Network simplex algorithm

最小费用流问题的最优性条件2

Reduced Cost Optimality Condition

A feasible solution x^* is optimal if any only if there exist node potentials π satisfying

$$c^{\pi}_{ij} \geq 0 \text{ for every arc } (i,j) \text{ in residual network } G(x^*)$$

- Reason: from the negative cycle optimality condition

→If there exist node potential π leading to all $c^{\pi}_{ij} \geq 0$, then $G(x^*)$ has no negative cycle (in terms of both link reduced cost and link original cost). So x^* is optimal.

←If x^* is optimal, then $G(x^*)$ has no negative cycle (in terms of original link cost). Shortest path (in terms of original link cost) is well defined on $G(x^*)$ with link original cost.

Let $d(j)$ be the shortest path distance from node 1 to node j on $G(x^*)$. Due to shortest path optimality condition, we have $d(j) \leq d(i)+c_{ij}$ for all (i,j).

Rewrite it as $0 \leq c_{ij} -(-d(i))+(-d(j)) = c^{\pi}_{ij}$.

So we can find node potentials by $\pi(j)=-d(j)$ for all node j, which makes all reduced cost ≥0

最小费用流问题的最优性条件3

Complementary Slackness Optimality Condition

A feasible solution x^* is optimal if any only if for some set of node potentials π, the reduced costs and flow values satisfy the following complementary slackness optimality conditions for every arc $(i,j) \in A$:

If c_{ij}^{π}>0, then x_{ij}^{*}=0;

If $0 < x_{ij}^{*} < u_{ij}$, then c_{ij}^{π}=0;

If $c_{ij}^{\pi} < 0$, then x_{ij}^{*}=u_{ij}.

Proof. →

Case I: If c_{ij}^{π}>0, then the residual network can not contain the arc (j,i) because c_{ji}^{π}=- c_{ij}^{π} <0 , which contradicts the reduced cost optimality condition. So x_{ij}^{*}=0 such that $u_{ji} = 0$;

Case II: If $0 < x_{ij}^{*} < u_{ij}$, the residual network contains both (i,j) and (j,i). Then the reduced cost optimality condition holds only if c_{ij}^{π}=c_{ji}^{π}=0;

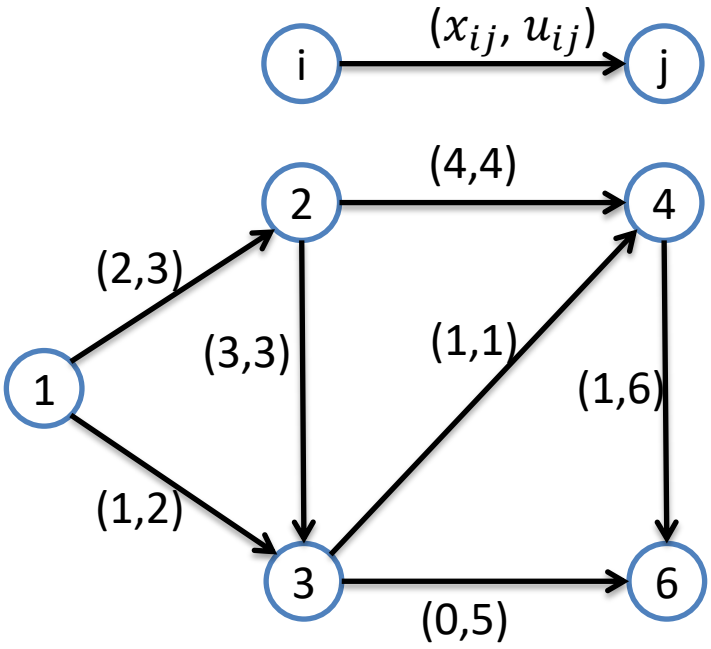
Case III: If $c_{ij}^{\pi} < 0$, the residual network can not contain the arc (i,j), so x_{ij}^{*}=u_{ij}.

