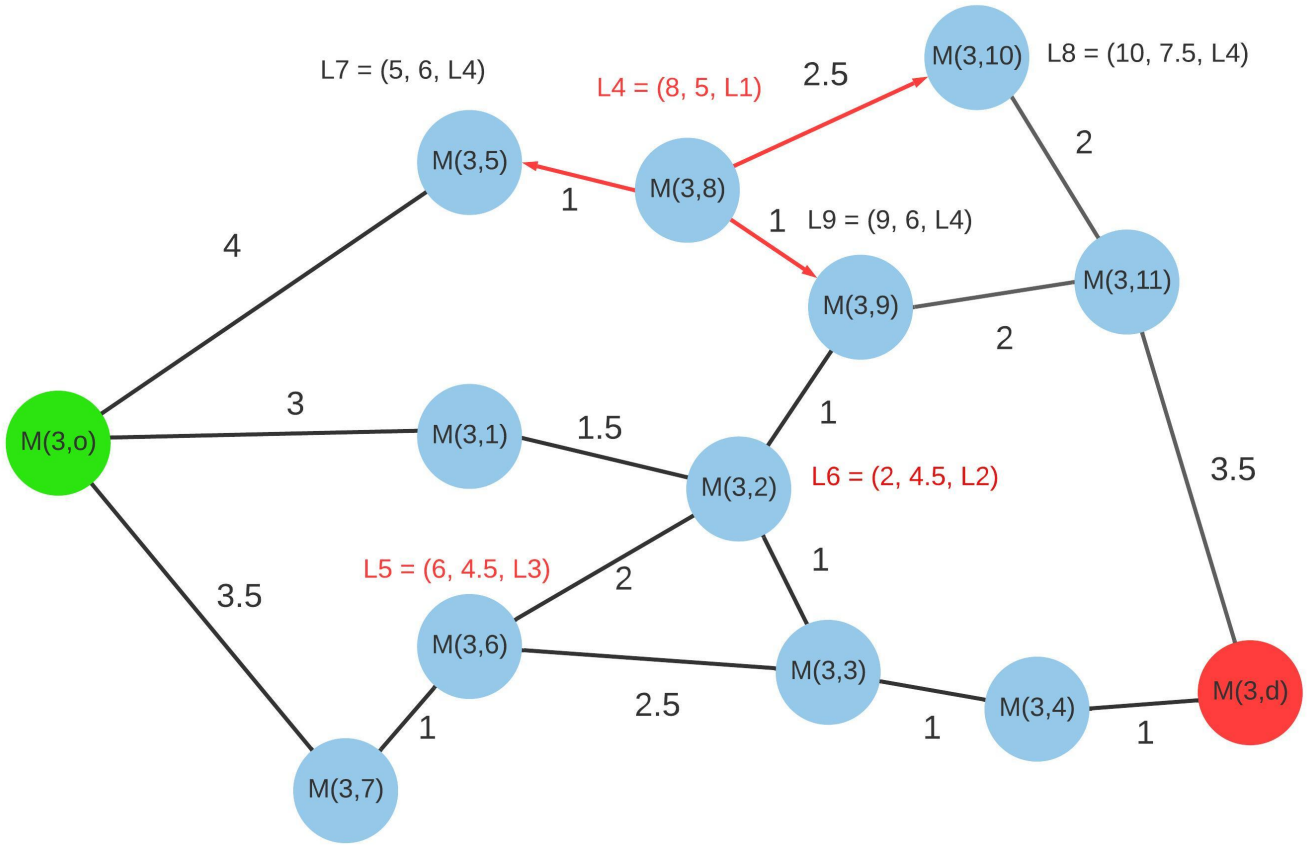
# Example: stage 1 (k = 1)



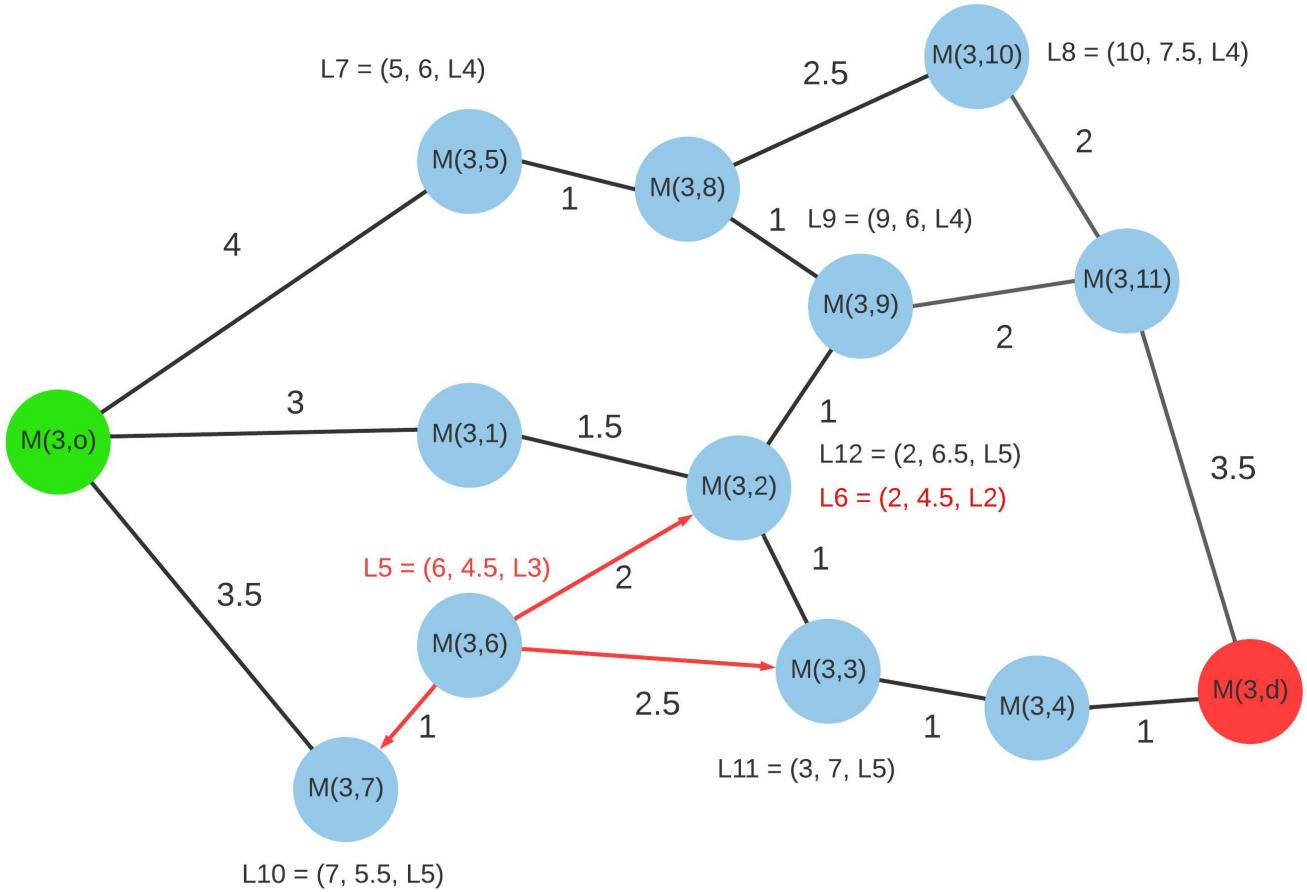
# Example: stage 2 (k = 2)



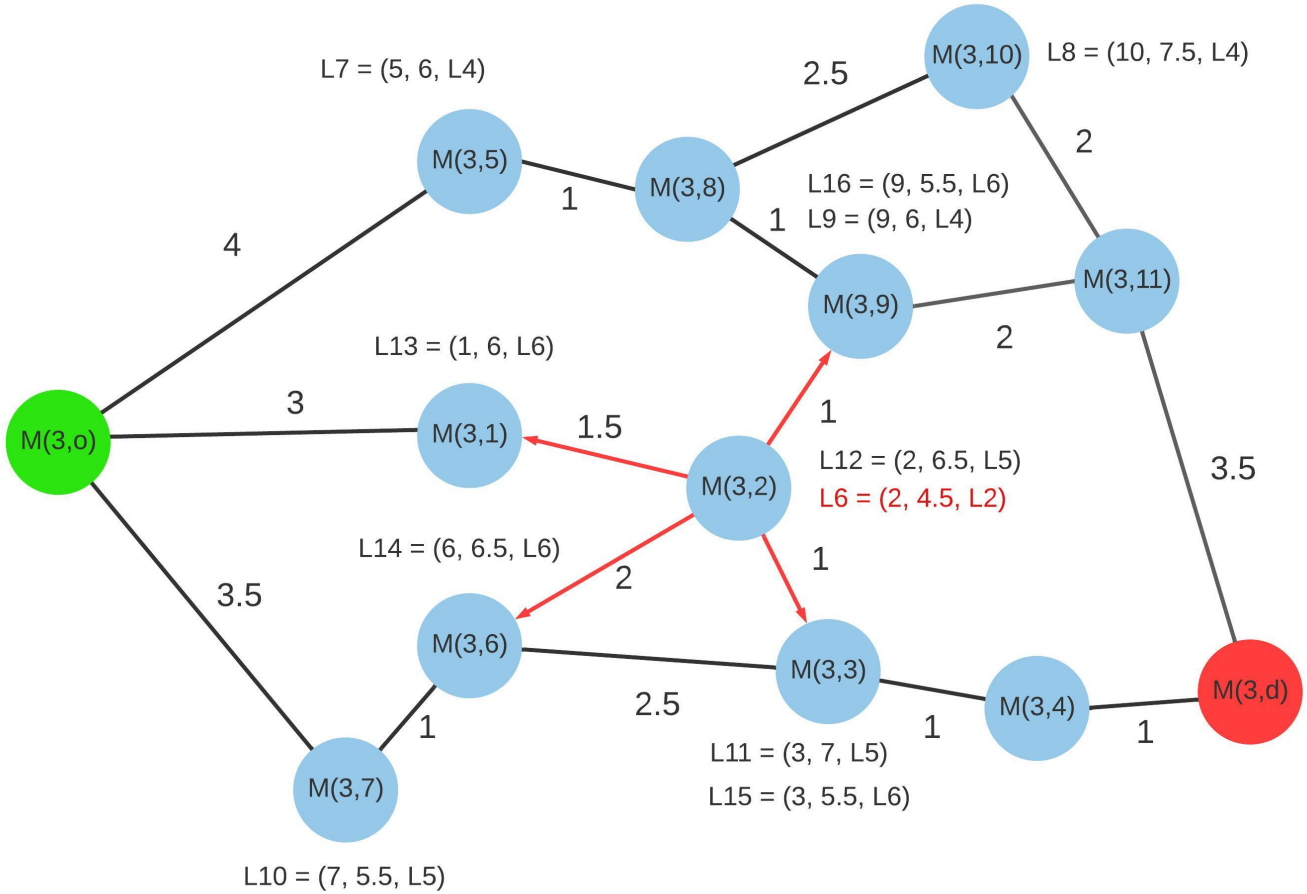
# Example: stage 3 (k = 3)



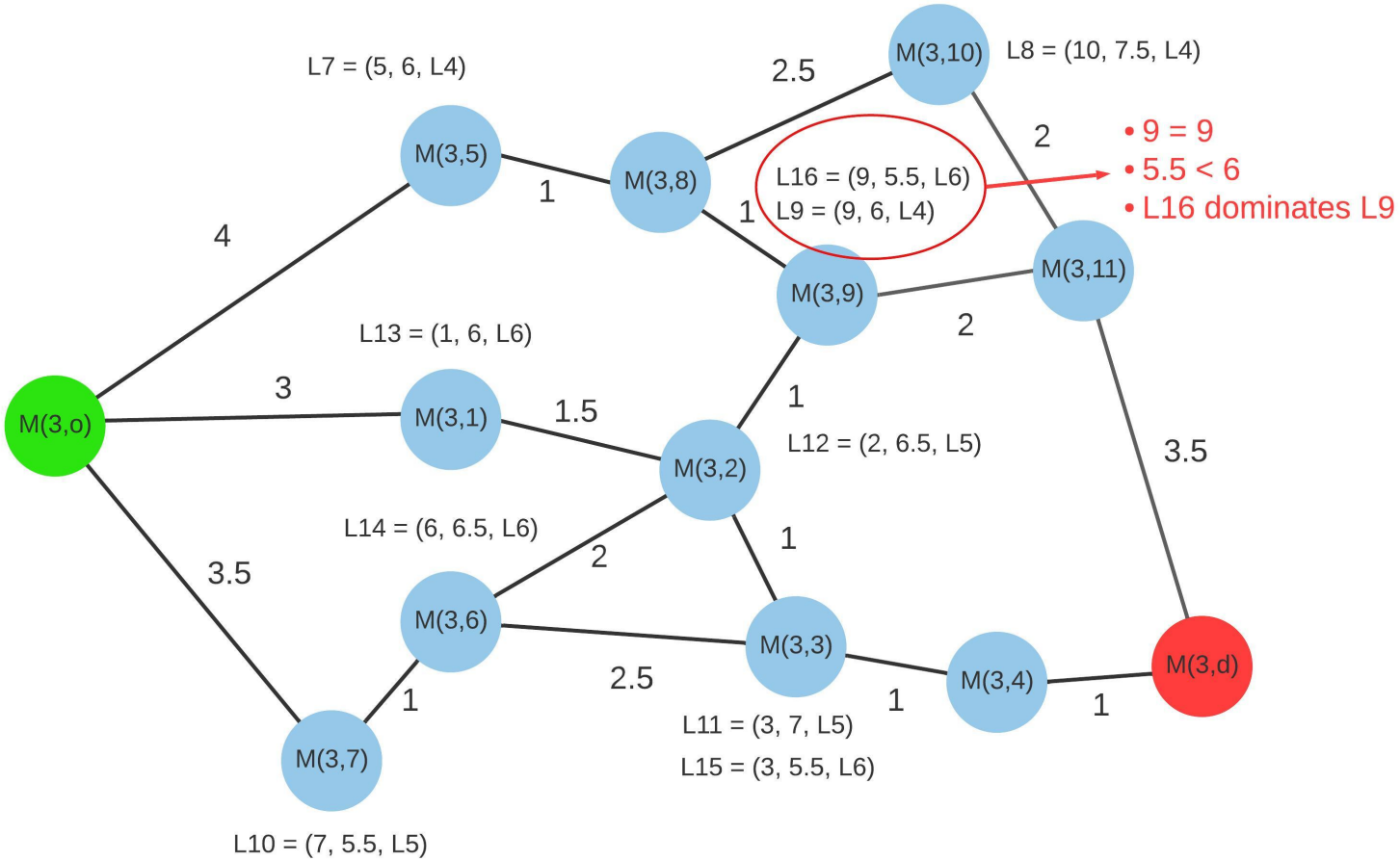
# Example: stage 3 (k = 3)



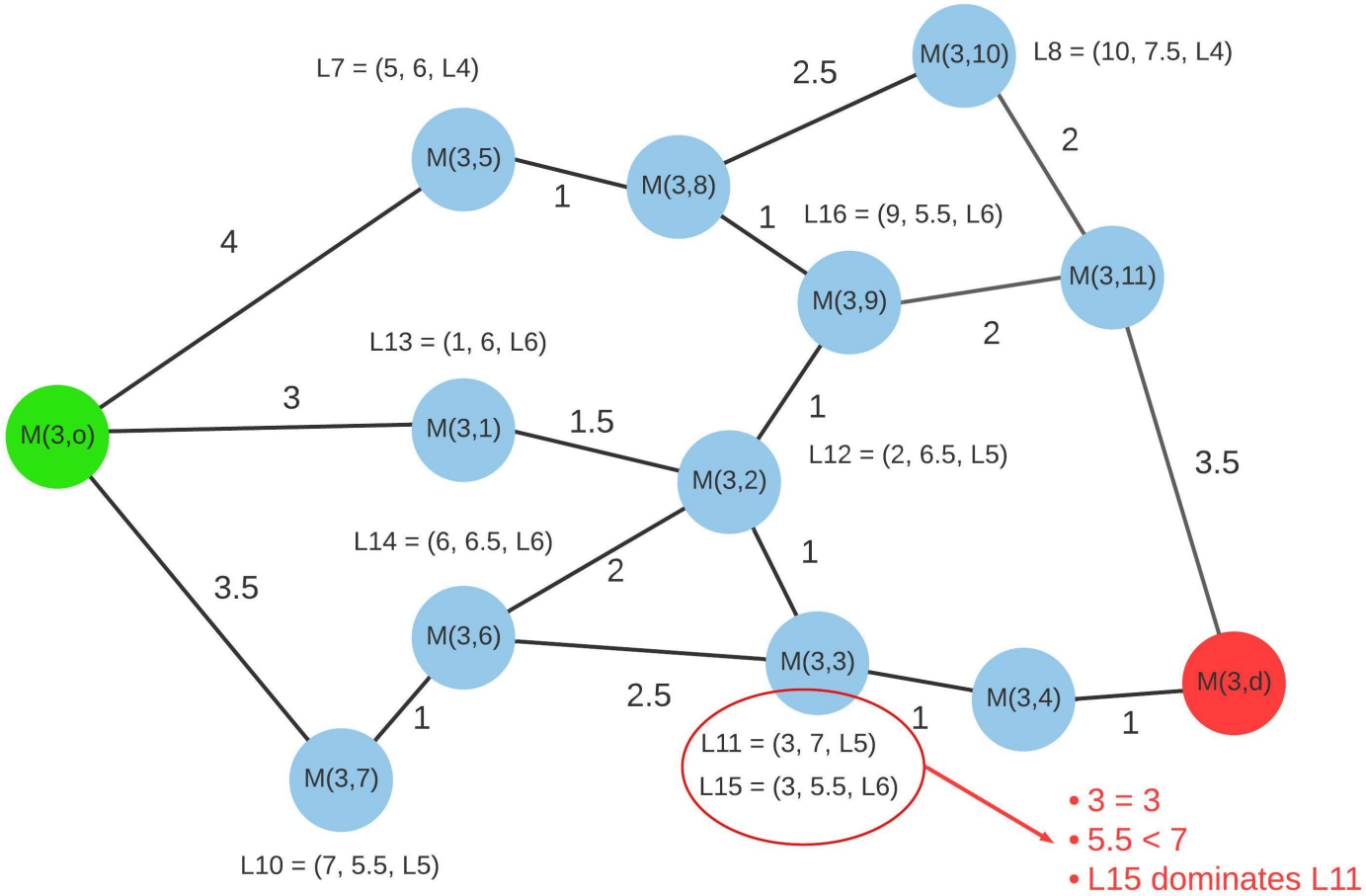
# Example: stage 3 (k = 3)



