Minimum Cost Flow Problems II

AU4606: Network Optimization

Xiaoming Duan
Department of Automation
Shanghai Jiao Tong University

November 6, 2023

Last few lectures

- Minimum cost flow problems: important concepts
  - Residual graphs
  - Negative cost cycles
  - Optimality conditions
- Minimum cost flow problems: algorithms
  - Generic cycle canceling algorithms $O(m^2 nCU)$
  - Minimum mean cost cycle canceling algorithms
    - Weak polynomiality analysis: $O(m^2 n^2 \log(nC))$
    - Strong polynomiality analysis: $O(n^2 m^3 \log(n))$

Today

1. Simplex method

2. Network simplex method

Today

1. Simplex method

2. Network simplex method

Linear programming: problem setup

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} c_i x_i \\
\text{subject to} \quad & Ax = b \\
& 0 \le x_i \le u_i \quad \text{for each } i \in \{1, \dots, m\}.
\end{aligned}
$$

where $A = (a_{ij}) \in \mathbb{R}^{n \times m}$ and A has full row rank ($\text{rank}(A) = n$)

Simplex method: concepts

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} c_i x_i \\ \text{subject to} \quad & Ax = b \\ & 0 \le x_i \le u_i \quad \text{for each } i \in \{1, \dots, m\}. \end{aligned}$$

- Since A has full row rank, there are n linearly independent columns
- Partitioning $A = \begin{bmatrix} A_B & A_L & A_U \end{bmatrix}$ and similarly $x = \begin{bmatrix} x_B^\top & x_L^\top & x_U^\top \end{bmatrix}^\top$
- W.L.O.G., suppose $A_B \in \mathbb{R}^{n \times n}$ has full rank

Basic feasible solution

A solution $x = \begin{bmatrix} x_B^\top & x_L^\top & x_U^\top \end{bmatrix}^\top$ is a basic feasible solution if

1. $x_i = 0$ for $i \in L$
2. $x_i = u_i$ for $i \in U$
3. $x_B = A^{-1}b - A^{-1}x_U$, and $x_i \in [0, u_i]$ for $i \in B$

Simplex method: concepts

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} c_i x_i \\ \text{subject to} \quad & Ax = b \\ & 0 \leq x_i \leq u_i \quad \text{for each } i \in \{1, \ldots, m\}. \end{aligned}$$

- Partition $c = \begin{bmatrix} c_B^\top & c_L^\top & c_U^\top \end{bmatrix}^\top$
- Since A_B has full rank, we can select $\pi \in \mathbb{R}^n$ such that

$$c_B^\top = \pi^\top A_B$$

- The objective function can be rewritten as

$$(c_L^\top - \pi^\top A_L)x_L + (c_U^\top - \pi^\top A_U)x_U + \pi^\top b \quad \triangleq \quad \hat{c}_L x_L + \hat{c}_U x_U + \pi^\top b$$

Certify a basic feasible solution $x = \begin{bmatrix} x_B^\top & x_L^\top & x_U^\top \end{bmatrix}^\top$ is optimal where

1. $x_i = 0$ for $i \in L$
2. $x_i = u_i$ for $i \in U$
3. $x_B = b - A^{-1} x_U$, and $x_i \in [0, u_i]$ for $i \in B$

Optimality condition

Let x be a basic feasible solution and $\hat{c}_L x_L + \hat{c}_U x_U + \pi^\top b$ be the transformed objective function, if

1. $\hat{c}_i \geq 0$ for $i \in L$
2. $\hat{c}_i \leq 0$ for $i \in U$

then x is optimal.