Preliminaries

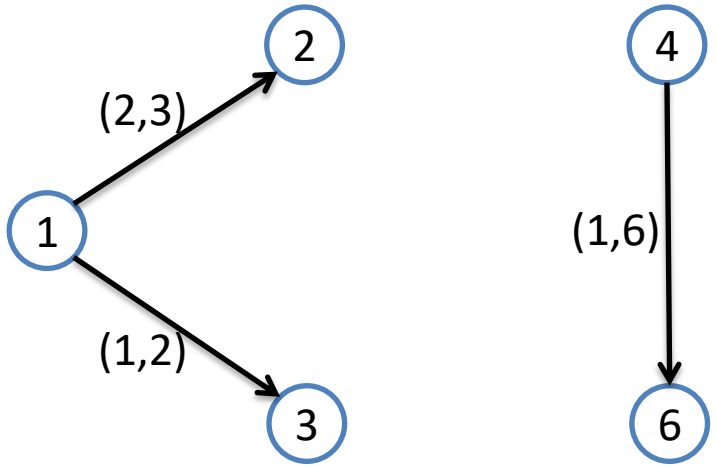
- For any feasible solution x, we say that an arc (i, j) is a **free arc** if $0 < x_{ij} < u_{ij}$, and is a **restricted arc** if x_{ij}=0 or x_{ij}=u_{ij}.
- We refer to a solution x as a **cycle free solution** if the network contains no cycle composed only of free arcs.

Cycle free property

- **Theorem (cycle free property).** If the objective function of a minimum cost flow problem is bounded from below over the feasible region, the problem always has an optimal cycle free solution.



Example network

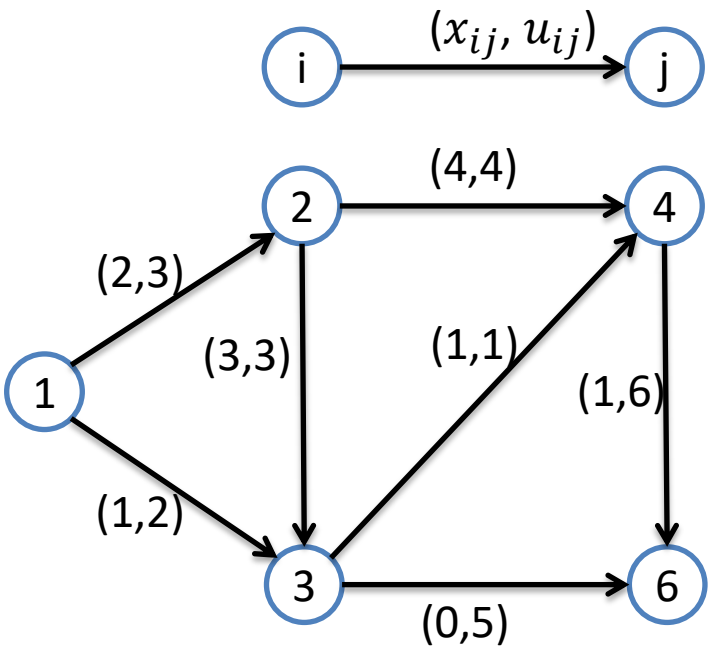


Set of free arcs

Construct a spanning tree solution based on a cycle free solution

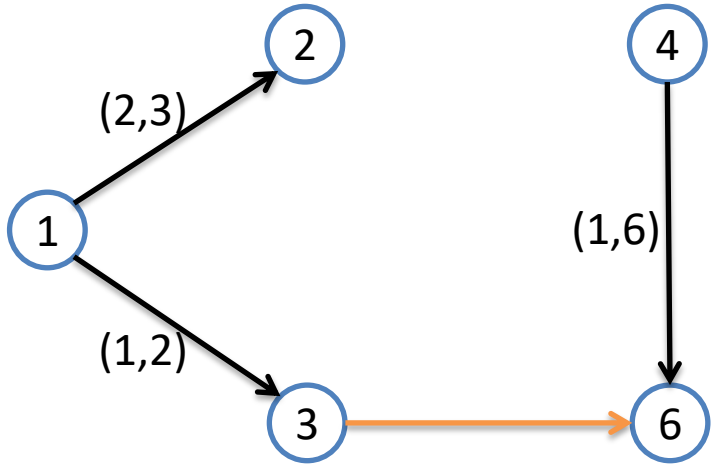
We refer to a feasible solution x and an associated spanning tree of the network as **a spanning tree solution** if every nontree arc is a restricted arc.