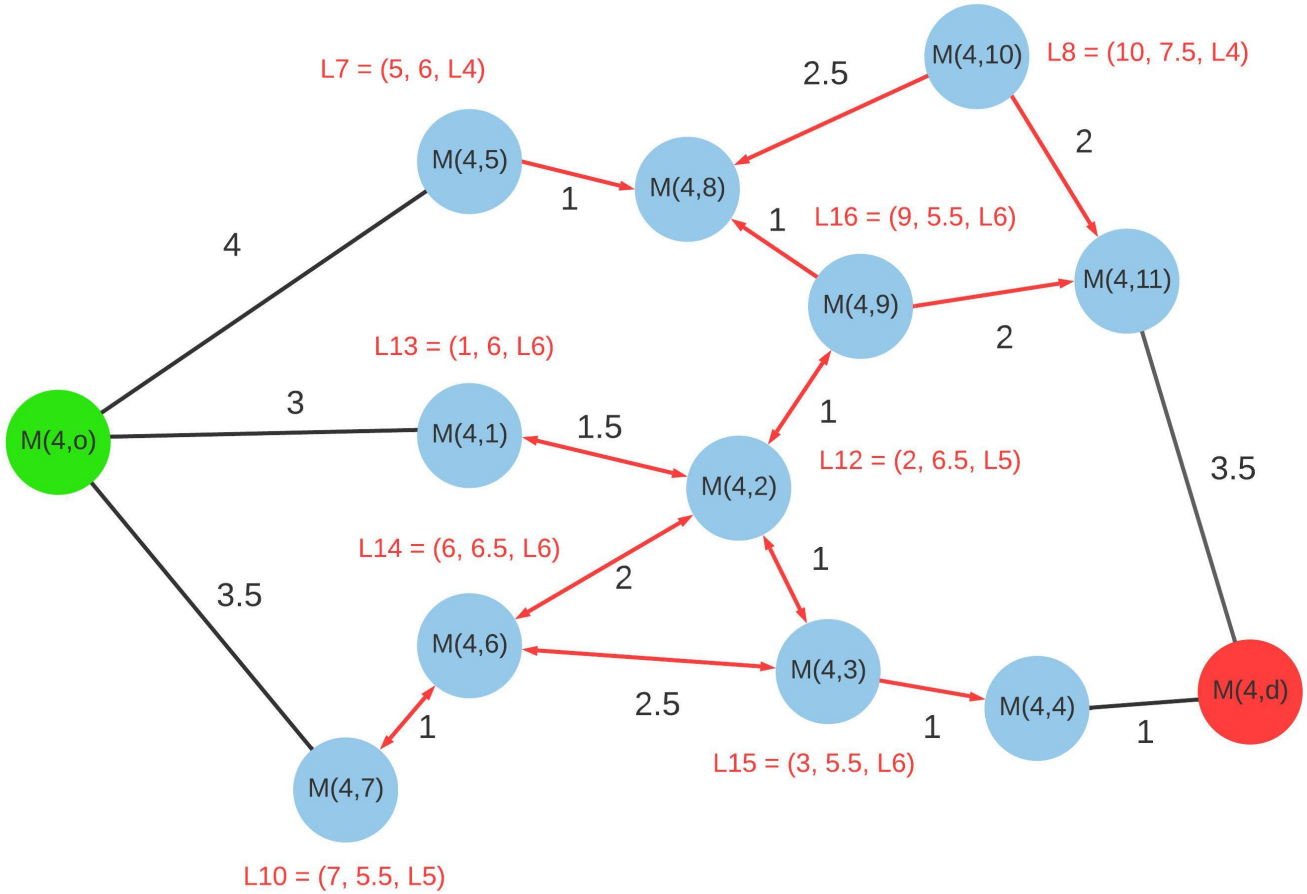
# Example: stage 3 (k = 3)



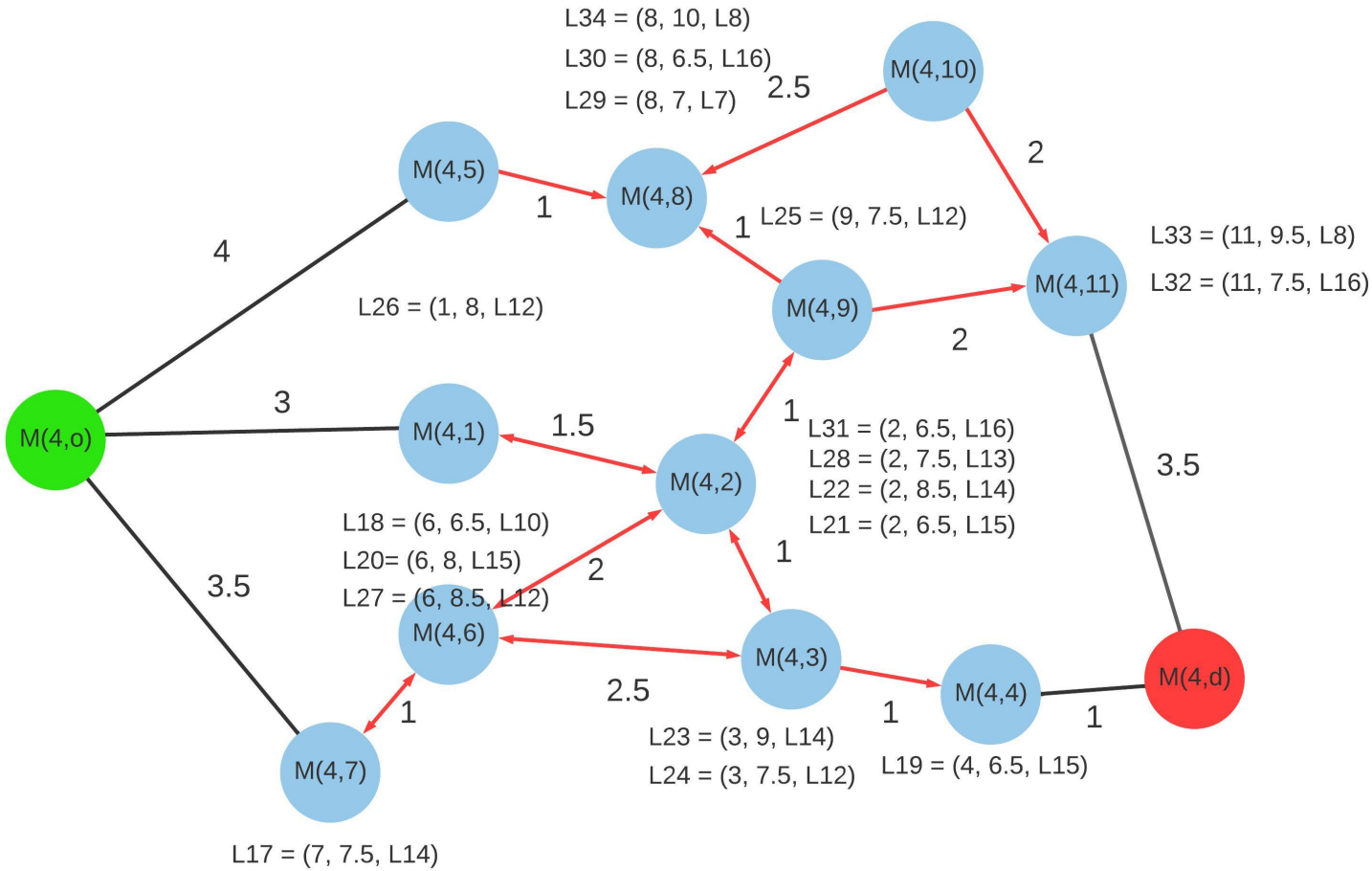
# Example: stage 3 (k = 3)



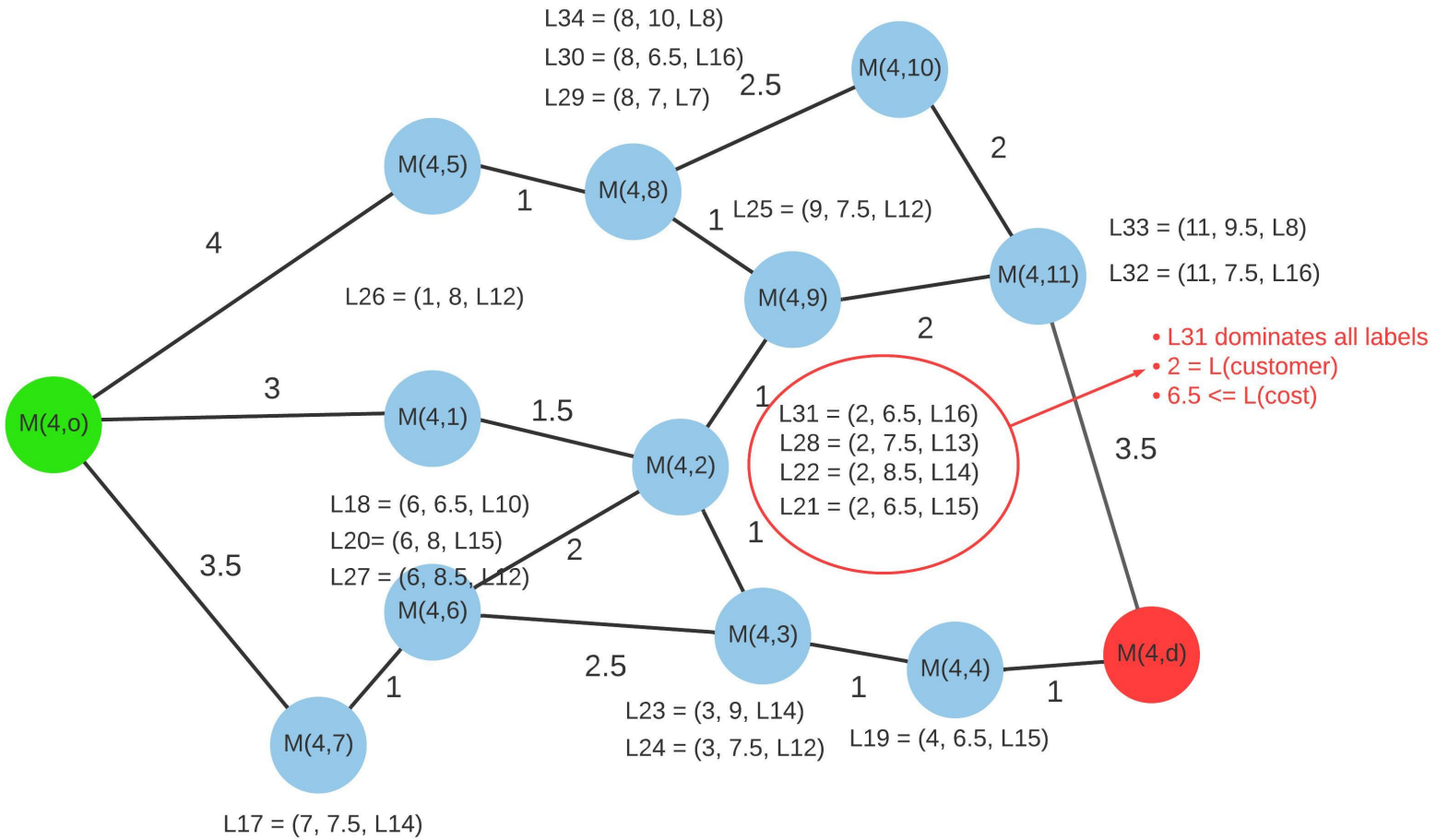
# Example: stage 4 (k = 4)



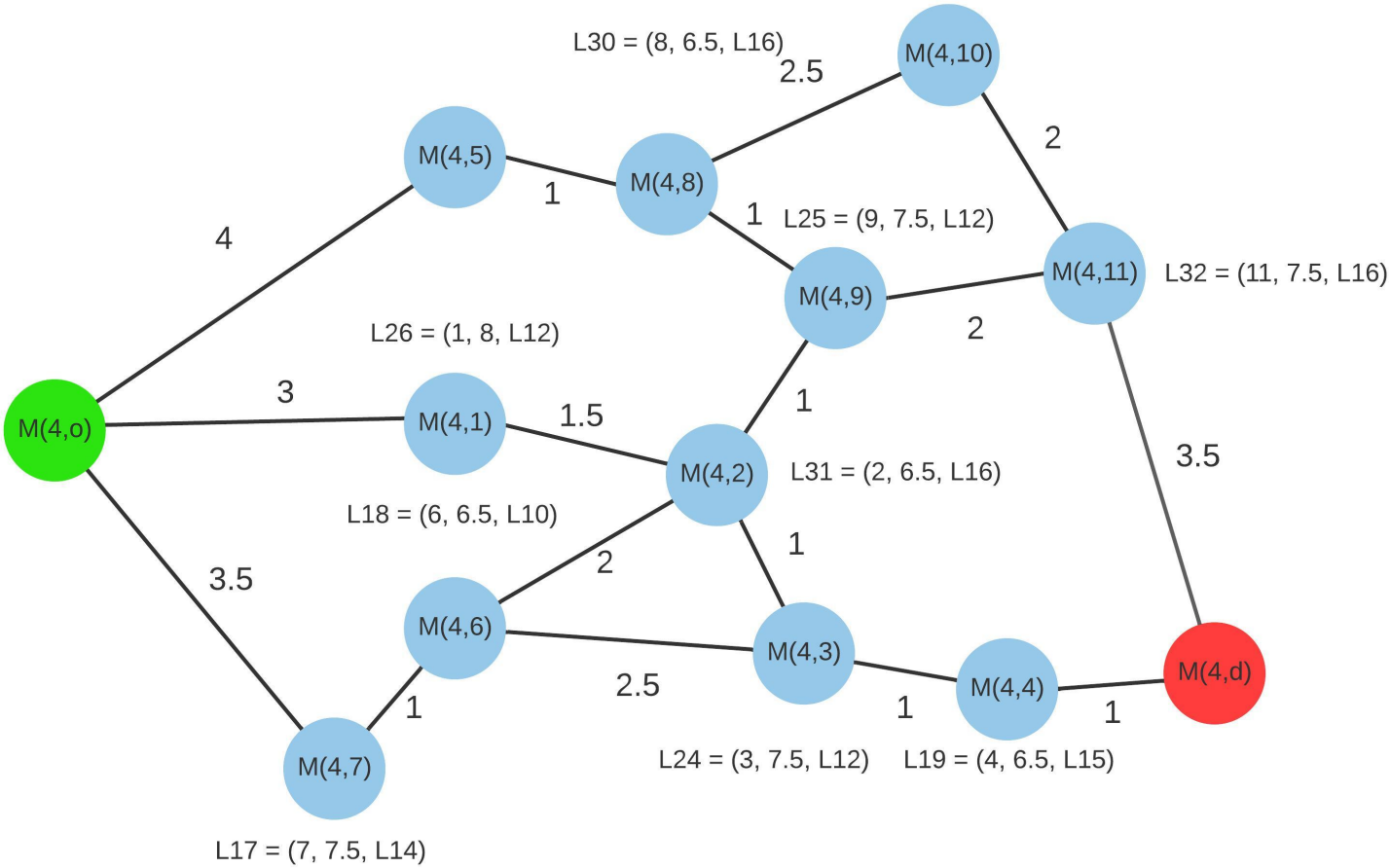
# Example: stage 4 (k = 4)