Algorithm Simplex method

Find a basic feasible solution x
Compute the transformed objective $\hat{c}_L x_L + \hat{c}_U x_U + \pi^\top b$
while x does not satisfy the optimality condition **do**
    Pick a leaving variable $x_k = 0$ ($x_k = u_k$) such that $\hat{c}_k < 0$ ($\hat{c}_k > 0$)
    Increase (decrease) x_k so that some variable x_i for $i \in B$ reaches boundary
    Remove i from B and add k to B
    Update the basic feasible solution x and A_B, A_L, A_U
    Compute the transformed objective function
end while

Simplex method: procedure

Algorithm Simplex method

1: Find a basic feasible solution x
2: Compute the transformed objective $\hat{c}_L x_L + \hat{c}_U x_U + \pi^\top b$
3: **while** x does not satisfy the optimality condition **do**
4: Pick a leaving variable $x_k = 0$ ($x_k = u_k$) such that $\hat{c}_k < 0$ ($\hat{c}_k > 0$)
5: Increase (decrease) x_k so that some variable x_i for $i \in B$ reaches boundary
6: Remove i from B and add k to B
7: Update the basic feasible solution x and A_B, A_L, A_U
8: Compute the transformed objective function
9: **end while**

Issues beyond our discussion:

- Why does this work?

Simplex method: procedure

Algorithm Simplex method

```
Find a basic feasible solution x
Compute the transformed objective ĉ_L x_L + ĉ_U x_U + π^⊤ b
while x does not satisfy the optimality condition do
    Pick a leaving variable x_k = 0 (x_k = u_k) such that ĉ_k < 0 (ĉ_k > 0)
    Increase (decrease) x_k so that some variable x_i for i ∈ B reaches boundary
    Remove i from B and add k to B
    Update the basic feasible solution x and A_B, A_L, A_U
    Compute the transformed objective function
end while
```

Issues beyond our discussion:

- Why does this work?
- How to find a basic feasible solution (detect feasibility)?

Simplex method: procedure

Algorithm Simplex method

Find a basic feasible solution x
Compute the transformed objective $\hat{c}_L x_L + \hat{c}_U x_U + \pi^\top b$
while x does not satisfy the optimality condition **do**
    Pick a leaving variable $x_k = 0$ ($x_k = u_k$) such that $\hat{c}_k < 0$ ($\hat{c}_k > 0$)
    Increase (decrease) x_k so that some variable x_i for $i \in B$ reaches boundary
    Remove i from B and add k to B
    Update the basic feasible solution x and A_B, A_L, A_U
    Compute the transformed objective function
end while

Issues beyond our discussion:

- Why does this work?
- How to find a basic feasible solution (detect feasibility)?
- How to ensure that after each update we still have a basis matrix?

Simplex method: procedure

Algorithm Simplex method

Find a basic feasible solution x
Compute the transformed objective $\hat{c}_L x_L + \hat{c}_U x_U + \pi^\top b$
while x does not satisfy the optimality condition **do**
    Pick a leaving variable $x_k = 0$ ($x_k = u_k$) such that $\hat{c}_k < 0$ ($\hat{c}_k > 0$)
    Increase (decrease) x_k so that some variable x_i for $i \in B$ reaches boundary
    Remove i from B and add k to B
    Update the basic feasible solution x and A_B, A_L, A_U
    Compute the transformed objective function
end while

Issues beyond our discussion:

- Why does this work?
- How to find a basic feasible solution (detect feasibility)?
- How to ensure that after each update we still have a basis matrix?
- How to detect whether the objective function is lower bounded?

Simplex method: procedure

Algorithm Simplex method

Find a basic feasible solution x
Compute the transformed objective $\hat{c}_L x_L + \hat{c}_U x_U + \pi^\top b$
while x does not satisfy the optimality condition **do**
    Pick a leaving variable $x_k = 0$ ($x_k = u_k$) such that $\hat{c}_k < 0$ ($\hat{c}_k > 0$)
    Increase (decrease) x_k so that some variable x_i for $i \in B$ reaches boundary
    Remove i from B and add k to B
    Update the basic feasible solution x and A_B, A_L, A_U
    Compute the transformed objective function
end while

Issues beyond our discussion:

- Why does this work?
- How to find a basic feasible solution (detect feasibility)?
- How to ensure that after each update we still have a basis matrix?
- How to detect whether the objective function is lower bounded?
- How to certify the algorithm terminates?

