15.093 Optimization Methods

Lecture 11: Network Optimization

The Network Simplex Algorithm

1 Network Optimization

1.1 Why do we care?

SLIDE 1

- Networks and associated optimization problems constitute reoccurring structures in many real-world applications.
- The network structure often leads to additional insight and improved understanding.
- Given integer data, the standard models have integer optimal solutions.
- The network structure also enables us to design more efficient algorithms.

1.2 A Comparison

1.2.1 Running Times

SLIDE 2

Algorithm	Running Time (sec)	# Iterations
Standard Simplex	334.59	42759
Network Simplex	7.37	23306
Ratio	2.2 %	54 %

Average over 5 random instances with $10,000$ nodes and $25,000$ arcs each.

2 Outline

2.1 Today's Lecture

SLIDE 3

- The Simplex Algorithm: A Reminder
- The Network Simplex: A Combinatorial View
- The Network Simplex: An Animated View
- The Network Simplex: An Algebraic View

3 The Simplex Algorithm

3.1 A Reminder

3.1.1 The Problem

SLIDE 4

$$\begin{array}{lrcl} \min & \boldsymbol{c}'\boldsymbol{x} & & \\ \text{s.t.} & \boldsymbol{A}\boldsymbol{x} & = & \boldsymbol{b} \\ & \boldsymbol{x} & \geq & \boldsymbol{0} \end{array}$$

4 The Simplex Algorithm

4.1 A Reminder

4.1.1 The Algorithm

SLIDE 5

1. Start with basis $\boldsymbol{B} = [\boldsymbol{A}_{B(1)}, \ldots, \boldsymbol{A}_{B(m)}]$ and BFS $\boldsymbol{x}$.
2. Compute $\overline{c}_j = c_j - \boldsymbol{c}'_B \boldsymbol{B}^{-1} \boldsymbol{A}_j$.
   - If $\overline{c}_j \geq 0$; x optimal; stop.
   - Select j such that $\overline{c}_j < 0$.
3. Compute $\boldsymbol{u} = \boldsymbol{B}^{-1}\boldsymbol{A}_j$. $\theta^* = \min\limits_{1 \leq i \leq m, u_i > 0} \dfrac{x_{B(i)}}{u_i} = \dfrac{x_{B(\ell)}}{u_\ell}$.
4. Form a new basis by replacing $\boldsymbol{A}_{B(\ell)}$ with $\boldsymbol{A}_j$.
5. $y_j = \theta^*$; $y_{B(i)} = x_{B(i)} - \theta^* u_i$.

5 The Network Simplex Algorithm

5.1 The Problem

5.1.1 Combinatorially

SLIDE 6

Determine a least cost shipment of a commodity through a network in order to satisfy demands at certain nodes from available supplies at other nodes. Arcs have costs associated with them.

5.1.2 Algebraically

SLIDE 7

- Network $G = (N, A)$.
- Arc costs $c : A \to \mathcal{Z}$.
- Node balances $b : N \to \mathcal{Z}$.

$$\begin{array}{llcll}
\min & \displaystyle\sum_{(i,j)\in A} c_{ij}x_{ij} & & & \\
\text{s.t.} & \displaystyle\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} & = & b_i & \text{for all } i \in N \\
& x_{ij} & \geq & 0 & \text{for all } (i,j) \in A
\end{array}$$

5.2 Tree Solutions

5.2.1 Definition

SLIDE 8

- A *tree* is a graph that is connected and has no cycles.
- A *spanning tree* of a graph G is a subgraph that is a tree and contains all nodes of G.
- A flow x forms a *tree solution* with a spanning tree of the network if every non-tree arc has flow 0.

5.2.2 Computing the Flow

SLIDE 9

5.2.3 Trees vs. Tree Flows

SLIDE 10

- Every tree flow has a corresponding tree (and perhaps more than one).
- Given a tree, we obtain a unique tree flow associated with it.

5.2.4 BFS Property

SLIDE 11

Theorem 1 *If the objective function is bounded from below, a min-cost flow problem always has an optimal tree solution.*

5.2.5 Optimality Condition

SLIDE 12

Theorem 2 *A (feasible) tree T is optimal if, for some choice of node potentials p_i,*

(a) $\overline{c}_{ij} = c_{ij} - p_i + p_j = 0$ *for all* $(i,j) \in T$,

(b) $\overline{c}_{ij} = c_{ij} - p_i + p_j \geq 0$ *for all* $(i,j) \in A \setminus T$.

Proof:

- $\min \sum_{(i,j)\in A} c_{ij}x_{ij}$ is equivalent to $\min \sum_{(i,j)\in A} \overline{c}_{ij}x_{ij}$.
- $\min \sum_{(i,j)\in A} \overline{c}_{ij}x_{ij}$ is equivalent to $\min \sum_{(i,j)\in A\setminus T} \overline{c}_{ij}x_{ij}$.
- For any solution x, $x_{ij} \geq x^*_{ij}$ for all $(i,j) \in A \setminus T$.

5.2.6 Computing Node Potentials

Slide 13

5.2.7 Updating the Tree

Slide 14

5.3 Overview of the Algorithm

Slide 15

1. Determine an initial feasible tree T. Compute flow x and node potentials p associated with T.
2. Calculate $\overline{c}_{ij} = c_{ij} - p_i + p_j$ for $(i, j) \notin T$.
   - If $\overline{c} \geq 0$, x optimal; stop.
   - Select (i, j) with $\overline{c}_{ij} < 0$.
3. Add (i, j) to T creating a unique cycle C. Send a maximum flow around C while maintaining feasibility. Suppose the exiting arc is (k, ℓ).
4. $T := (T \setminus (k, \ell)) \cup (i, j)$.