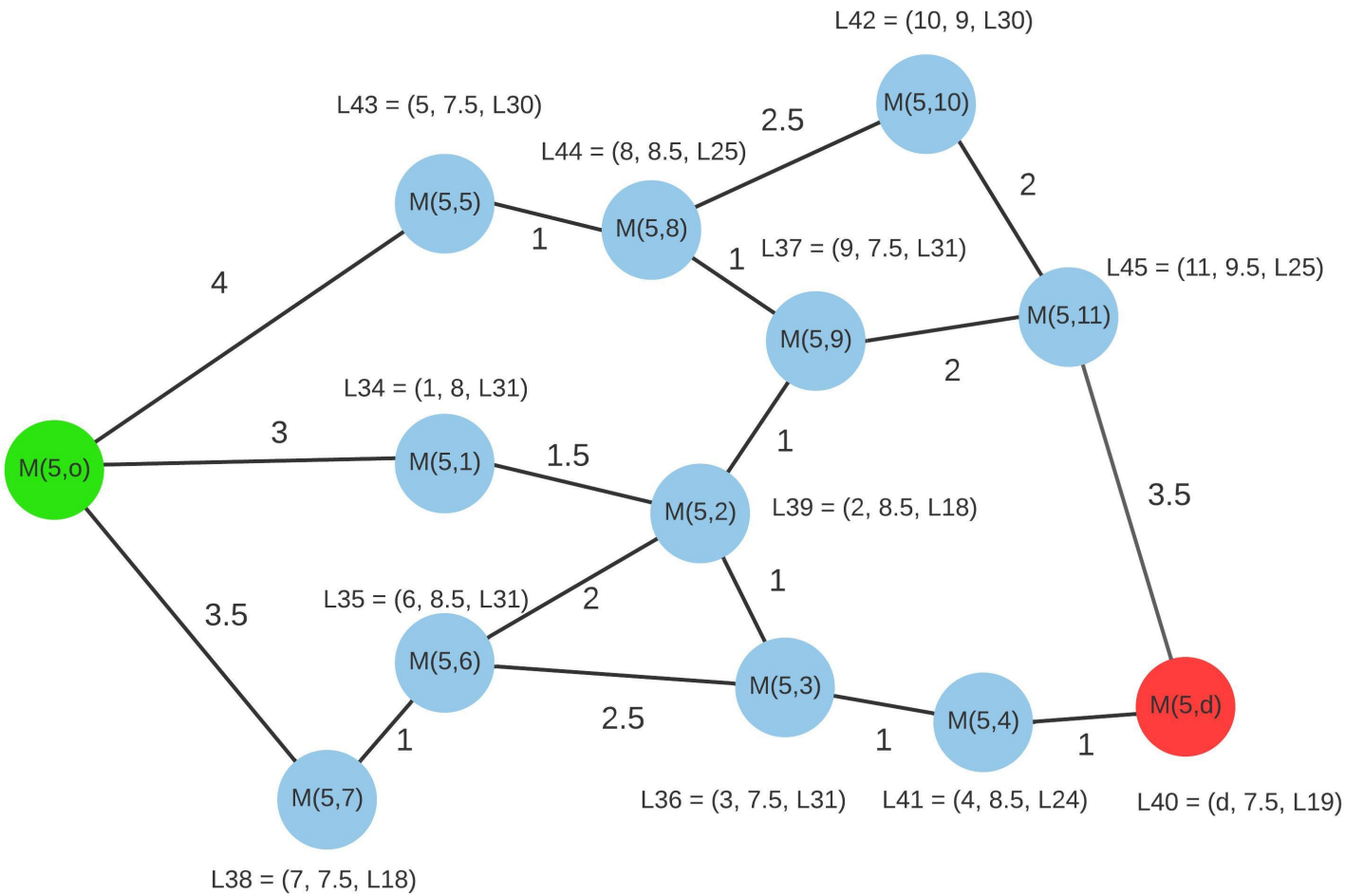
# Example: stage 4 (k = 4)



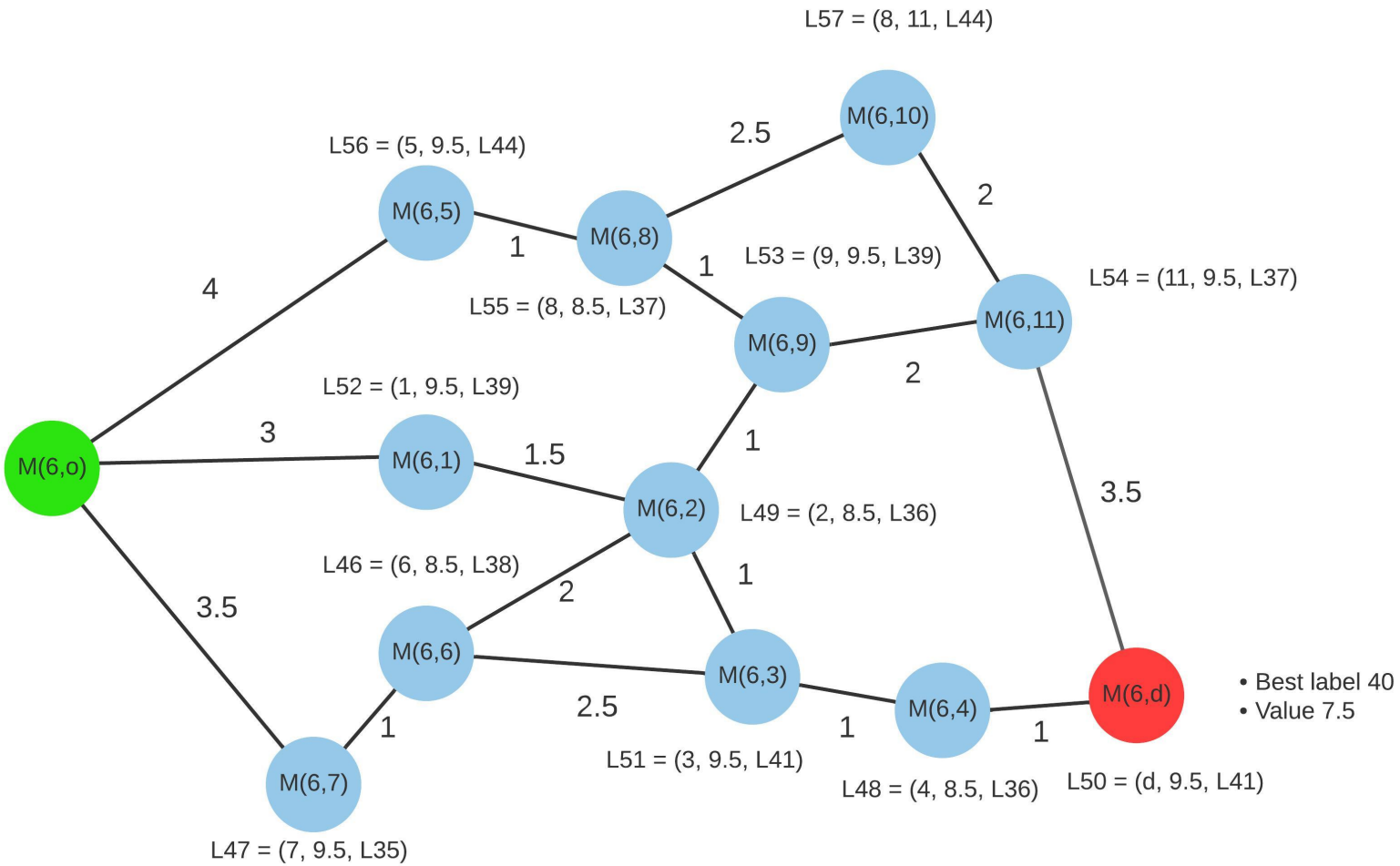
# Example: stage 4 (k = 4)



# Example: stage 5 (k = 5)



# Example: stage 6 (k = 6)



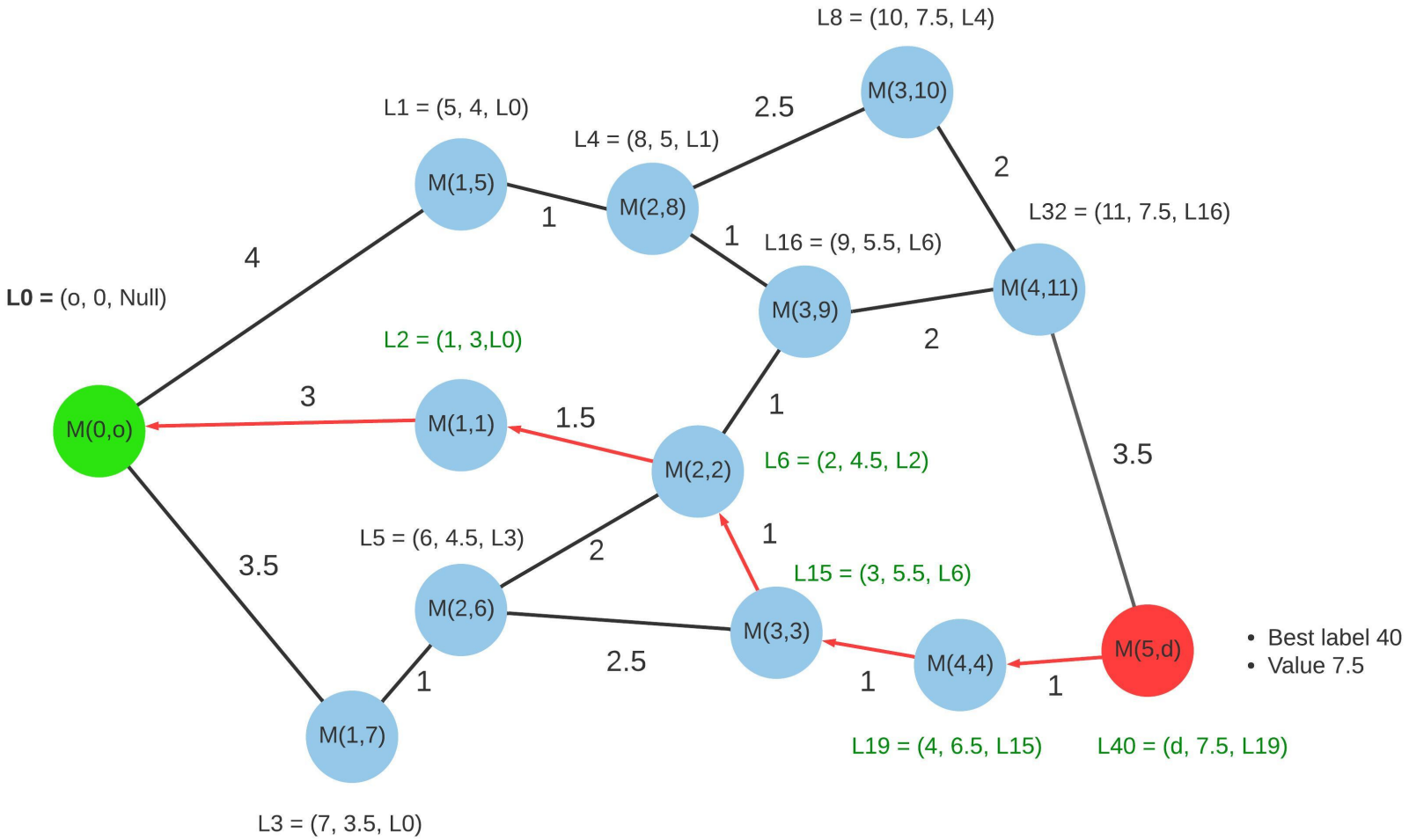
# Example: Dominating labels from all stages

**Recall the dominance rules**

1. $i_1 = i_2$
2. $c_1 \leq c_2$

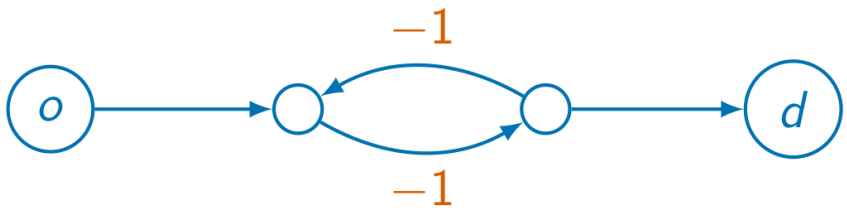
We can use other labels stored in $M(k, i)$ such that $k <= 6$ if the dominance rules are true, then we delete dominated labels.

# Example: Backtracking to get solution



## Negative cost cycle

If there exists a forward path from $o$ to $d$ containing a negative cost cycle, no forward path is the shortest path from $o$ to $d$.



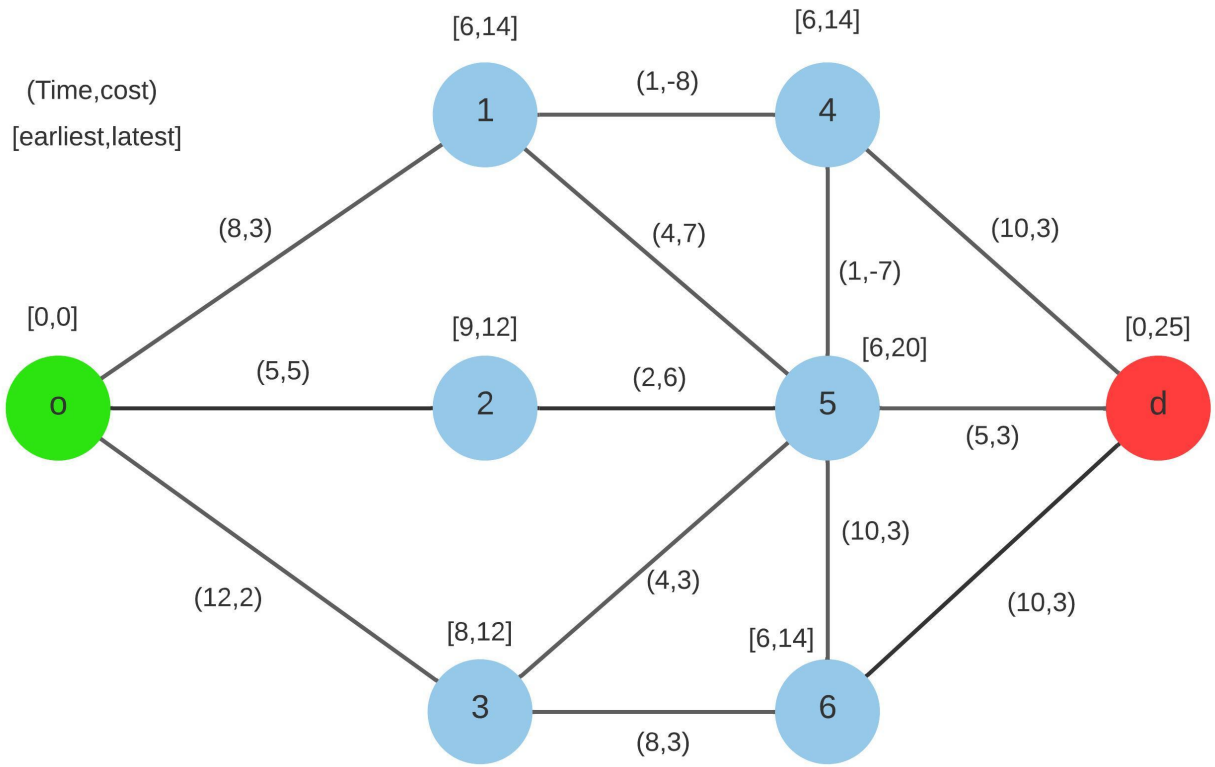

The shortest path problem is **Unbounded!** We can cycle forever.

# Outline



EPFL

# Shortest Path Problem with Resource Constraints



# Shortest Path Problem with Resource Constraints

**Time constraints**

- $e_i$ earliest arrival
- $b_i$ latest arrival,
- $t_{ij}$ time consumed in arc $(i, j)$,

**Capacity constraints**

- The total load of a path cannot exceed $Q$,
- $q_i$ is the demand of node $i$.

EPFL

# Labels

$$\mathcal{L}_{extend} = (i^e, c^e, T^e, Q^e, L^e)$$

Let $\mathcal{L}_{extend} = (i^e, c^e, T^e, Q^e, L^e)$ be a label that we want to extend from $i^e$ to $j^n$.

- $i^e$ is the current node,
- $c^e$ is the total cost traveled in the path of label,
- $T^e$ is the total time travel in the path that ends in the current node, i.e., $i^e$,
- $Q^e$ is the total load, i.e., the sum of all demand of nodes on the path.
- $j^n$ is a node connected to $i^e$.