Simplex method: procedure

Algorithm Simplex method

Find a basic feasible solution x
Compute the transformed objective $\hat{c}_L x_L + \hat{c}_U x_U + \pi^\top b$
while x does not satisfy the optimality condition **do**
    Pick a leaving variable $x_k = 0$ ($x_k = u_k$) such that $\hat{c}_k < 0$ ($\hat{c}_k > 0$)
    Increase (decrease) x_k so that some variable x_i for $i \in B$ reaches boundary
    Remove i from B and add k to B
    Update the basic feasible solution x and A_B, A_L, A_U
    Compute the transformed objective function
end while

Issues beyond our discussion:

- Why does this work?
- How to find a basic feasible solution (detect feasibility)?
- How to ensure that after each update we still have a basis matrix?
- How to detect whether the objective function is lower bounded?
- How to certify the algorithm terminates?
- How do we know it is possible to find a leaving variable?

Simplex method: an example

$$\begin{aligned} \text{minimize} \quad & -x_1 - 3x_2 \\ \text{subject to} \quad & 2x_1 + 3x_2 + x_3 = 6 \\ & -x_1 + x_2 + x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0. \end{aligned}$$

Today

Network simplex method: concepts

Free and restricted arcs

Given a feasible flow f

- An arc $(i, j) \in A$ is a free arc if $0 < f_{ij} < u_{ij}$
- An arc $(i, j) \in A$ is a restricted arc if $f_{ij} = 0$ or $f_{ij} = u_{ij}$

Cycle free flow

A feasible flow f is cycle free if it does not contain a cycle composed of only free arcs.

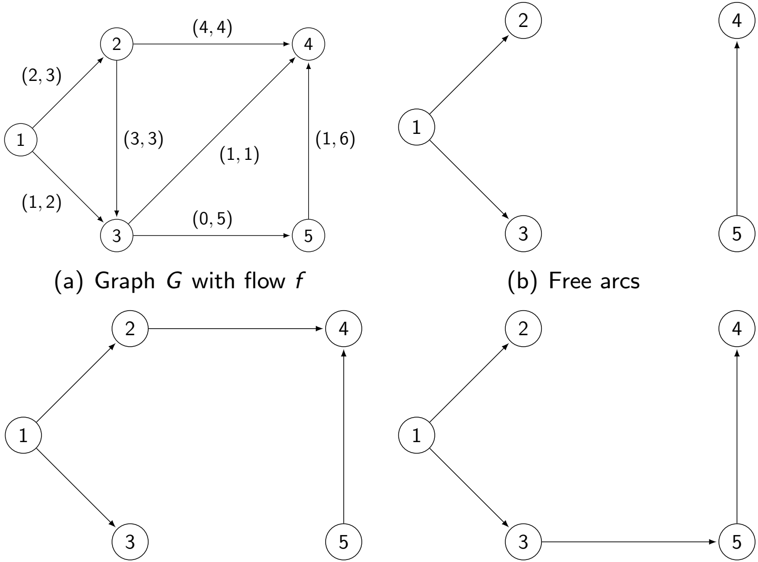
Cycle free property

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.

Spanning tree solution

Given a cycle free solution, a spanning tree T is a tree that contains all free arcs (and perhaps some restricted arcs).

Examples of spanning tree solutions



(a) Graph G with flow f

(b) Free arcs

(c) Spanning tree 1

(d) Spanning tree 2

Optimality condition

A spanning tree solution partitions the arcs into three disjoint sets

- $B = \{(i,j) \in A \,|\, (i,j) \text{ is a tree arc}\}$
- $L = \{(i,j) \in A \,|\, (i,j) \text{ is a nontree arc and } f_{ij} = 0\}$
- $U = \{(i,j) \in A \,|\, (i,j) \text{ is a nontree arc and } f_{ij} = u_{ij}\}$

Optimality condition

A spanning tree structure (B, L, U) is an optimal spanning tree structure of the minimum cost flow problem if it is feasible and for some node potentials p, the reduced costs satisfy the following conditions:

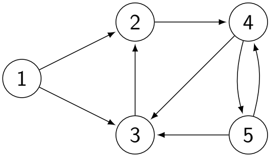
1. $c^p_{ij} = c_{ij} + p(i) - p(j) = 0$ for $(i,j) \in B$;
2. $c^p_{ij} \geq 0$ for $(i,j) \in L$ (arc in G(f));
3. $c^p_{ij} \leq 0$ for $(i,j) \in U$ (reverse arc in G(f)).

This is consistent with previous optimality conditions for min cost flow!