(In a spanning tree solution, the tree arcs can be free or restricted.)



Example network

Add some restricted arcs to produce a cycle free solution

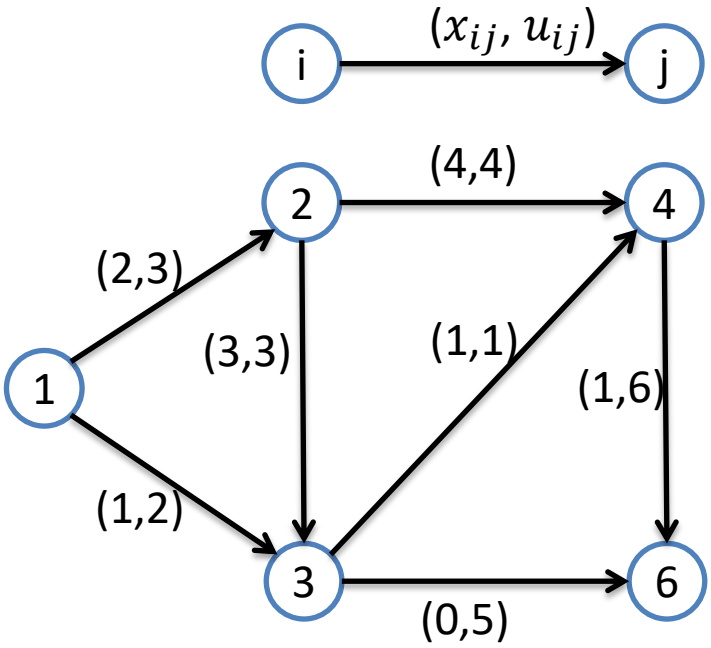


A spanning tree solution

A spanning tree structure

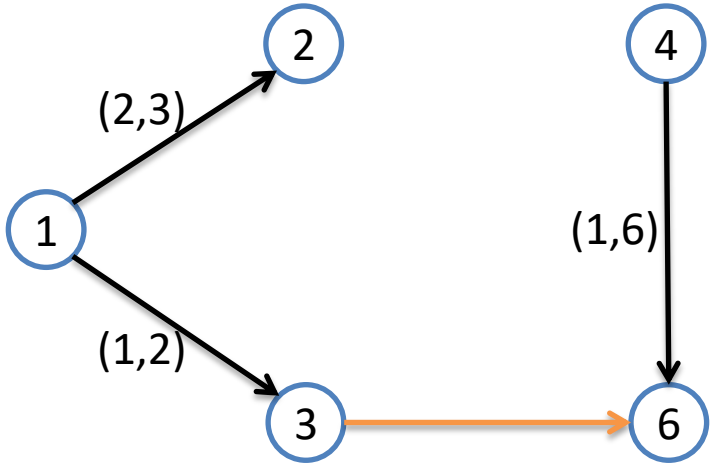
- A spanning tree solution partitions the arc set A into three subsets:
  - T, the arcs in the spanning tree
  - L, the nontree arcs whose flow is restricted to value zero
  - U, the nontree arcs whose flow is restricted in value to the arcs' flow capacities
- We refer to the triple (T,L,U) as a spanning tree structure.

A spanning tree structure



Example network

Add some restricted arcs to produce a cycle free solution



A spanning tree solution

T={(1,2),(2,3),(3,6),(4,6)}
L=∅
U={(2,3),(3,4,(2,4))}

Spanning tree property

- **Theorem (Spanning tree property).** If the objective function of a minimum cost flow problem is bounded from below over the feasible region, the problem always has an optimal spanning tree solution.