EPFL

# Label extensions

- Extend label $\mathcal{L}_{extend}$ from node $i^e$ to node $j^n$,
- We have to create a new label $\mathcal{L}_{new}$,
- The cost is extended by adding the cost in arc $(i^e, j^n)$
- **Resources must be extended too!**
- **Resource constraints have to be checked before we extend!**

EPFL

# Resource Extension Functions (REF)

**REFs keep track of the resources that are used.**

At each extension we consume resources. REFs are functions that return the total amount of resource used by passing through an arc $(i, j)$.

- Capacity Resource Extension Function $f_Q = Q_i + q_j$
- Time Resource Extension Function $f_T = T_i + t_{ij}$

# Feasibility checks

**Before we extend a label we must check resource constraints.**

- Some extensions might not be feasible,
- Resource constraints could be violated,
- Time windows requires the path to arrive within the time window,
- Capacity constraints require the path to not exceed the capacity,
- At each extension we need to check all constraints allow the extension.

EPFL

# Feasibility checks

**If:**

- $T^e + t_{(i^e, j^n)} > b_{j^n}$, **Then** the extension is not feasible.

  **or**

- $Q^e + q_{j^n} > Q$, **Then** the extension is not feasible.

# Feasibility checks

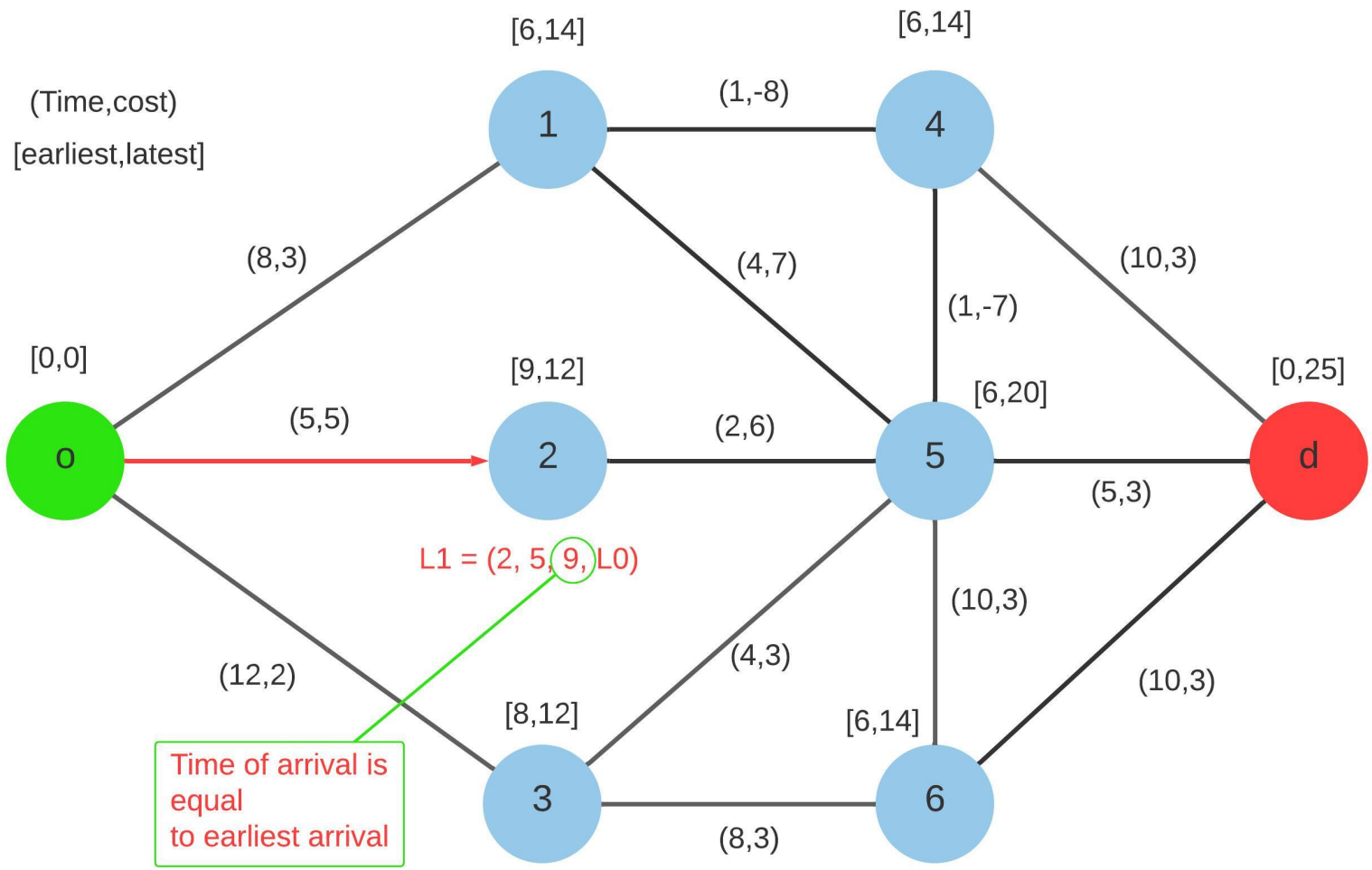
**What happens if we arrive early?**

1. Waiting is feasible,
2. If $T^e + t_{(i^e,j^n)} < e_{j^n}$ we can wait at node $j^n$ until $e_{j^n}$,
3. The waiting time must be added to the resource consumption,
4. The REF for time is modified to $f_T = max\{e_{j^n}, T^e + t_{(i^e,j^n)}\}$.

New label is

$$\mathcal{L}_{new} = (j^n, c^n, f_T, f_Q, \mathcal{L}_{extend})$$

# Example waiting



# Dominance

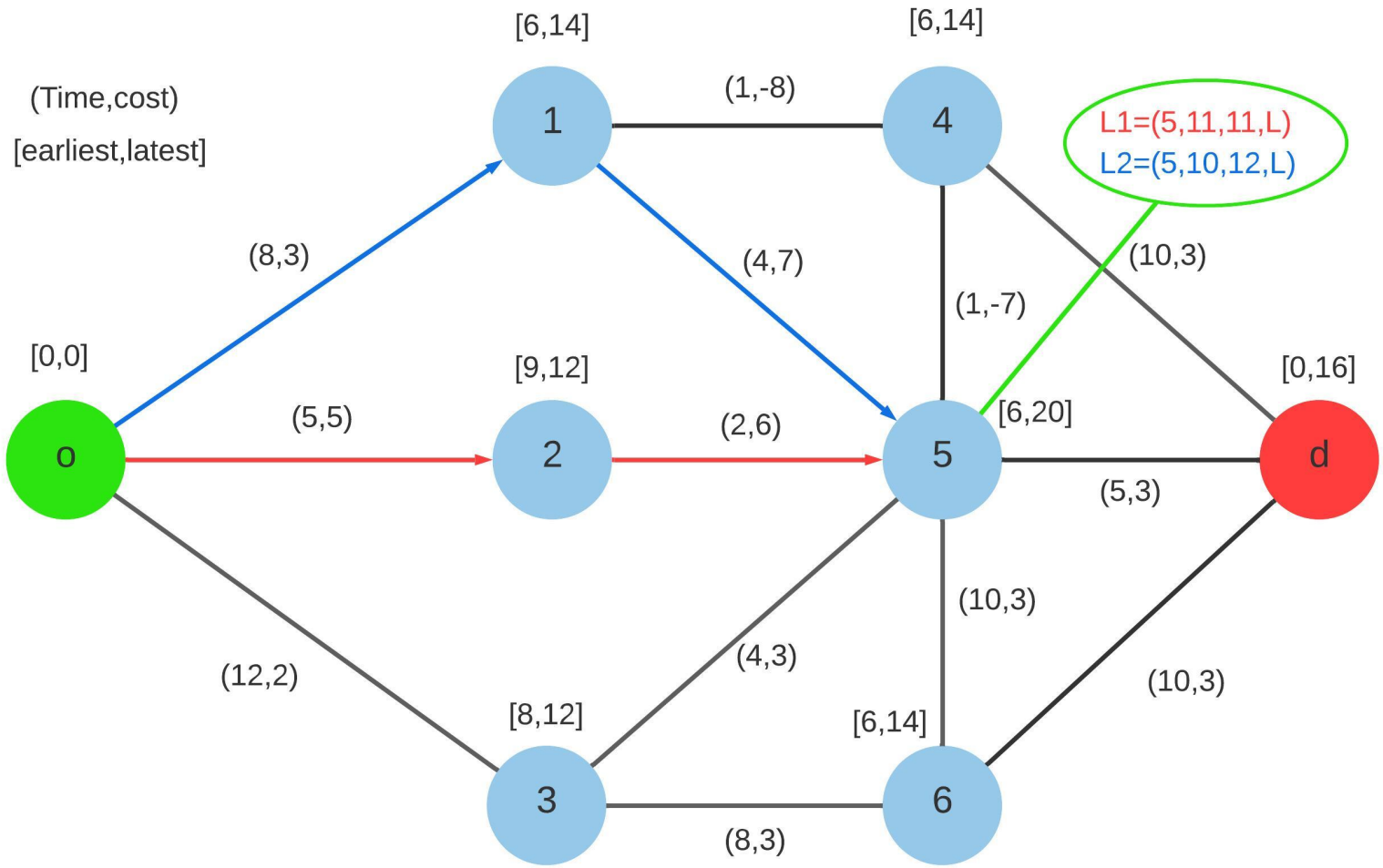
We can use dominance rules to delete labels as before, however, dominance rules need to be modified.

**Dominance rules:**
Label $\mathcal{L}_1 = (i_1, c_1, T_1, Q_1, L_1)$ dominates label $\mathcal{L}_2 = (i_2, c_2, T_2, Q_2, L_2)$ if the following is true:

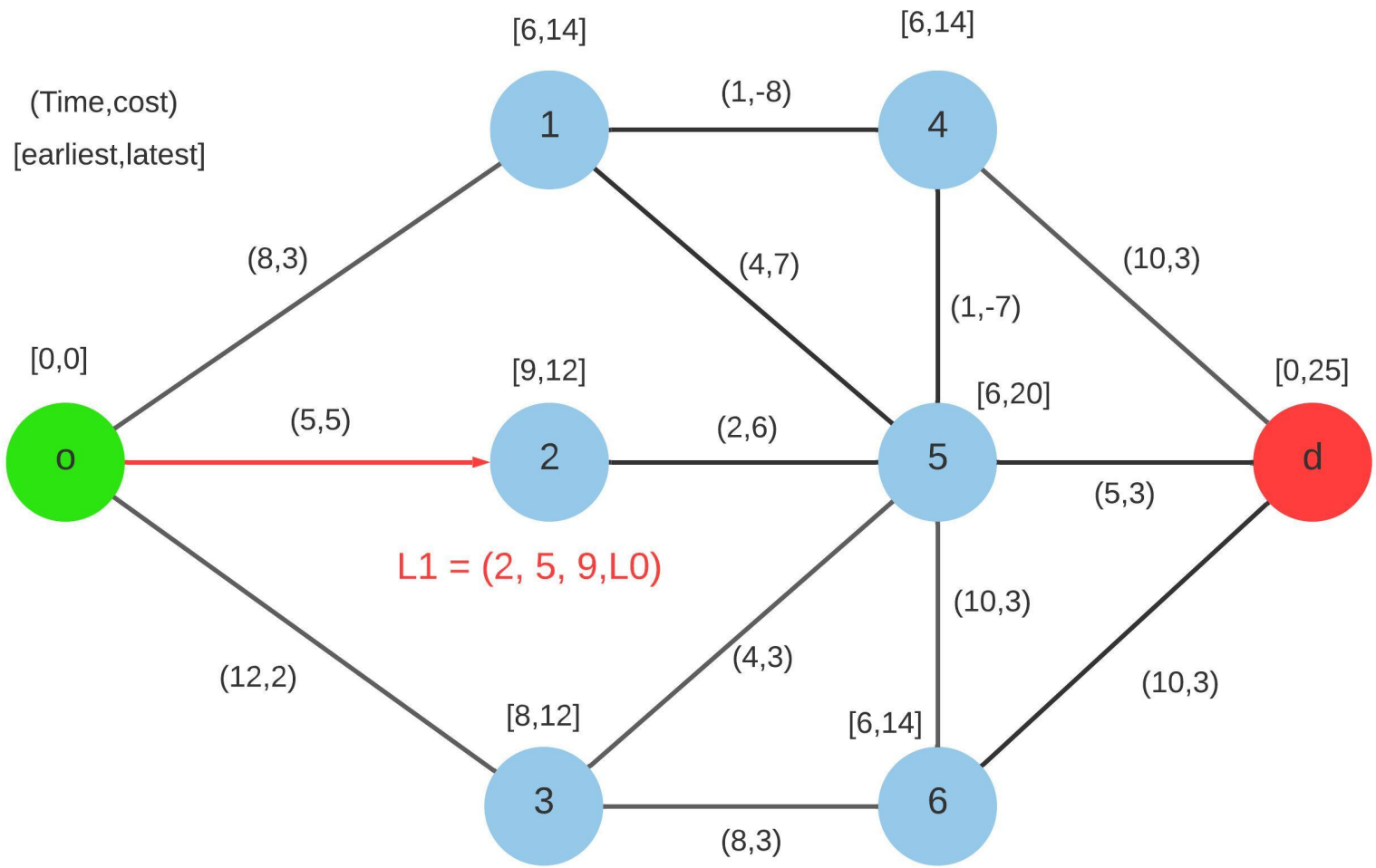
1. $i_1 = i_2$
2. $c_1 \leq c_2$
3. $T_1 \leq T_2$
4. $Q_1 \leq Q_2$