6 Min-Cost Flow

6.1 Integrality

Slide 16

Our reasoning has two important and far-reaching implications:

- There always exists an integer optimal flow
  (if node balances b_i are integer).

- There always exist optimal integer node potentials (if arc costs c_{ij} are integer).

7 The Network Simplex Algorithm

7.1 An Animation

SLIDE 17

7.2 The Algebraic View

SLIDE 18

- Bases and trees.
- Dual variables and node potentials.
- Changing bases and updating trees.
- Optimality testing.

7.2.1 Bases vs. Trees

SLIDE 19

The constraint matrix A of the min-cost flow problem is the node-arc incidence matrix of the underlying network.

	(1,2)	(2,6)	(3,2)	(4,3)	(4,5)	(5,3)	(5,6)	(6,7)	(7,1)
1	+1	0	0	0	0	0	0	0	−1
2	−1	+1	−1	0	0	0	0	0	0
3	0	0	+1	−1	0	−1	0	0	0
4	0	0	0	+1	+1	0	0	0	0
5	0	0	0	0	−1	+1	+1	0	0
6	0	−1	0	0	0	0	−1	+1	0
7	0	0	0	0	0	0	0	−1	+1

SLIDE 20

The rows of A are linearly dependent.
Let B be the submatrix corresponding to the tree

SLIDE 21

	(1,2)	(2,6)	(3,2)	(4,3)	(5,3)	(7,1)
1	+1	0	0	0	0	−1
2	−1	+1	−1	0	0	0
3	0	0	+1	−1	−1	0
4	0	0	0	+1	0	0
5	0	0	0	0	+1	0
6	0	−1	0	0	0	0
7	0	0	0	0	0	+1

Let B be the submatrix corresponding to the tree

	(1,2)	(2,6)	(3,2)	(4,3)	(5,3)	(7,1)
4	0	0	0	+1	0	0
5	0	0	0	0	+1	0
6	0	−1	0	0	0	0
7	0	0	0	0	0	+1
3	0	0	+1	−1	−1	0
2	−1	+1	−1	0	0	0
1	+1	0	0	0	0	−1

Permuting Rows

Slide 22

Let B be the submatrix corresponding to the tree

	(4,3)	(5,3)	(2,6)	(7,1)	(3,2)	(1,2)
4	+1	0	0	0	0	0
5	0	+1	0	0	0	0
6	0	0	−1	0	0	0
7	0	0	0	+1	0	0
3	−1	−1	0	0	+1	0
2	0	0	+1	0	−1	−1
1	0	0	0	−1	0	+1

Permuting Columns

Slide 23

Corollary 1

(a) *The matrix A has rank $n-1$.*

(b) *Every tree solution is a basic solution.*

Slide 24

Theorem 3 *Every tree defines a basis and, conversely, every basis definies a tree.*

Suppose the graph defined by a basis contains a cycle $1-2-3-4-5-6$:

	(1,2)	(2,3)	(4,3)	(5,4)	(5,6)	(1,6)
1	+1	0	0	0	0	+1
2	−1	+1	0	0	0	0
3	0	−1	−1	0	0	0
4	0	0	+1	−1	0	0
5	0	0	0	+1	+1	0
6	0	0	0	0	−1	−1

7.2.2 Dual Variables vs. Node Potentials

SLIDE 25

Remember, the simplex algorithm computes the dual variables p as the solution to $p' B = c'_B$.

$$(p_4, p_5, p_6, p_7, p_3, p_2) \begin{pmatrix} +1 & 0 & 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 & 0 & 0 \\ -1 & -1 & 0 & 0 & +1 & 0 \\ 0 & 0 & +1 & 0 & -1 & -1 \end{pmatrix}$$

$$= \quad (c_{43},\ c_{53},\ c_{26},\ c_{71},\ c_{32},\ c_{12})$$

Hence, $p_2 = -c_{12}$, $p_3 = c_{32} + p_2$, $p_7 = c_{71}$, ...

7.2.3 Optimality Testing

SLIDE 26

Remember, the simplex algorithm computes the reduced costs $\overline{c}$ as $\overline{c}_{ij} = c_{ij} - p' A_{ij}$.

	(1,2)	(2,6)	(3,2)	(4,3)	(4,5)	(5,3)	(5,6)	(6,7)	(7,1)
1	+1	0	0	0	0	0	0	0	−1
2	−1	+1	−1	0	0	0	0	0	0
3	0	0	+1	−1	0	−1	0	0	0
4	0	0	0	+1	+1	0	0	0	0
5	0	0	0	0	−1	+1	+1	0	0
6	0	−1	0	0	0	0	−1	+1	0
7	0	0	0	0	0	0	0	−1	+1

Therefore, $\overline{c}_{ij} = c_{ij} - p_i + p_j$.

7.3 Summary

SLIDE 27

- The network simplex algorithm is extremely fast in practice.
- Relying on network data structures, rather than matrix algebra, causes the speedups. It leads to simple rules for selecting the entering and exiting variables.
- Running time per pivot:
  - arcs scanned to identify an entering arc,
  - arcs scanned of the basic cycle,

  - nodes of the subtree.

- A good pivot rule can dramatically reduce running time in practice.