最小费用流问题的最优性条件4

Theorem. A spanning tree structure (T,L,U) is an optimal spanning tree structure of the minimum cost flow problem if it is feasible and for some choice of node potentials π, the arc reduced costs c_{ij}^{π} satisfy the following conditions:

(a) $c_{ij}^{\pi} = 0,\ for\ all\ (i,j) \in T;$

(b) $c_{ij}^{\pi} \geq 0,\ for\ all\ (i,j) \in L;$

(c) $c_{ij}^{\pi} \leq 0,\ for\ all\ (i,j) \in U.$

Network simplex算法的基本思想

- Network simplex算法的基本思想就是不断调整spanning tree，使得前述最优性条件能够满足。
  - 把条件b和c作为spanning tree调整的依据；
  - 在每一次调整spanning tree后，通过调整node potential来保证条件a始终满足；
  - 只要有nontree arc违反条件b或c，就把违反的边加入spanning tree，update node potential和flow，使新的node potential能保证条件a在新的spanning tree上被满足。

Network simplex algorithm

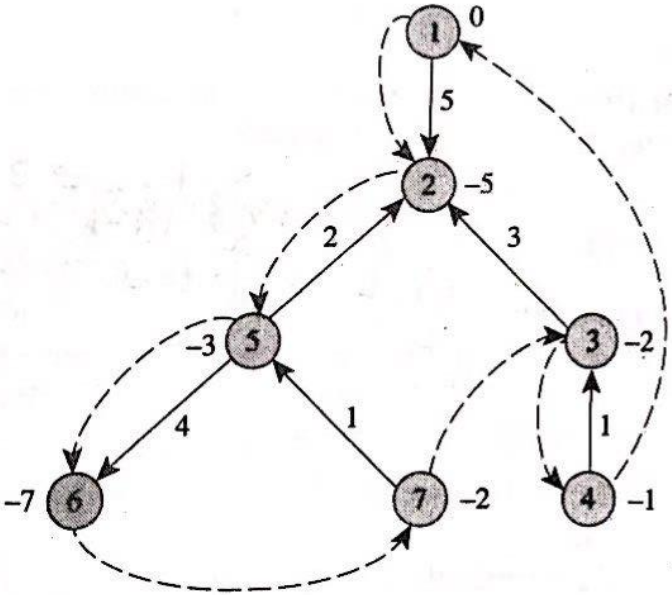
```
algorithm network simplex;
begin
    determine an initial feasible tree structure (T, L, U);
    let x be the flow and π be the node potentials associated with this tree structure;
    while some nontree arc violates the optimality conditions do
    begin
        select an entering arc (k, l) violating its optimality condition;
        add arc (k, l) to the tree and determine the leaving arc (p, q);
        perform a tree update and update the solutions x and π;
    end;
end;
```

Computing node potentials

- The network simplex algorithm moves from one spanning tree to the next, it always maintains the condition that the reduced cost of every arc (i,j) in the current spanning tree is zero (i.e, c_{ij}^{π}=0).
- Given the current spanning tree structure (T,L,U), how to determine the values for the node potentials that satisfy this condition for the tree arcs?
  - Set $\pi(1) = 0$, then calculate the remaining node potentials by traversing nodes using the thread indices, using the fact that the reduced cost of every spanning tree arc is zero:$c_{ij} - \pi(i) + \pi(j) = 0$ for every (i,j)$\in T$

```
procedure compute-potentials;
begin
    π(1) : = 0;
    j : = thread(1);
    while j ≠ 1 do
    begin
        i : = pred(j);
        If (i, j) ∈ A then π(j) : = π(i) − c_ij;
        If (j, i) ∈ A then π(j) : = π(i) + c_ji;
        j : = thread(j);
    end;
end;
```

Figure 11.4 Procedure *compute-potentials*.