EPFL

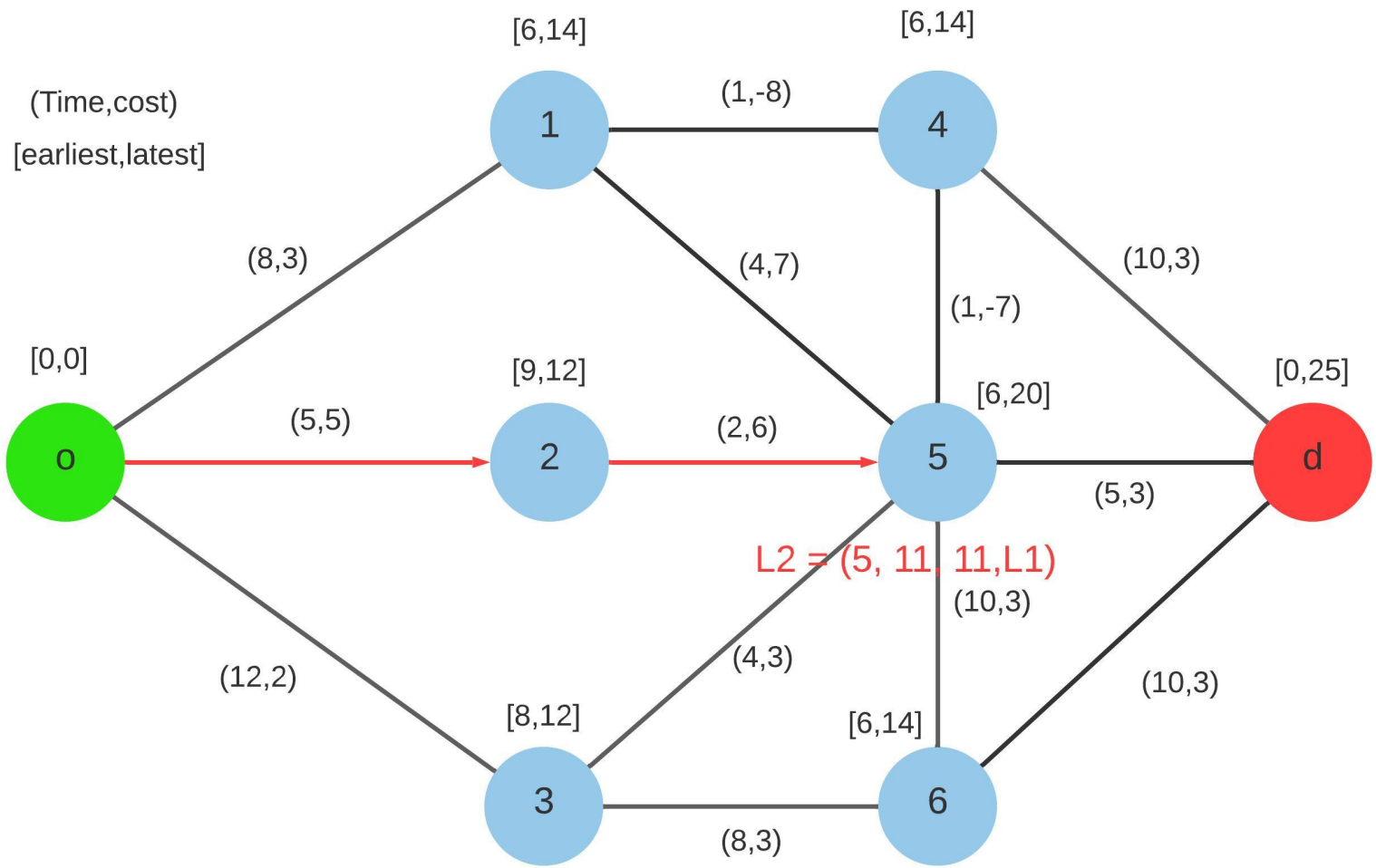
# Example: Dominance



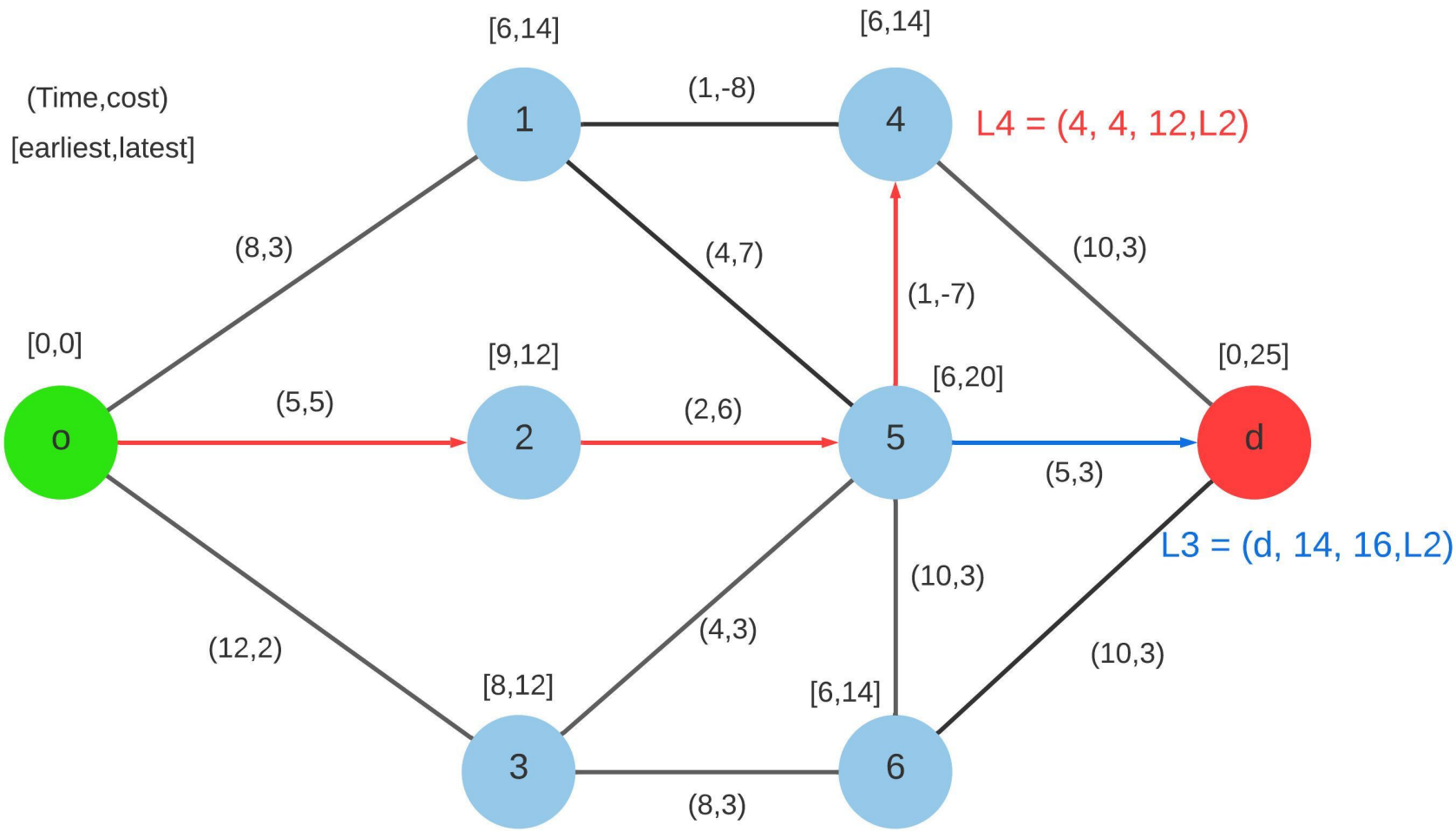
# Example: negative cost cycle



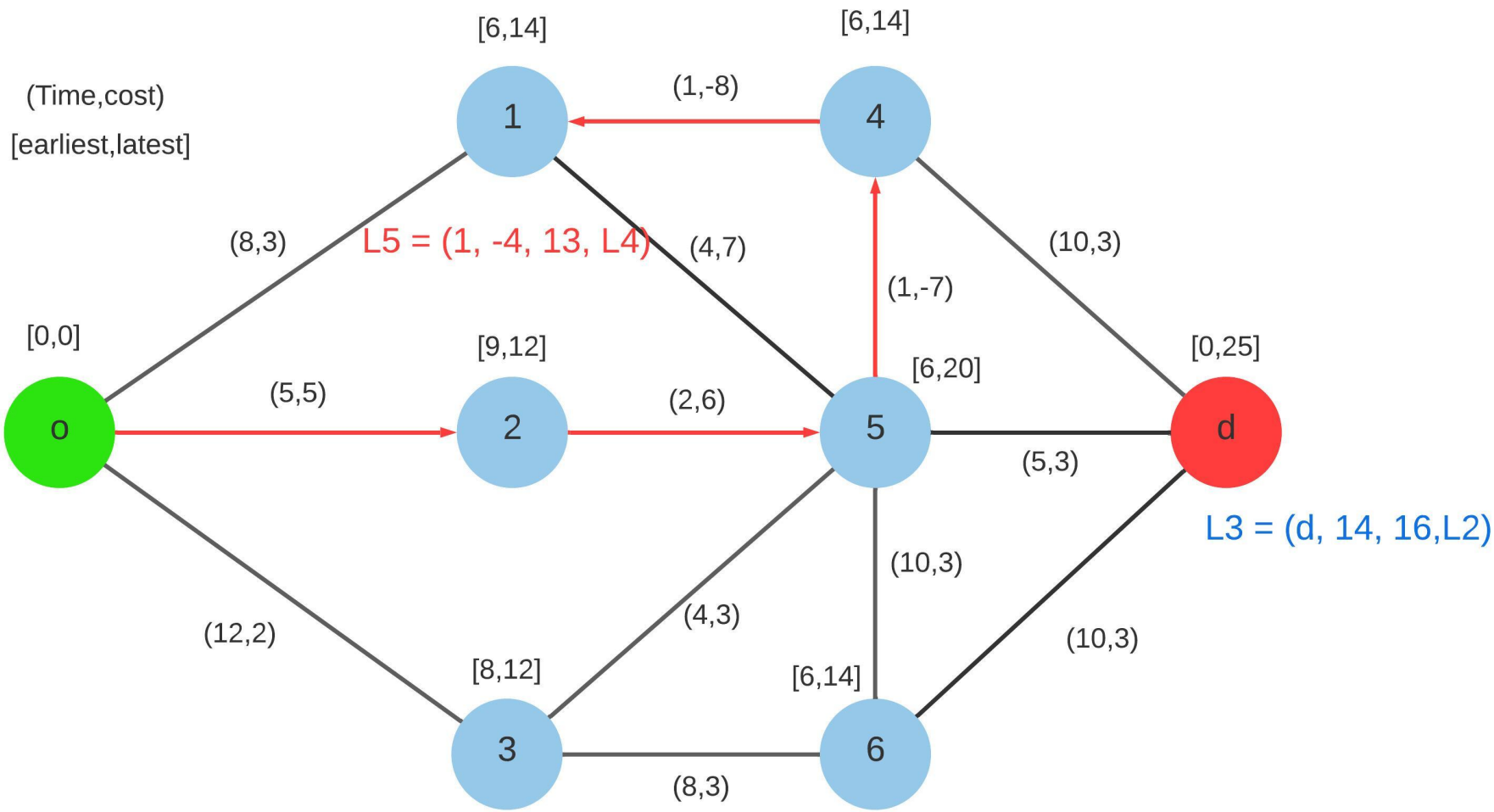
# Example: negative cost cycle



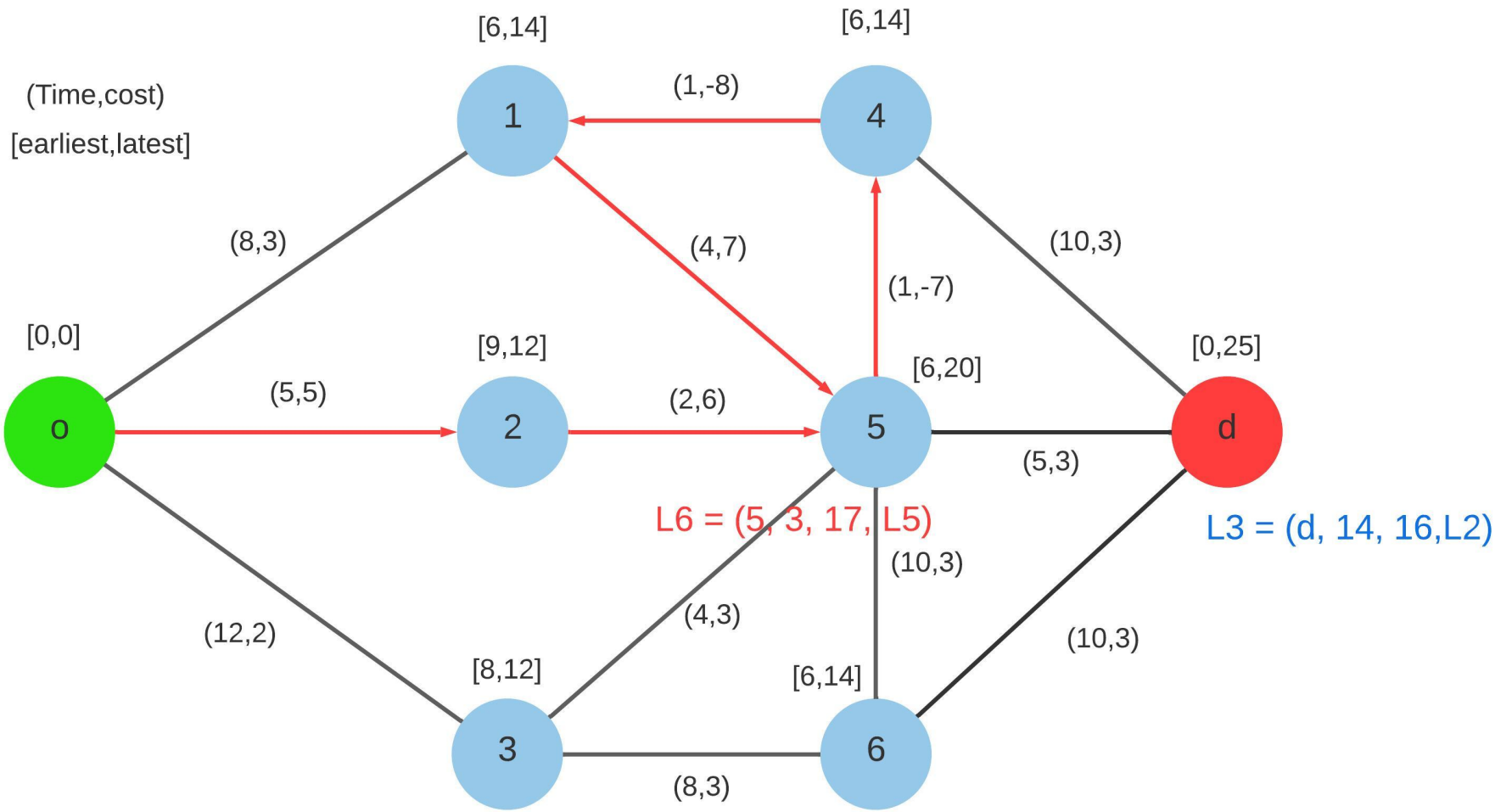
# Example: negative cost cycle



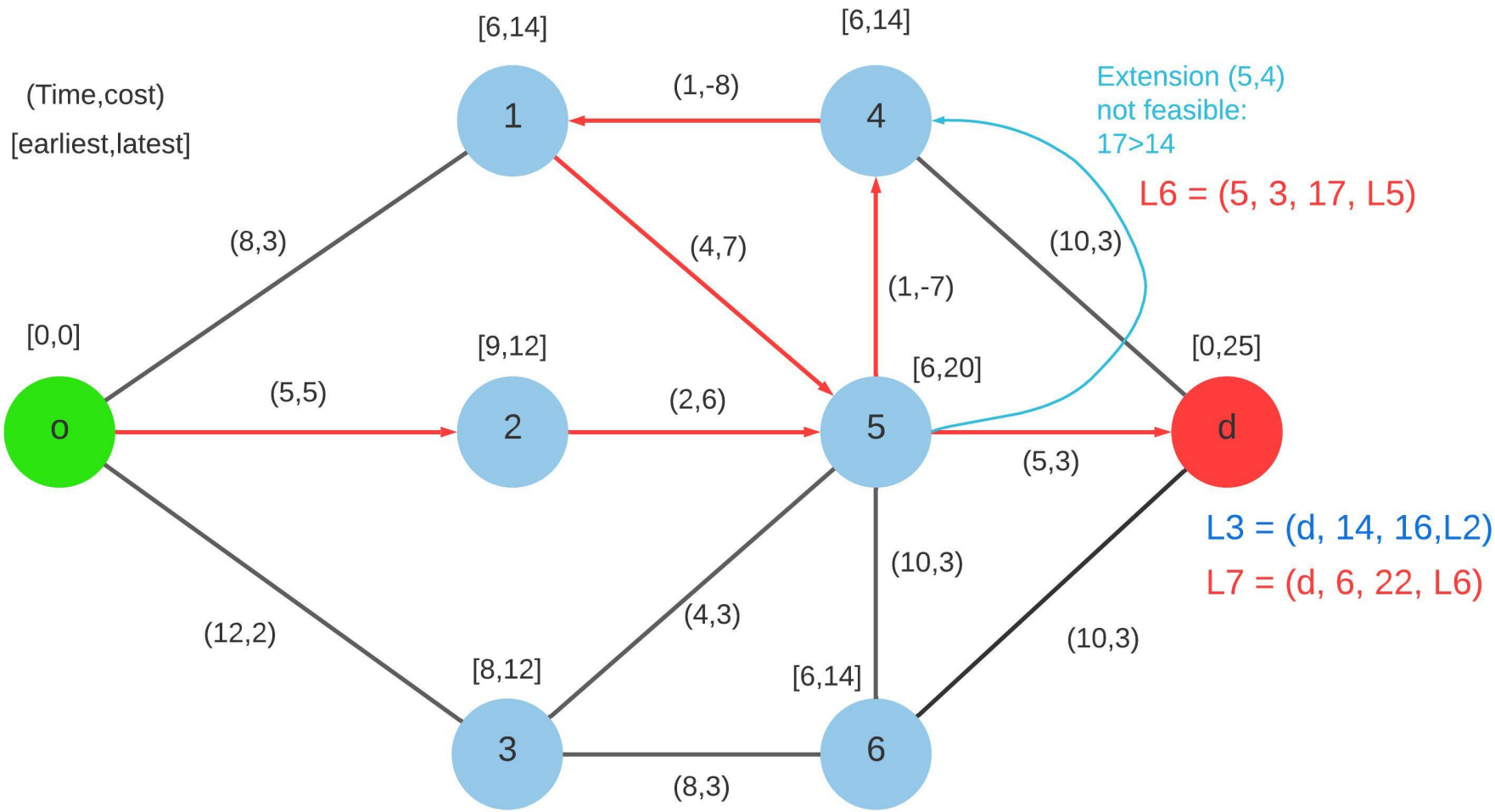
# Example: negative cost cycle



# Example: negative cost cycle



# Example: negative cost cycle



# Outline



EPFL

# Find the route with minimum reduced cost

Elemetary Shortest Path Problem with Resource Constraints (ESPPRC)

$$
\begin{aligned}
&Min \sum_{i \in V} \sum_{j \in V} \hat{c}_{ij} x_{ij} \\
&s.t. \sum_{i \in N} q_i \sum_{j \in V} x_{ij} \leq Q, \\
&\sum_{j \in N} x_{0j} = 1, \\
&\sum_{i \in V} x_{i0} = 1, \\
&\sum_{i \in N} x_{ih} - \sum_{j \in N} x_{hj} = 0 && \forall h \in N, \\
&T_i + t_{ij} - M_{ij}(1 - x_{ij}) \leq T_j && \forall i, j \in V, \\
&a_i \leq T_i \leq b_i && \forall i \in N, \\
&x_{ij} \in \{0, 1\} && \forall i, j \in N.
\end{aligned}
$$

# Negative reduced cost

- The pricing problem for the VRP has negative cost cycles,
- Elementarity constraints (i.e., no cycles allowed) are used to prevent cycles from forming,
- Vehicles cannot visit the same customer multiple times.