Algorithm Network simplex method

```
Compute a feasible flow f
Find a spanning tree structure and a node potential
while optimality not satisfied do
    Add a nontree arc violating optimality (entering arc) to T
    Cancel the cycle and determine a leaving arc
    Form a new spanning tree structure and compute the node potential
end while
```

Incidence matrix



	(1,2)	(1,3)	(2,4)	(3,2)	(4,3)	(4,5)	(5,3)	(5,4)
1	1	1	0	0	0	0	0	0
2	−1	0	1	−1	0	0	0	0
3	0	−1	0	1	−1	0	−1	0
4	0	0	−1	0	1	1	0	−1
5	0	0	0	0	0	−1	1	1

- Incidence matrix $H = \{h_{ij}\}$ of $G = (N, A)$ with n nodes and m arcs
  1. $H \in \mathbb{R}^{n \times m}$
  2. Each row corresponds to a node, each column corresponds to an arc
  3. $h_{ij} = 1$ if node i is the head of arc j (arc j has node i as head)
  4. $h_{ij} = -1$ if node i is the tail of arc j (arc j has node i as tail)
  5. Exactly one 1 and one -1 in each column

Flow Balance and Incidence matrix

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij} f_{ij} \\
\text{subject to} \quad & \sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b(i) \quad \text{for all } i \in \{1, \ldots, n\} \\
& 0 \leq f_{ij} \leq u_{ij} \quad \text{for each } (i,j) \in A.
\end{aligned}
$$

Flow Balance and Incidence matrix

$$\begin{aligned}\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij} f_{ij} \\ \text{subject to} \quad & \sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b(i) \quad \text{for all } i \in \{1,\ldots,n\} \\ & 0 \le f_{ij} \le u_{ij} \quad \text{for each } (i,j) \in A.\end{aligned}$$

Note that

$$\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b(i) \quad \text{for all } i \in \{1,\ldots,n\}$$

Flow Balance and Incidence matrix

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,j)\in A} c_{ij} f_{ij} \\ \text{subject to} \quad & \sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b(i) \quad \text{for all } i \in \{1,\ldots,n\} \\ & 0 \le f_{ij} \le u_{ij} \quad \text{for each } (i,j) \in A. \end{aligned}$$

Note that

$$\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b(i) \quad \text{for all } i \in \{1,\ldots,n\}$$

can be written as

$$Hf = b$$

Flow Balance and Incidence matrix

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in A} c_{ij} f_{ij} \\
\text{subject to} \quad & \sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b(i) \quad \text{for all } i \in \{1, \dots, n\} \\
& 0 \le f_{ij} \le u_{ij} \quad \text{for each } (i,j) \in A.
\end{aligned}$$

Note that

$$\sum_{j:(i,j)\in A} f_{ij} - \sum_{j:(j,i)\in A} f_{ji} = b(i) \quad \text{for all } i \in \{1, \dots, n\}$$

can be written as

$$Hf = b$$

Rank of incidence matrix

Let H be the incidence of a directed graph $G = (N, A)$. If G is connected, then $\text{rank}(H) = n - 1$.

Network simplex method: interpretation

Algorithm Network simplex method

Compute a feasible flow f
Find a spanning tree structure and a node potential
while optimality not satisfied **do**
    Add a nontree arc violating optimality (entering arc) to T
    Cancel the cycle and determine a leaving arc
    Form a new spanning tree structure and compute the node potential
end while

- Spanning tree structure: basis matrix
- Optimality condition: transformed objective function
- Adding nontree arc: finding an entering variable
- Cycle canceling: finding a leaving variable

Upcoming

Week 1-8 (AU4606 & AI4702):

- Introduction (1 lecture)
- Preparations (3 lectures)
  - basics of graph theory
  - algorithm complexity and data structure
  - graph search algorithm
- Shortest path problems (3 lectures)
- Maximum flow problems (5 lectures)
- Minimum cost flow problems (3 lectures)
- Introduction to multi-agent systems (1 lecture)
- Introduction to cloud networks (1 lecture)

Week 9-16 (AU4606):

- Simplex and network simplex methods (this lecture)
- Global minimum cut problems (3 lectures)
- Minimum spanning tree problems (3 lectures)