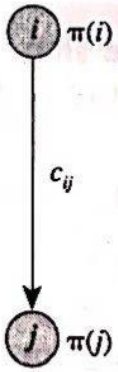
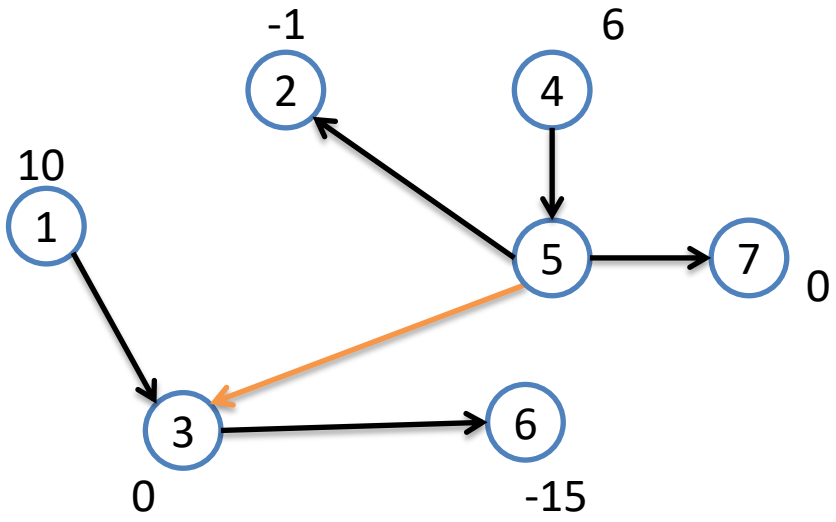


Figure 11.5 Computing node potentials for a spanning tree.

Computing arc flows

- Determining the flows on the tree arcs of a given spanning tree structure.
- If we delete a tree arc, say arc (i,j) from the spanning tree, then the tree decomposes into two subtrees. Let T_1 be the subtree containing node i and T_2 be the subtree containing node j.



T_1={1,3,6},
T_2={2,4,5,7}

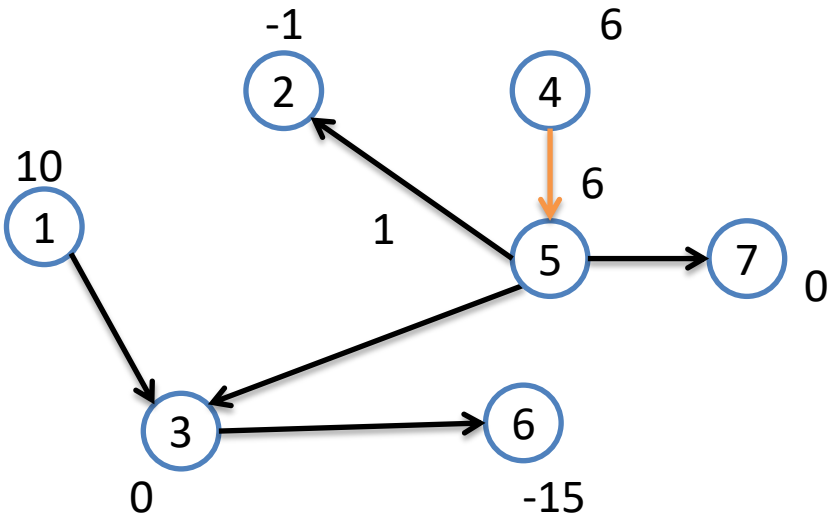
$x_{35} = 5$

The cumulative supply of nodes in T_1: $\sum_{k \in T_1} b(k)$

The cumulative supply of nodes in T_2: $\sum_{k \in T_2} b(k)$ (=$-\sum_{k \in T_1} b(k)$)

