Graphs and Network Flows
IE411

Lecture 20

Dr. Ted Ralphs
Network Simplex Algorithm

**Input:** A network $G = (N, A)$, a vector of capacities $u \in \mathbb{Z}^A$, a vector of costs $c \in \mathbb{Z}^A$, and a vector of supplies $b \in \mathbb{Z}^N$

**Output:** $x$ represents a minimum cost network flow

Determine an initial feasible tree structure $(T, L, U)$

Let $x$ be flow and $\pi$ be node potentials associated with $(T, L, U)$

**while** Some non-tree arc violates the optimality conditions **do**

Select an entering arc $(k, l)$ violating optimality conditions

Add arc $(k, l)$ to tree and determine leaving arc $(p, q)$

Perform a tree update and update solutions $x$ and $\pi$

**end while**
Example

\begin{center}
\begin{tikzpicture}
    \node[draw, circle] (i) at (0,0) {i} ;
    \node[draw, circle] (j) at (2,0) {j} ;
    \draw[->] (i) -- node[above]{{(c,x,u)}} (j) ;
\end{tikzpicture}
\end{center}

\begin{center}
\begin{tikzpicture}
    \node[draw, circle] (1) at (-3,-3) {1} ;
    \node[draw, circle] (2) at (-1,-1) {2} ;
    \node[draw, circle] (3) at (0,-3) {3} ;
    \node[draw, circle] (4) at (1,-1) {4} ;
    \node[draw, circle] (5) at (2,-3) {5} ;
    \node[draw, circle] (6) at (3,-1) {6} ;
    \draw[->] (1) -- node[left, near start]{{(20,0)}} (2) ;
    \draw[->] (1) -- node[above, near start]{{(7,0,20)}} (3) ;
    \draw[->] (2) -- node[above]{{(5,5,15)}} (4) ;
    \draw[->] (2) -- node[above]{{(0,-4)}} (6) ;
    \draw[->] (3) -- node[below]{{(0,-4)}} (5) ;
    \draw[->] (3) -- node[above]{{(4,5,15)}} (1) ;
    \draw[->] (4) -- node[below]{{(6,5,10)}} (5) ;
    \draw[->] (4) -- node[above, near end]{{(0,-10)}} (6) ;
    \draw[->] (5) -- node[below, near end]{{(0,-10)}} (1) ;
\end{tikzpicture}
\end{center}
Degeneracy in Network Simplex

• Network simplex does not necessarily terminate in a finite number of iterations

• Poor choices of entering and leaving arcs lead to cycling

• Maintaining a strongly feasible spanning tree guarantees finite termination and speeds up the running time

• A pivot iteration is non-degenerate if $\delta > 0$ and is degenerate if $\delta = 0$

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

• If two arcs tie while determining the value of $\delta$, the next spanning tree will be degenerate.
Strongly Feasible Spanning Trees

Let \((T, L, U)\) be a spanning tree structure for a MCFP with integral data. A spanning tree \(T\) is strongly feasible if

- every tree arc with zero flow is upward pointing (toward root) and every tree arc with flow equal to capacity is downward pointing (away from root)
- we can send a positive amount of flow from any node to the root along the tree path without violating any flow bound.

These two definitions are equivalent. Proof?
Modifications to Network Simplex Algorithm

• Initial Strongly Feasible Spanning Tree
  – Does our construction algorithm work?
    * A non-degenerate spanning tree is always strongly feasible.
    * A degenerate spanning tree is sometimes strongly feasible.

• Leaving Arc Rule
  – Select the leaving arc as the last blocking arc encountered in traversing the pivot cycle $W$ along its orientation starting at the apex $w$.
  – Proof: Show that next spanning tree is strongly feasible.
Termination

• Each non-degenerate pivot strictly decreases objective function, so number of non-degenerate pivots is finite.