EPFL

# Unreachable nodes

- Let $V^e$ be the set of unreachable nodes for label $\mathcal{L}_{extend}$,
- If a node is in set $V^e$ then it is unfeasible to visit this node from label $\mathcal{L}_{extend}$.

$$\mathcal{L}_{extend} = (i^e, c^e, T^e, Q^e, V^e, L^e)$$

EPFL

# Label extensions

**Add an extension for the set of unreachable nodes!**

1. $V^n = V^e \cup \{j^n\}$
2. The REF for capacity is $f_Q = Q^e + q_{j^n}$
3. The REF for time is $f_T = max\{e_{j^n}, T^e + t_{(i^e, j^n)}\}$.

New label is:

$$\mathcal{L}_{new} = (j^n, c^n, f_T, f_Q, V^n, \mathcal{L}_{extend})$$

# Feasibility checks

**Before we extend a label we must check all constraints.**

- $j^n \in V^e$, **Then** the extension is not feasible.
  **or**
- $T^e + t_{(i^e, j^n)} > b_{j^n}$, **Then** the extension is not feasible.
  **or**
- $Q^e + q_{j^n} > Q$, **Then** the extension is not feasible.

EPFL

# Dominance

We can use dominance rules to delete labels as before, however, dominance rules need to be modified.

**Dominance rules:**
Label $\mathcal{L}_1 = (i_1, c_1, T_1, Q_1, V_1, L_1)$ dominates label $\mathcal{L}_2 = (i_2, c_2, T_2, Q_2, V_2, L_2)$ if the following is true:

1. $i_1 = i_2$
2. $c_1 \leq c_2$
3. $T_1 \leq T_2$
4. $Q_1 \leq Q_2$
5. $V_1 \subseteq V_2$

EPFL

# Dominance

**Exponential growth of labels!**

- Dominance rule (5) makes it difficult to delete labels through dominance,
- There is an exponential number of subsets of nodes,
- Labels grow exponentially.

EPFL

# Relaxation

$$\text{ESPPRC} \xrightarrow{relax} \text{SPPRC}$$

When we relax the elemetarity constraints we have the SPPRC as a relaxation, i.e., the solution of the SPPRC will give us a lower bound on the ESPPRC.

- What about cycles?
- Cycles are added as feasible routes.
- Will the solution to the VRP contain routes that visit customers multiple times?

EPFL

# Master Problem: The set-covering formulation

- We replace the set-partitioning formulation with the **Set-covering formulation**

**Master problem (MP)**

$$\text{Minimize} \sum_{r \in \Omega} c_r \lambda_r$$

$$s.t. \sum_{r \in \Omega} a_{ir} \lambda_r \geq 1 \leftarrow \qquad \forall i \in N,$$

$$\lambda_r \in \{0, 1\} \qquad \forall r \in \Omega.$$

# Master Problem: The set-covering formulation

- **Will the solution to the VRP contain routes that visit customers multiple times?**
- No!
- **Why?**
- Routes with cycles are more expensive than routes without cycles,
- The optimal integer solution will have routes that visit customers only once,
- However, the lower bounds obtained by the relaxation are weak.

EPFL

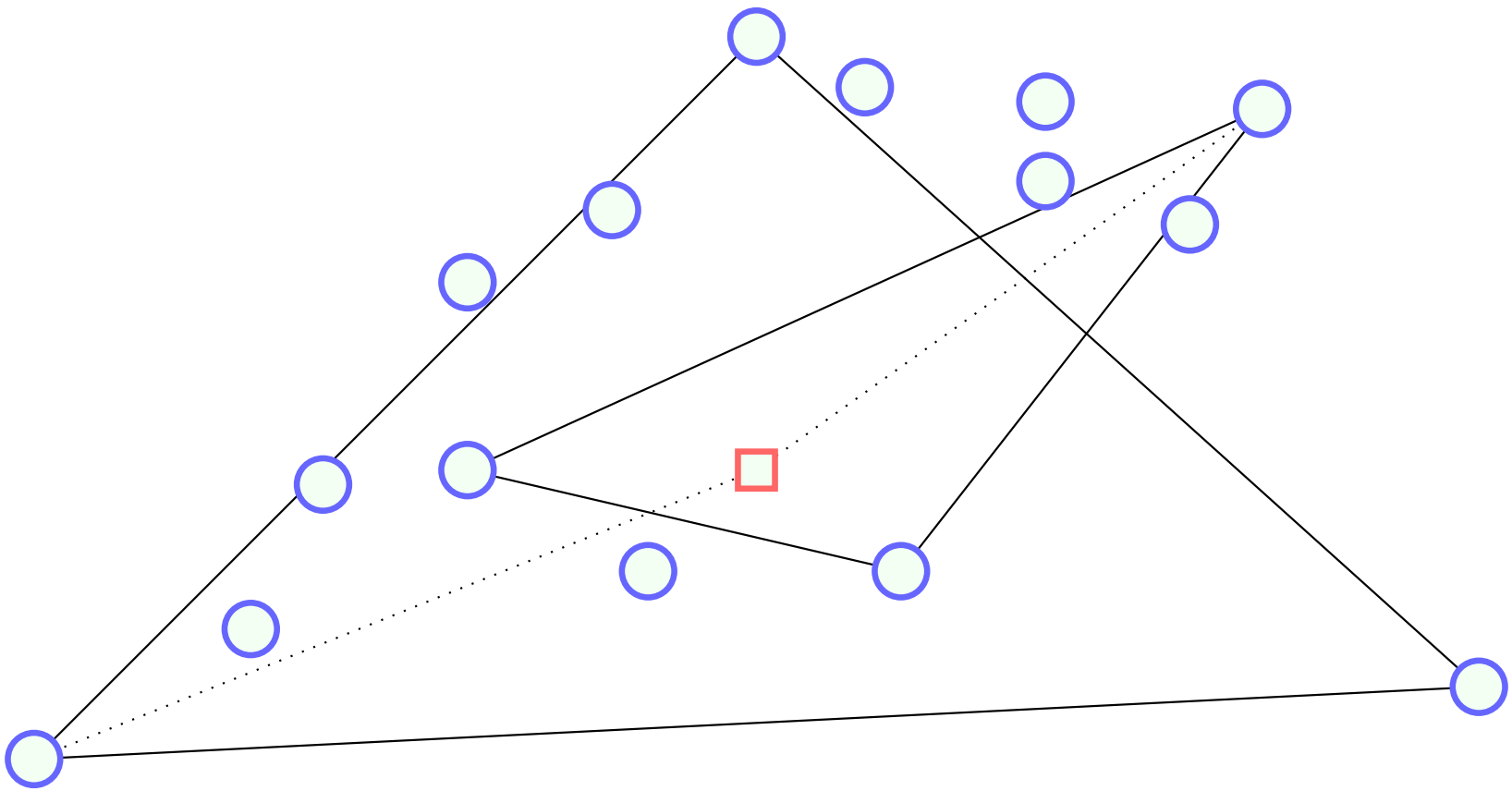
# Relaxation: k-cycle

Only cycles containing more than k nodes are allowed.

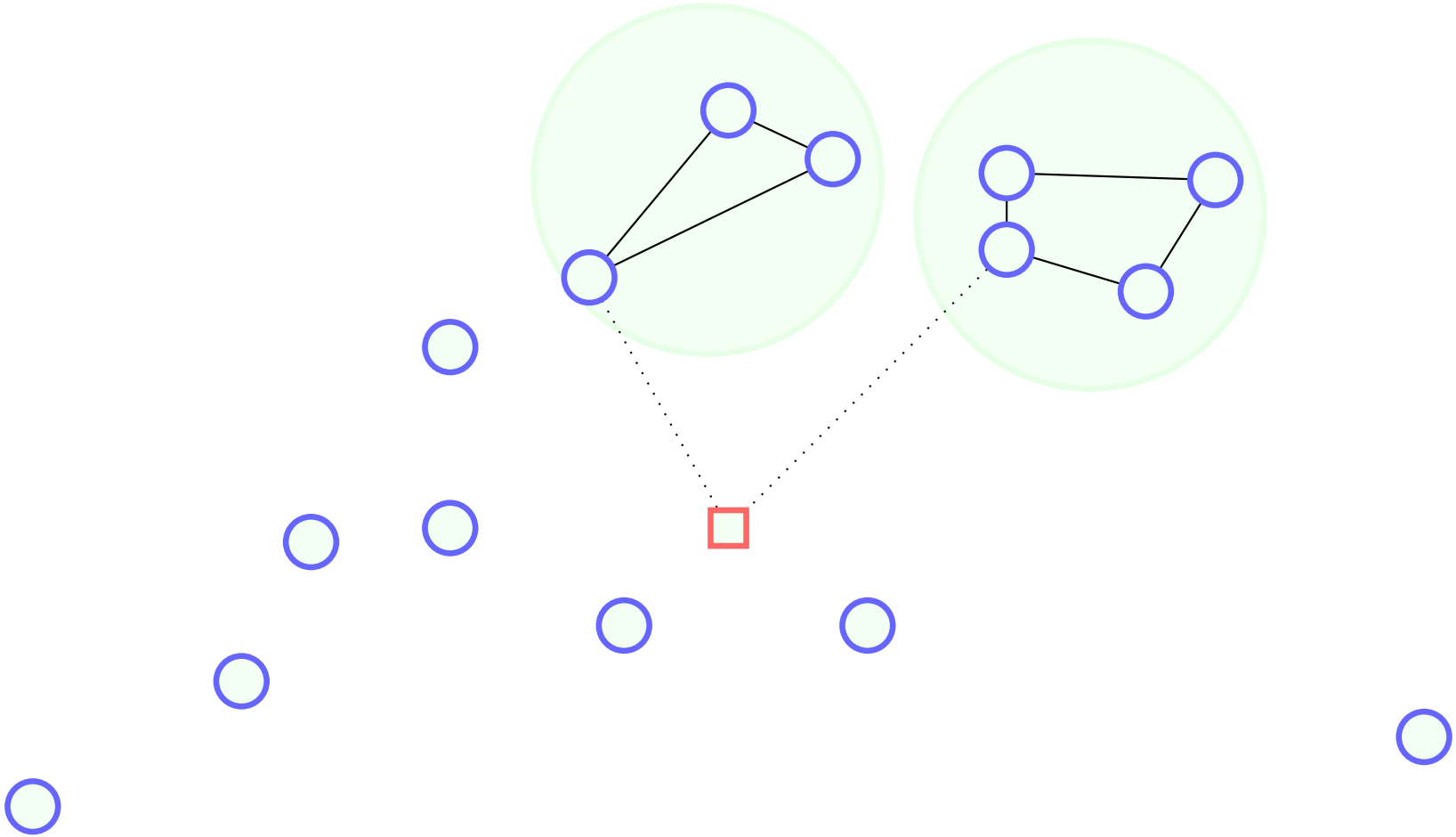
**Lower bound**

The lower bound from this relaxation is better than the SPPRC.

# Relaxation: 2-cycle



# Relaxation: 2-cycle



# Relaxation: Ng-route

- Eliminate cycles formed by close neighbors
- Only allow cycles formed by distant customers.

**Lower bound**

The lower bound from the Ng-route relaxation is better than the k-cycle relaxation.

# Relaxation: Ng-route

- $Ng^i$ is a set of nearest customers to customer $i \in N$.
- For every customer we create an Ng-set that contains its $\Delta$ nearest neighbors, e.g., $\Delta = 5$ means than the set $Ng^i$ contains the 5 nearest neighbors of the customer $i$.
- When we extend a label from $i^e$ to a distant neighbor not in set $j^n \notin Ng^{i^e}$ we can forget some customers that we could visit again.

EPFL

# Ng-route relaxation: Labels

Labels keep track of the set of unreachable customers that are allowed to be visited next.
The full set of unreachable customers $V$ is replaced by a smaller set $U$.

$$\mathcal{L} = (i, c, T, Q, V, L) \rightarrow \mathcal{L} = (i, c, T, Q, U, L)$$

$$\mathcal{L}_{extend} = (i^e, c^e, T^e, Q^e, U^e, L^e)$$

- Let $V^e$ be the set of unreachable nodes for label $\mathcal{L}_{extend}$,
- If a node is in set $V^e$ then it is unfeasible to visit this node from label $\mathcal{L}_{extend}$.

EPFL

# Label extensions

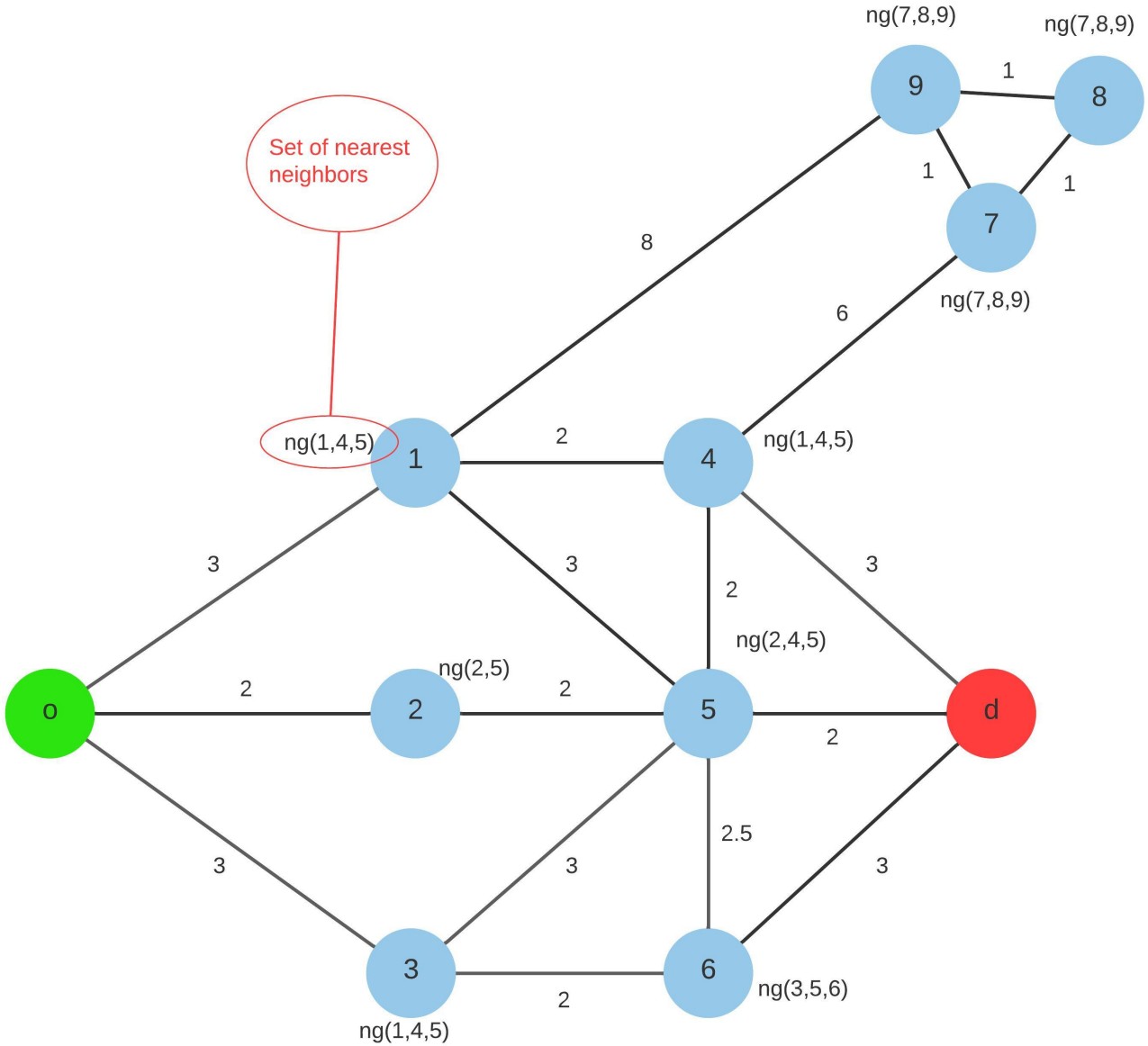
**Add an extension for the set of unreachable nodes!**

1. $U^n = U^e \cap Ng^n \rightarrow$ Only common elements are kept.
2. $U^n = U^n \cup \{j^n\} \rightarrow$ Add the current customer to the set.
3. The REF for capacity is $f_Q = Q^e + q_{j^n}$
4. The REF for time is $f_T = max\{e_{j^n}, T^e + t_{(i^e, j^n)}\}$.