$$0 = x_{ij} - \sum_{k \in T_1} b(k) = x_{ij} + \sum_{k \in T_2} b(k)$$

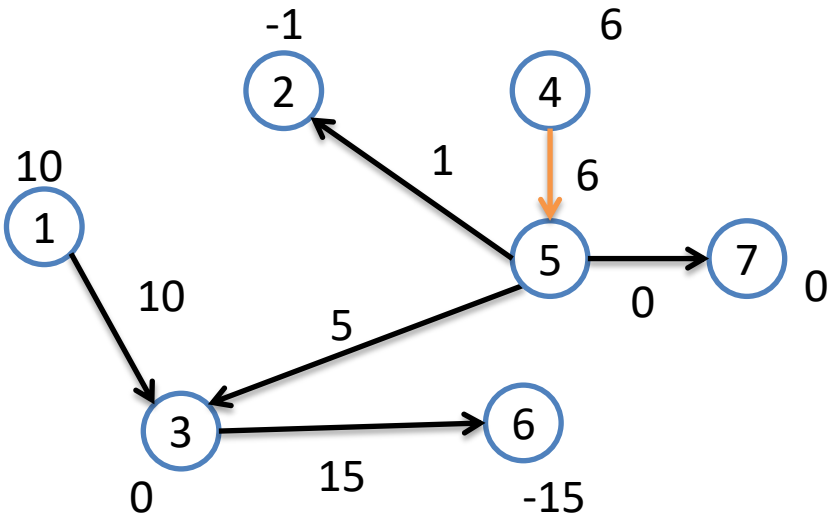
Computing arc flows



Suppose all non-tree arcs carry zero flow.

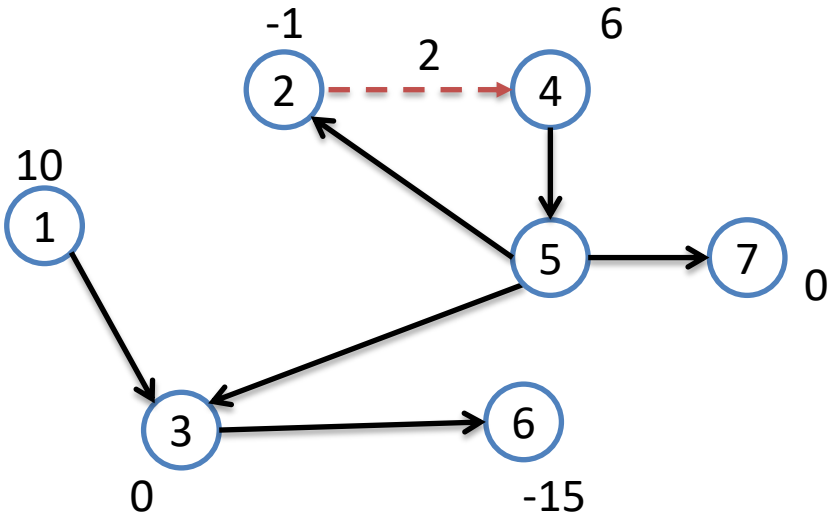
- If j is a leaf node, then we can easily determine the flow on the tree arc (i,j) or (j,i):
  - $x_{ji} = b(j)$
  - $x_{ij} = -b(j)$
- Setting the flow on this arc to this value has an effect on the mass balance of its incident nodes:
  - Reduce b(j) to zero, and transfer b(j) to b(i)

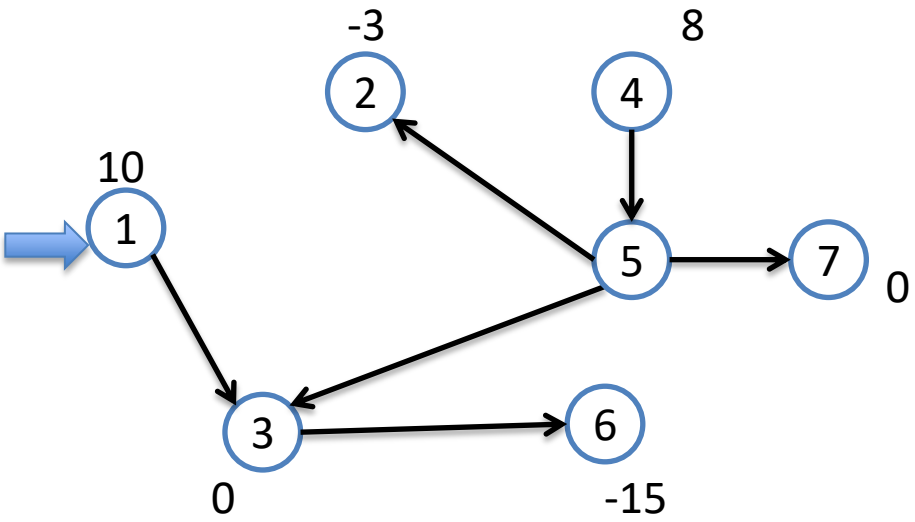
Example



Suppose all non-tree arcs carry zero flow.