• To show: The pivot rule maintains the invariant that each spanning tree solution is strongly feasible.
  - Consider $W_2$, the part of the cycle from $p$ to apex: no arc can be blocking by pivot rule.
  - Consider $W_1$, the part of the cycle from apex to $q$:
    * If pivot is non-degenerate, then must be able to send flow backwards to root.
    * If pivot is degenerate, then $(p, q)$ must be contained in the part of the cycle from apex to $k$. Since the previous tree was strongly feasible and flows don’t change, we must still be able to send positive flow back along $W_1$.

• Note that each degenerate pivot must decrease the sum of the node potentials, so the number of denegerate pivots in between each successive non-degenrate pivot must also be finite.
Network Simplex and Simplex for LP

- Network simplex is an implementation of the simplex method for general LPs with upper and lower bounds on the variables.
- Tree solutions correspond to basic solutions in the simplex method.
- To see this, recall from the homework that a directed graph is acyclic if and only if its arc-node incidence matrix is lower triangularizable.
- The number of linearly independent constraints in our formulation of the MCFP is \( n - 1 \).
- Any basis matrix thus consists of \( n - 1 \) linearly independent columns.
- It is easy to show that such a basis matrix must have all 1’s on the diagonal and must be a tree.
Network Simplex and Simplex for LP (cont.)

- The node potentials are the dual values from the LP and reduced costs are the reduced costs of the arcs.

- Each iteration of network simplex corresponds to a pivot operation in general simplex.
  - Find a nonbasic (nontree) variable (arc) with negative reduced cost fixed at its lower or positive reduced cost fixed at its upper bound.
  - Increase the value of this variable until one of the basic variables hits its bound.
  - Remove the blocking variable from the basis.

- Because of the special form of the problem, we do not need to maintain the basis inverse explicitly.
Dual Network Simplex

• As in general simplex, there is a dual version of the algorithm.

• In this version, we maintain optimality conditions, while trying to achieve feasibility.

• We start with a (possibly infeasible) solution that satisfies optimality conditions and choose a tree arc whose flow violates its bounds.

• This arc is the leaving arc.

• We want to push flow around some cycle until the arc reaches its bound.

• The entering arc is the one with the “correct” orientation that has the smallest reduced cost (absolute value).

• There is a finite version of this algorithm that uses a perturbation technique similar to that used in general simplex.
Polynomial Algorithms for MCFPs

- As with the maximum flow problem, we can use scaling to reduce the dependence of running time on $U$ and $C$.
- By scaling the capacities, we can get a running time of $O(m \log US(n, m, nC))$.
- By scaling the costs, we can get a running time of $O(n^2 m \log(nC))$.
- By scaling both, we get a running time of $O(nm \log U \log nC)$.
- The minimum mean cycle-canceling algorithm has a strongly polynomial running time of $O(n^2 m^3 \log n)$ (or $O(n^2 n^2 \log nC)$).
Sensitivity Analysis

• Determine changes in optimal solution resulting from changes in data
  – arc cost
  – supply/demand
  – arc capacity

• Assuming spanning tree structure remains unchanged, if change in data affects
  – optimality $\rightarrow$ perform primal pivots to achieve optimality
  – feasibility $\rightarrow$ perform dual pivots to achieve feasibility
Cost Sensitivity Analysis

Suppose the cost of arc \((p, q)\) increases by \(\lambda\) units.

Case 1 \((p, q)\) is a non-tree arc

Case 2 \((p, q)\) is a tree arc
Supply/Demand Sensitivity

- Suppose supply/demand $b(k)$ of node $k$ increases by $\lambda$ units. Then, the supply/demand $b(l)$ of some node $l$ decreases by $\lambda$ units.

- From the mass balance constraints, we know that we must ship $\lambda$ units of flow from node $k$ to node $l$.

- Let $P$ be the unique tree path from node $k$ to node $l$. And let $\delta = \min\{\delta_{ij} : (i, j) \in P\}$.

- If $\lambda \leq \delta$, then $\cdots$

- If $\lambda > \delta$, then $\cdots$
 Capacity Sensitivity Analysis

• Suppose capacity of \((p, q)\) increases by \(\lambda\) units.

• What do we know about previous optimal solution?

• If \((p, q)\) is a tree arc or a non-tree arc at its lower bound

• If \((p, q)\) is a non-tree arc at its upper bound