

# Advanced Operations Research Techniques

## IE316

### Lecture 18

Dr. Ted Ralphs

---

## Reading for This Lecture

- Bertsimas 7.3-7.5

## Tree Solutions

- From now on, we assume that  $\sum_{i \in N} b_i = 0$  and that  $G$  is connected.
- A flow vector  $f$  is called a *tree solution* if it can be constructed by the following procedure:
  - Pick a set of arcs  $T$  of  $n - 1$  arcs that form a tree when their direction is ignored.
  - Set  $f_{ij} = 0$  for every  $(i, j) \notin T$ .
  - Use the flow balance equations  $\tilde{A}f = \tilde{b}$  to determine the values of the flow variables  $f_{ij}, (i, j) \in T$ .
- Note that the flow balance equations always have a unique solution.
- A tree solution that also satisfies  $f \geq 0$  is called a *feasible tree solution*.

**Theorem 1.** *A flow vector is a basic solution if and only if it is a tree solution.*

## Network Simplex Method

- We now introduce a simple version of the simplex method for solving network flow problems.
- We have already seen what basic solutions look like.
- How do we change the basis?
  - Choose a nonbasic variable—this is an arc not in  $T$ .
  - Adding this arc to  $T$  forms a cycle.
  - To increase flow on the new arc, push  $\theta$  units of flow around the cycle.
  - Let  $F$  be the set of forward arcs and  $B$  be the set of backward arcs. Then the new flow is

$$\hat{f}_{kl} = \begin{cases} f_{kl} + \theta, & \text{if } (k, l) \in F, \\ f_{kl} - \theta, & \text{if } (k, l) \in B, \\ f_{kl}, & \text{otherwise.} \end{cases}$$

- The maximum value for  $\theta$  is  $\theta^* = \min_{(k,l) \in B} f_{kl}$ .

## Reduced Costs in the Network Simplex Method

- Given the current dual vector  $p$  and defining  $p_n = 0$  for convenience, we can easily derive that  $\bar{c}_{ij} = c_{ij} - (p_i - p_j)$ .
- To calculate the values of the dual variables, simply set  $p_n = 0$  and solve the system  $p_i - p_j = c_{ij}$  for every  $(i, j) \in T$ .
- This can be done easily.
- Alternatively, the reduced cost of a nonbasic arc  $(i, j)$  is the sum of the costs of the arcs forming a cycle with  $(i, j)$  in the current tree solution (why?).

## Overview of the Network Simplex Method

1. Start with a **basic feasible solution** associated with a tree  $T$ .
2. Compute the **reduced costs** for all arcs  $(i, j) \notin T$ .
3. If all the reduced costs are nonnegative, the current solution is **optimal**.
4. Otherwise, choose an arc with **negative reduced cost** to enter the basis.
5. If all the arcs in the resulting cycle are forward arcs, then the problem is **unbounded**.
6. Otherwise, push  $\theta^* = \min_{(k,l) \in B} f_{kl}$  units of flow around the cycle.
7. Remove one of the arcs whose flow becomes zero from the basis. Iterate.

## Integrality of Solutions

**Theorem 2.** Consider an *uncapacitated network flow problem* in which the underlying graph is connected.

1. For every basis matrix  $B$ , the matrix  $B^{-1}$  has integer entries.
2. If the supplies  $b_i$  are integer, then every basic solution has integer entries.
3. If the cost coefficients  $c_{ij}$  are integer, then every dual basic solution has integer entries.

Proof:

**Corollary 1.** If the supplies  $b_i$  are integers and the optimal cost is finite, then there exists an optimal flow that is integer.

## Network Simplex with General Upper and Lower Bounds

- We can easily derive a version of network simplex with **upper and lower bounds**.
- We have two sets of nonbasic arcs—those at lower bound and those at upper bound.
- In each iteration, we look for
  - an arc at its **lower bound** whose reduced cost is **negative**, or
  - an arc at its **upper bound** whose reduced cost is **positive**.
- In the first case, we “**push flow around the cycle**” created in the same direction as the arc.
- In the second case, we push flow around in the **opposite direction** (thereby reducing flow on the arc).
- We have to change our definition of  $\theta^*$  to account for upper bounds.

$$\theta^* = \min\left\{ \min_{(i,j) \in B} (f_{ij} - d_{ij}), \min_{(i,j) \in F} (u_{ij} - f_{ij}) \right\}$$



## Negative Cost Cycle Algorithm

- For the rest of the lecture, we will assume that the lower bound on each arc is zero (for convenience).
- In network simplex, the **sum of the reduced costs around any cycle** is equal to the sum of the original costs (with the backward arc costs negated).
- The cycle created by an arc with negative reduced cost is hence a “**negative cost cycle**” in the original graph.
- Because of degeneracy, it is not guaranteed that we can push flow around this cycle.
- We now discuss an algorithm that is guaranteed to decrease the objective function at each iteration.
- The idea is to locate a “**negative cost cycle**” around which flow can be pushed.

## Pushing Flow Around Cycles

- Consider a cycle  $C$  and the **simple circulation** associated with it.

$$h_{ij}^C = \begin{cases} 1, & \text{if } (i, j) \in F, \\ -1, & \text{if } (i, j) \in B, \\ 0, & \text{otherwise.} \end{cases}$$

- If  $f$  is a feasible flow,  $\delta$  is a scalar and we change the flow to  $f + \delta h^C$ , we say we are **pushing  $\delta$  units of flow around  $C$** .
- Notice that the new flow is also feasible, as long as

$$0 \leq f_{ij} + \delta h_{ij}^C \leq u_{ij}.$$

- This is equivalent to

$$\delta \leq \delta(C) = \min\left\{ \min_{(i,j) \in B} f_{ij}, \min_{(i,j) \in F} (u_{ij} - f_{ij}) \right\}$$

## Negative Cost Cycles

- The cost change per unit of flow pushed around cycle  $C$  is called the *cost of a cycle  $C$*  and is

$$c^T h^C = \sum_{(i,j) \in F} c_{ij} - \sum_{(i,j) \in B} c_{ij}$$

- This gives *negative cost cycle* a more formal definition.
- If there is a negative cost cycle for which  $\delta(C) = \infty$ , then the problem is unbounded.
- Any cycle for which  $\delta(C) > 0$  is called *unsaturated*.
- We are interested in finding *unsaturated cycles with negative cost*.
- The basic algorithm is to start with a feasible flow and then iteratively *saturate* all unsaturated cycles with negative cost.

## Questions to be Answered

- How do we start the algorithm (find a feasible flow)?
- How do we find unsaturated, negative cost cycles?
- If there are no unsaturated negative cost cycles, is the current solution optimal?
- Is the algorithm guaranteed to terminate?