Example





Suppose U={(2,4)}

For each (i,j) ∈ U , set $x_{ij} \coloneqq u_{ij}$, substract u_{ij} from b(i) and add u_{ij} to b(j);

Computing arc flows

- Based on the previous observation, we can devise the following method to compute the flows on all the tree arcs:

  For each (i,j)$\in L$, set $x_{ij} := 0$;

  For each (i,j)$\in U$, set $x_{ij} := u_{ij}$, substract u_{ij} from b(i) and add u_{ij} to b(j);

  T'=:T

  While T'$\neq \{1\}$ do

    select a leaf node j (other than node 1) in the subtree T';

    i:=pred(j);

    if (i,j) $\in$T' then x_{ij} :=-b(j)

    else x_{ji} :=-b(j)

    add b(j) to b(i)

    delete node j and the arc incident to it from T'

  end

Optimality testing and the entering arc

- To determine whether the spanning tree structure is optimal, we check to see whether the spanning tree structure satisfies the following conditions:

$$\text{(a) } c_{ij}^{\pi} \geq 0, \text{for all } (i,j) \in L;$$

$$\text{(b) } c_{ij}^{\pi} \leq 0, \text{for all } (i,j) \in U.$$

  - Any nontree arc that violates one of the above conditions are eligible to enter the tree.
  - Gantzig's pivot rule: when there are multiple arcs violating the above conditions, select the one with the maximum violation $\left| c_{ij}^{\pi} \right|$ to enter the tree.

Leaving arc

- After selecting arc (k,l) as the entering arc, the addition of this arc would create exactly one cycle W.
- Sending additional flow around the cycle W in the direction which leads to negative cycle cost
  - If (k,l)$\in L$, the orientation of the cycle should be the same as that of (k,l)
  - If (k,l) $\in U$, the orientation of the cycle should be the opposite of (k,l)

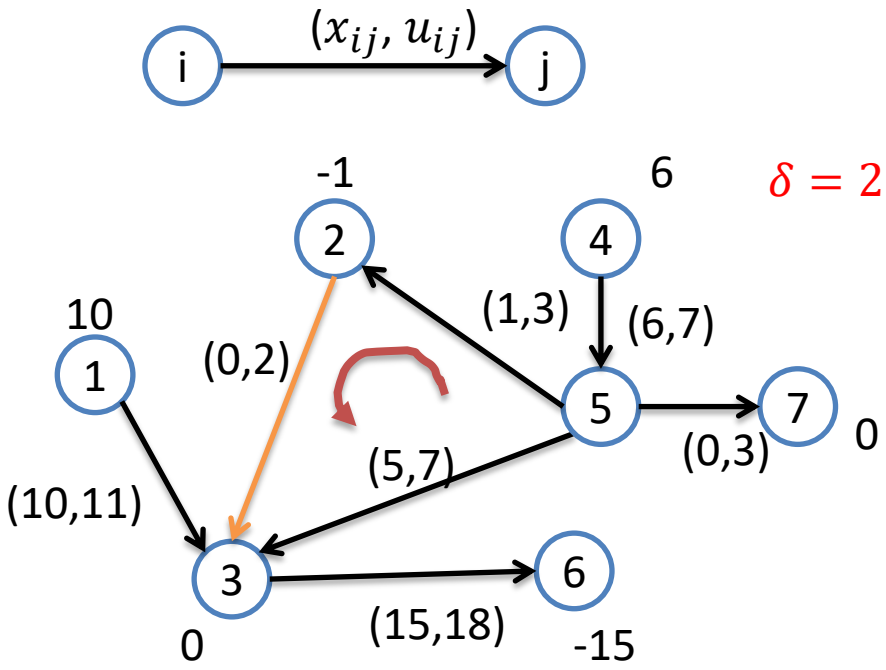
Why?