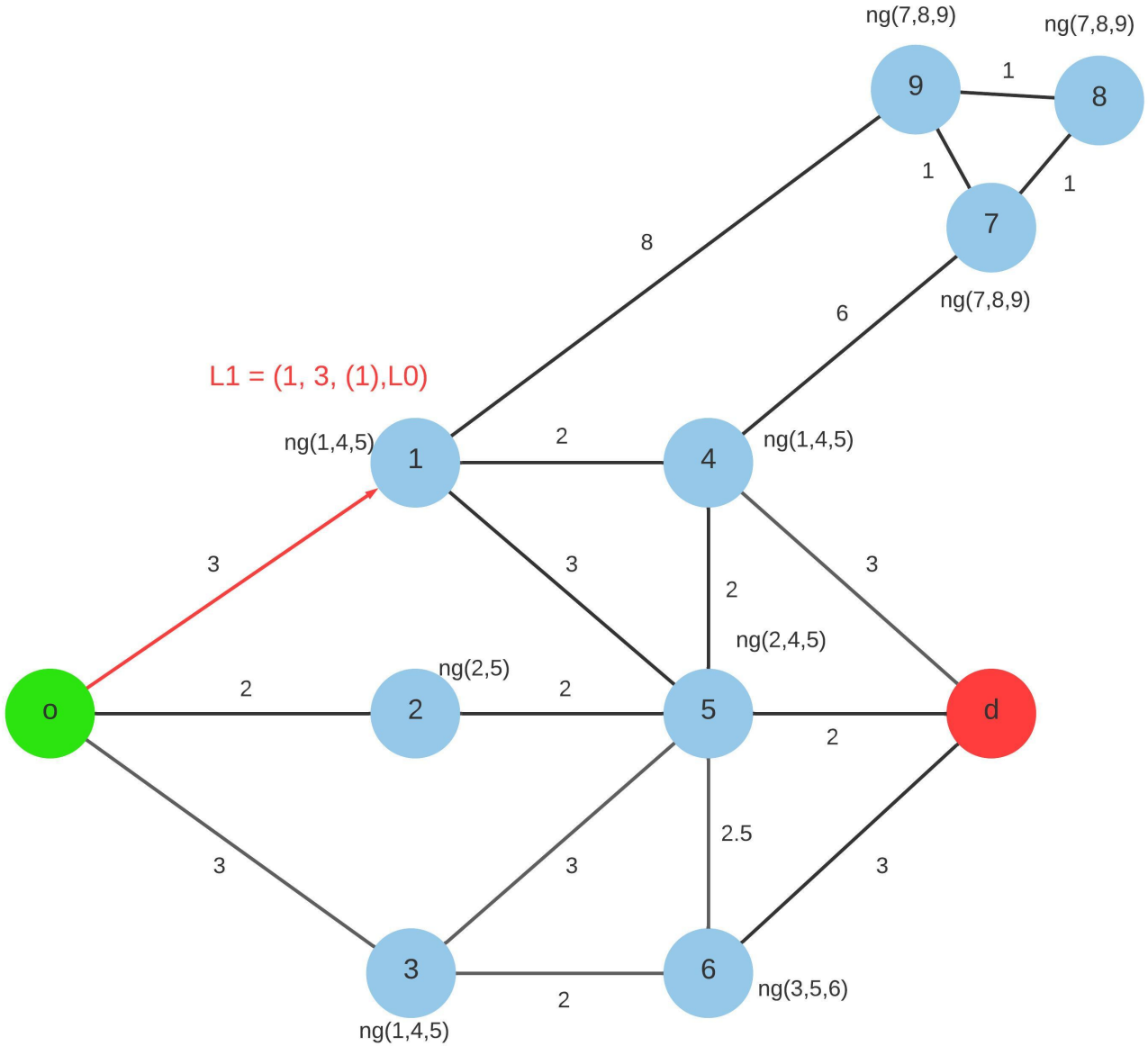
New label is:

$$\mathcal{L}_{new} = (j^n, c^n, f_T, f_Q, U^n, \mathcal{L}_{extend})$$

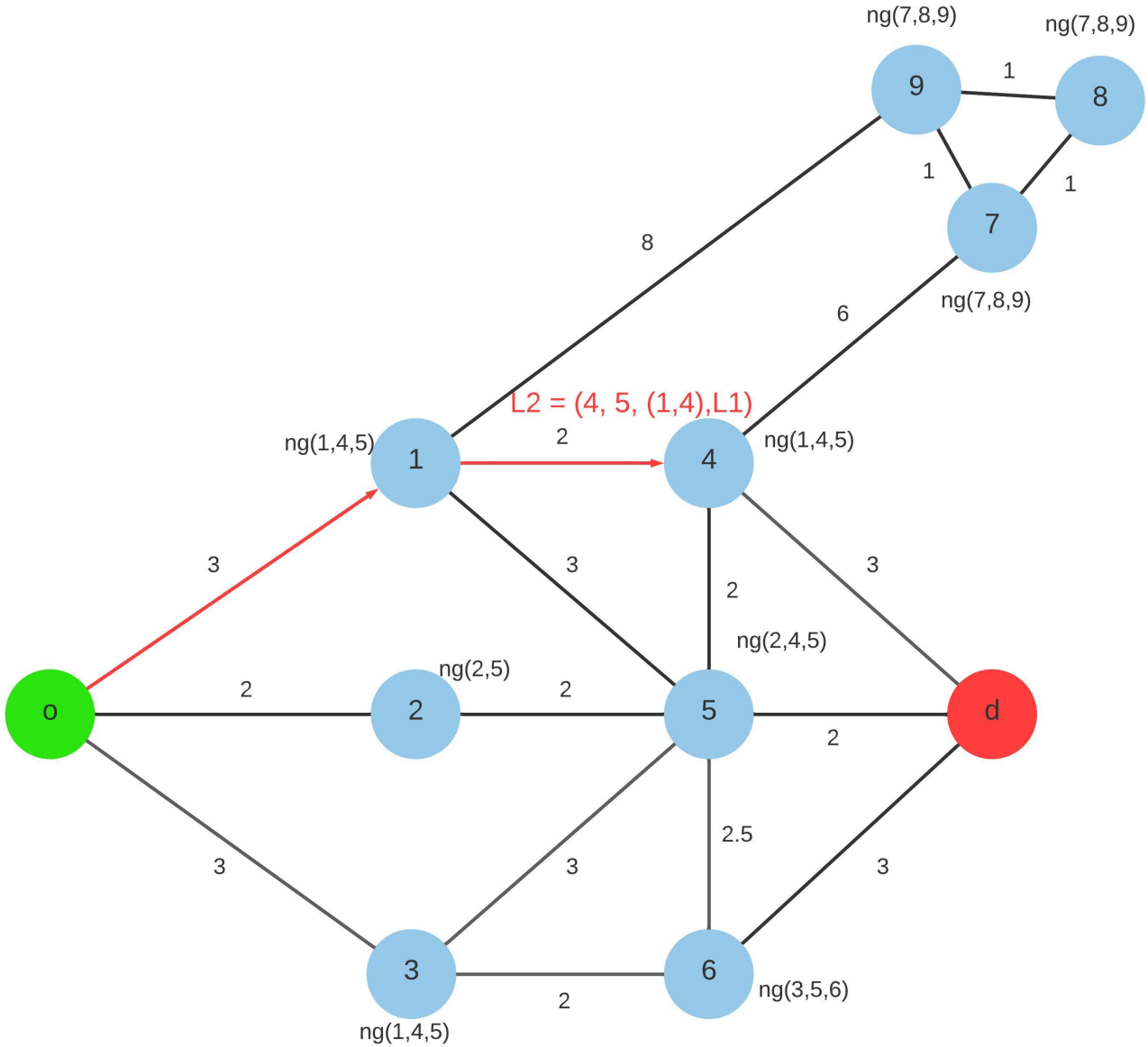
# Example:ng-route



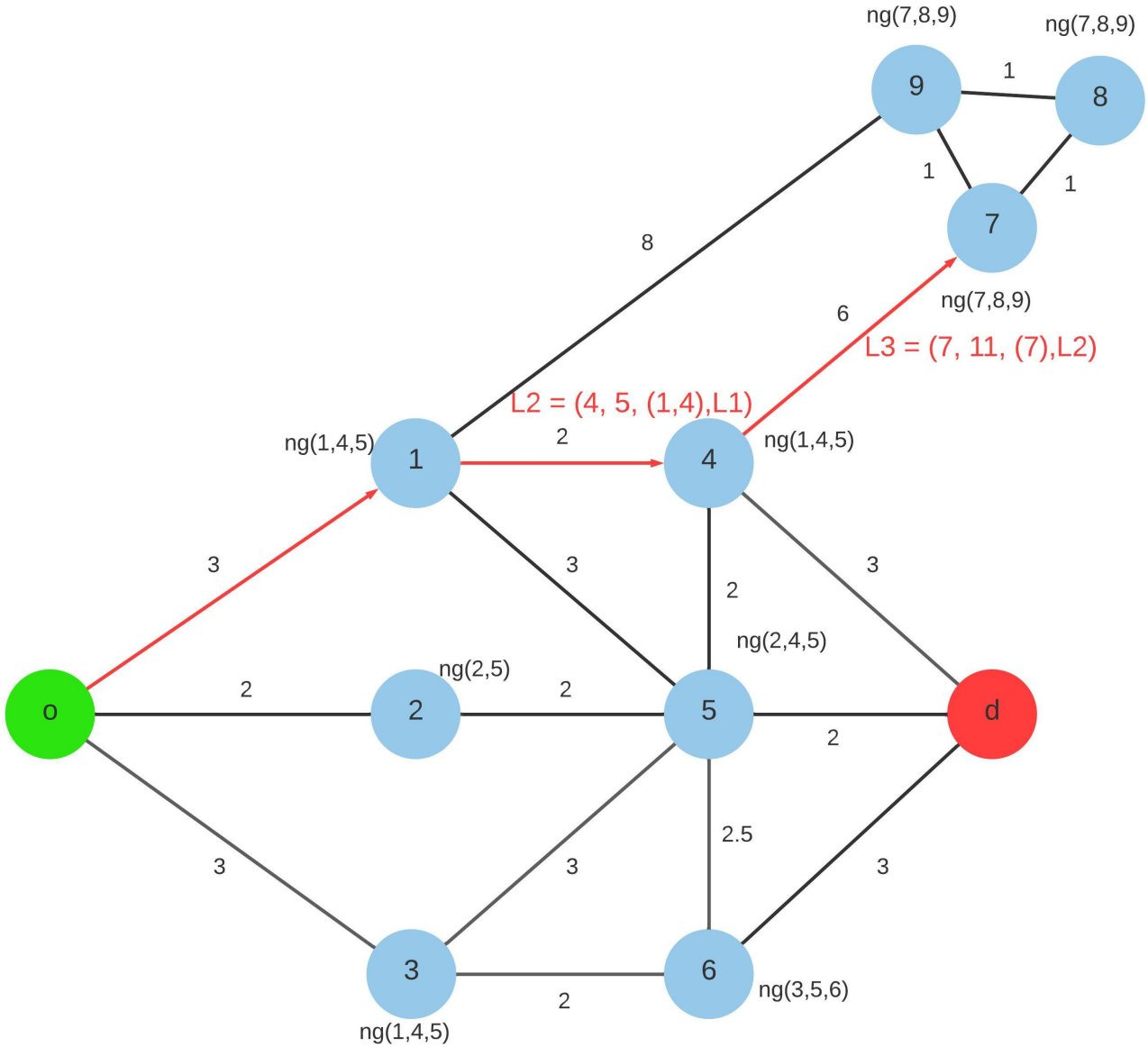
# Example:ng-route



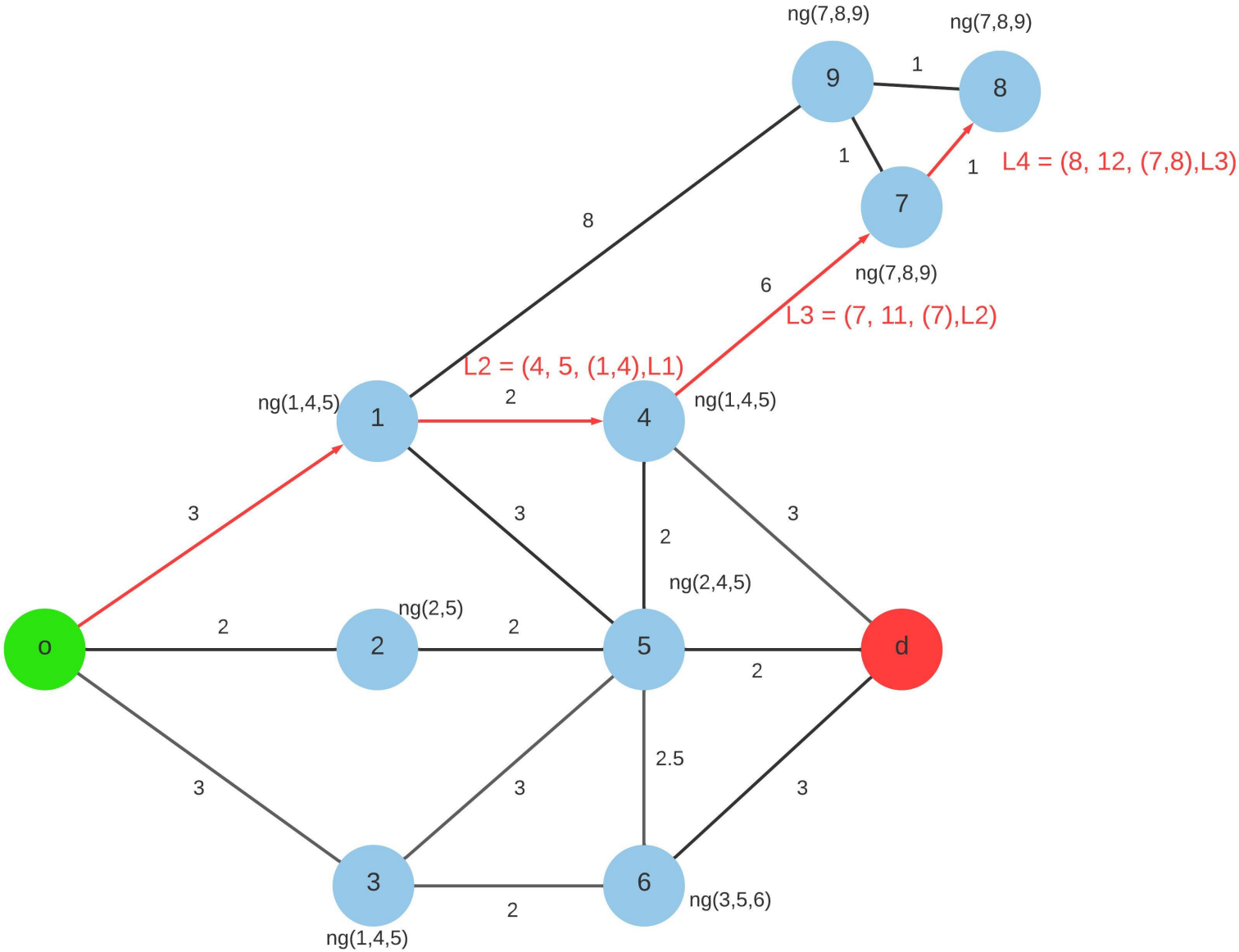
# Example:ng-route



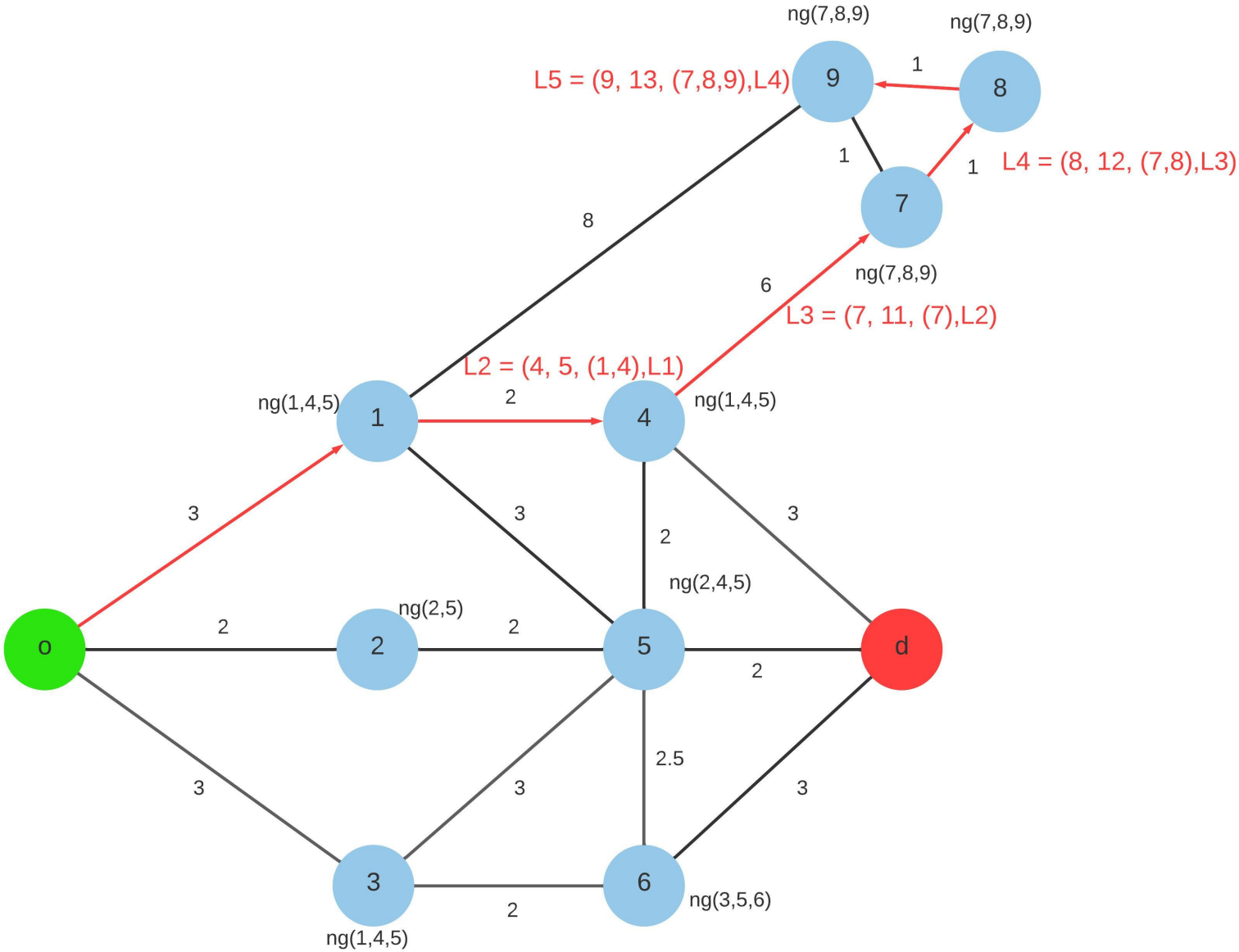
# Example:ng-route



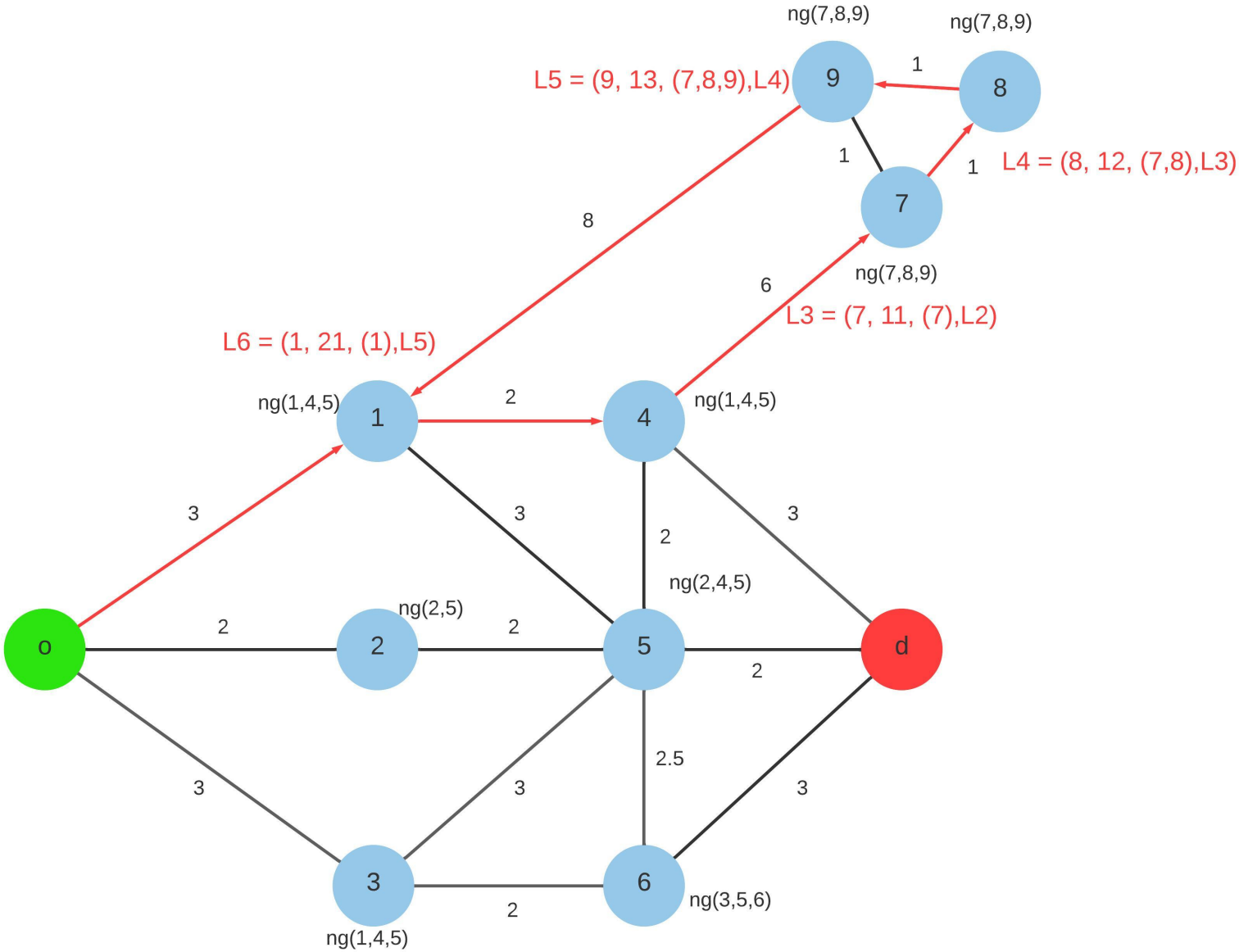
# Example:ng-route



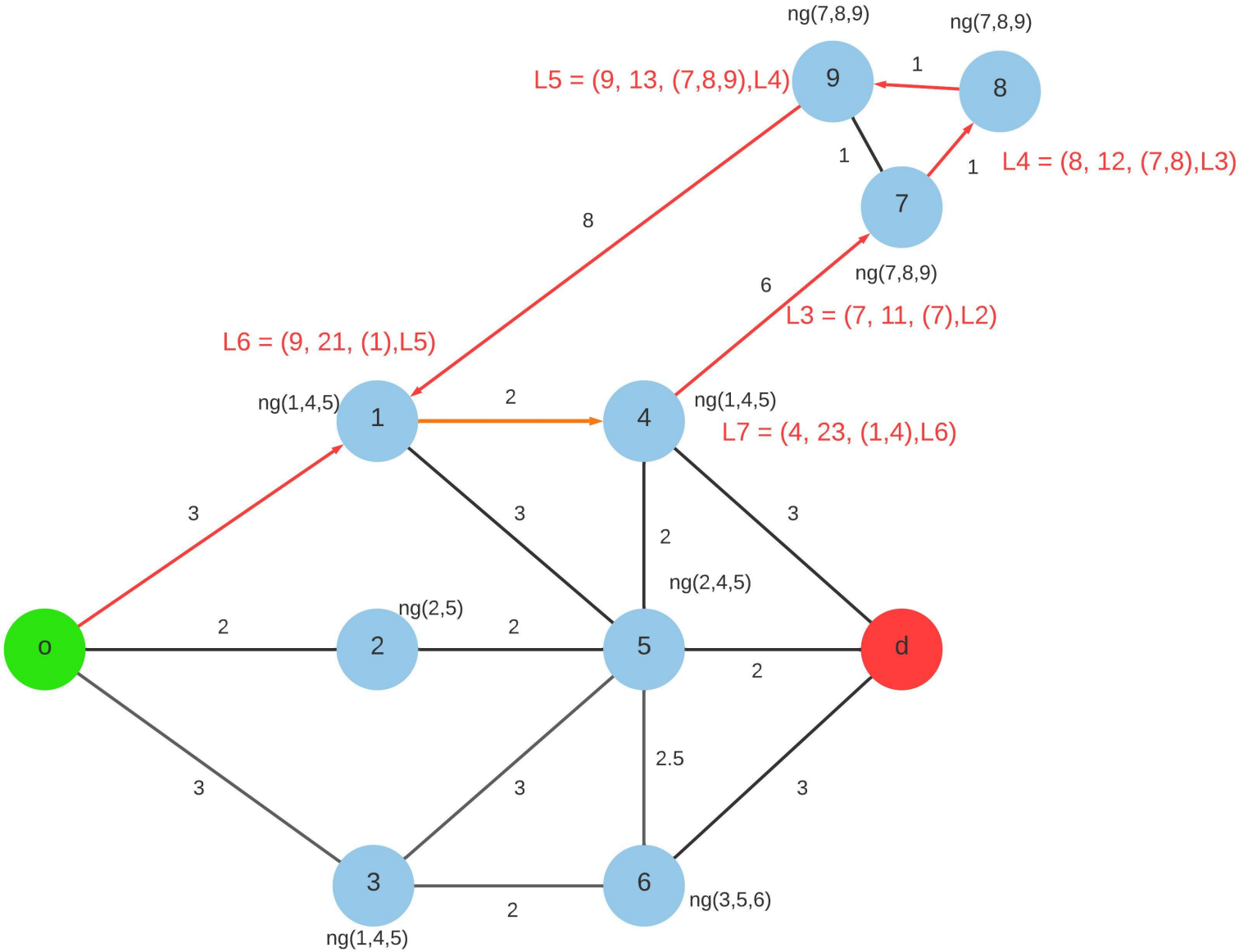
# Example:ng-route



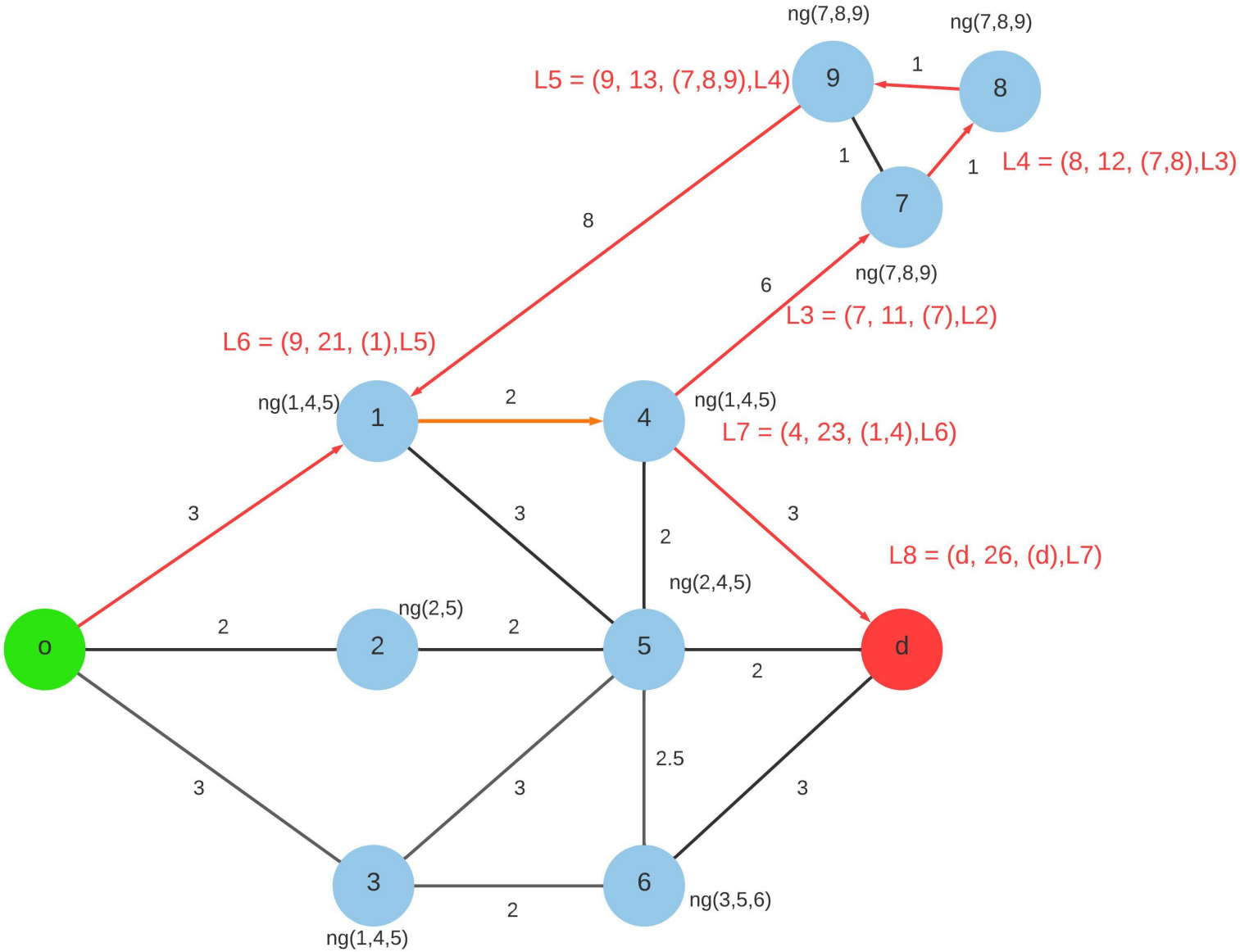
# Example:ng-route



# Example:ng-route



# Example:ng-route



# Feasibility checks

**Before we extend a label we must check all constraints.**

- $j^n \in U^e$, **Then** the extension is not feasible.
  **or**
- $T^e + t_{(i^e, j^n)} > b_{j^n}$, **Then** the extension is not feasible.
  **or**
- $Q^e + q_{j^n} > Q$, **Then** the extension is not feasible.

EPFL

# Dominance

We can use dominance rules to delete labels as before, however, dominance rules need to be modified.

**Dominance rules:**
Label $\mathcal{L}_1 = (i_1, c_1, T_1, Q_1, U_1, L_1)$ dominates label $\mathcal{L}_2 = (i_2, c_2, T_2, Q_2, U_2, L_2)$ if the following is true:

1. $i_1 = i_2$
2. $c_1 \leq c_2$
3. $T_1 \leq T_2$
4. $Q_1 \leq Q_2$
5. $U_1 \subseteq U_2$

EPFL

# Example: Dominance

$$\mathcal{L}_1 = (i_1 = 9, c_1 = -3, T_1 = 55, Q_1 = 44, U_1 = (1, 2, 3, 9), L_1)$$

$$\mathcal{L}_2 = (i_2 = 9, c_2 = -3, T_2 = 55, Q_2 = 44, U_2 = (1, 3, 9), L_2)$$

**Which label dominates?**

1. $i_1 = i_2 \Longrightarrow 9 = 9$
2. $c_1 \leq c_2 \Longrightarrow -3 \leq -3$
3. $T_1 \leq T_2 \Longrightarrow 55 \leq 55$
4. $Q_1 \leq Q_2 \Longrightarrow 44 \leq 44$
5. $U_1 \subseteq U_2 \Longrightarrow U_1 \not\subseteq U_2$ 2 is not in $U_2$

Rule (5) is violate and we cannot delete $\mathcal{L}_2$

# Example: Dominance

**The other way around**

$$\mathcal{L}_1 = (i_1 = 9, c_1 = -3, T_1 = 55, Q_1 = 44, U_1 = (1, 2, 3, 9), L_1)$$

$$\mathcal{L}_2 = (i_2 = 9, c_2 = -3, T_2 = 55, Q_2 = 44, U_2 = (1, 3, 9), L_2)$$

**Which label dominates?**

1. $i_2 = i_1 \Longrightarrow 9 = 9$
2. $c_2 \leq c_1 \Longrightarrow -3 \leq -3$
3. $T_2 \leq T_1 \Longrightarrow 55 \leq 55$
4. $Q_2 \leq Q_1 \Longrightarrow 44 \leq 44$
5. $U_2 \subseteq U_1 \Longrightarrow U_2 \subset U_1$ All elements are contained

All rules are satisfied label $\mathcal{L}_1$ is dominated by $\mathcal{L}_2$ and is deleted.

EPFL

# Summary

- SPP find the shortest path from the origin to the destination node in a graph
  - The Principle of optimality shows that any sub path of the shortest path must also be a shortest path.
  - The SPP can be solved with a labeling algorithm in a polynomial number of operations,
  - A SPP where the graph has a negative cost cycle is unbounded.
- SPPRC some graphs have resources that are used as a path passes through an arc.
  - Non negative consumption of resources bound the problem since the resources run out. The path cannot cycle on a negative cost cycle for ever.
  - Resource Extension Functions (REFs) extend the resources used when a path passes through an arc.
  - Feasibility check are necessary to not violate resource constraints.

EPFL

# Summary

- ESPPRC: Elememtarity constraints prevent cycles from forming in the shortest path.
  - Exponential growth of paths make the problem difficult to solve
  - The SPPRC is a relaxation of the ESPPRC that is produced when the elementarity constraints are dropped. However, the bound is weak.
  - k-cycle relaxations allow only cycles larger than k to be formed. The k-cycle relaxation imporves the bound of the SPPRC relaxation.
  - Ng-route relaxation only allows cycles with distant nodes to be formed and prevents cycles formed by nearest neighbors.