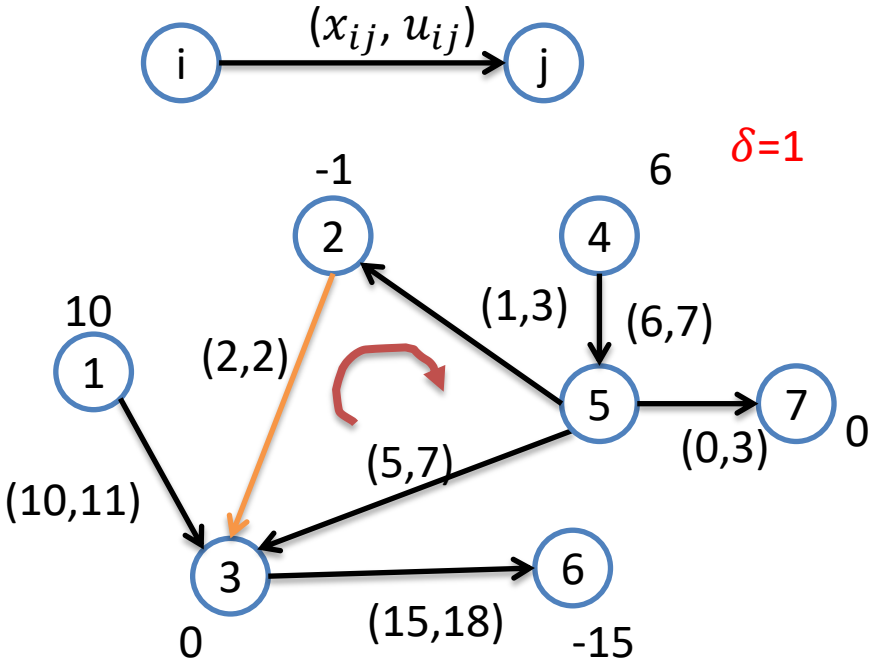
- Augment $\delta = \min\{\delta_{ij}, (i,j) \in W\}$ units of flow along W:

$$\delta_{ij} = \begin{cases} u_{ij} - x_{ij}, & if\ (i,j) \in \overline{W}\ (forward\ arcs) \\ x_{ij}, & if\ (i,j) \in \underline{W}\ (backward\ arcs) \end{cases}$$

- Select an arc (p,q) with $\delta_{pq} = \delta$ as the leaving arc

Suppose (2,3) is added to the spanning tree, then we have the following cycle:





If (2,3)∈ L, then we must have c_{23}^{π}<0 (otherwise (2,3) would not be chosen), such the cycle with the same orientation of (2,3) has a negative cost ($c_{35}^{\pi} = 0$, $c_{52}^{\pi} = 0$)..

If (2,3)∈ U, then we must have c_{23}^{π}>0, such the cycle with the same orientation of (3，2) has a negative cost .

After each iteration, the objective value reduce by $\delta \left| c_{ij}^{\pi} \right|$

Degenerate iterations

- We refer to the spanning tree as degenerate if $x_{ij} = 0$ or $x_{ij} = u_{ij}$ for some arc $(i, j) \in T$. In a non-degenerate spanning tree, $0 < x_{ij} < u_{ij}$ for every arc $(i, j) \in T$.

- We call an iteration degenerate if $\delta = 0$ and nondegenerate if $\delta > 0$。

- A degenerate iteration occurs only if T is a degenerate spanning tree.

Relationship to Simplex Method

$$\begin{aligned} &\text{Min } cx \\ \text{s.t.} \quad &Nx = b, \\ &0 \leq x \leq u \end{aligned}$$

The bounded simplex method for linear programming contains a basis structure (**B,L,U**) at every iteration and moves from one basis structure to another until it obtains an optimal basis structure. The set **B** is the set of basic variables, and the sets **L** and **U** are the nonbasic vairables at their lower and upper bounds.

Simplex method v.s. Network simplex method

Simplex method	Network simplex method
Given a basis structure (B,L,U), determining the associated basic feasible solution	Given a spanning tree structure (T,L,U), computing flows on tree arcs
Given a basis structure (B,L,U), determining the associated simplex multipliers π	Given a spanning tree structure (T,L,U), computing node potentials
Given a basis structure (B,L,U), check whether it is optimal, and if not, then determine an entering nonbasic variable	Given a spanning tree structure (T,L,U), check whether it is optimal, and if not, then determine an entering nontree arc