IN7K3-1: Flujo en Redes - Primavera 2011

4: Minimum Cost Flow

Profesor: Fernando Ordóñez

1 Introduction

We are given a network $G = (N, A)$ with

u_{ij} - capacity of arc (i, j)

c_{ij} - unit cost of sending flow on arc (i, j)

x_{ij} - value of flow on arc (i, j) (decision variable)

We want to decide the transportation from our sources to destinations at minimum cost:

$$\begin{array}{lll} \min & \displaystyle\sum_{(i,j)\in A} c_{ij}x_{ij} & \\ \text{s.t.} & \displaystyle\sum_{\{j \mid (i,j)\in A\}} x_{ij} - \sum_{\{k \mid (k,i)\in A\}} x_{ki} = b_i & \text{for any } i \in N \\ & 0 \le x_{ij} \le u_{ij} & \text{for any } (i,j) \in A\ . \end{array}$$

Assumptions

- Data is integral
- Network is directed
- $\sum_{i\in N} b_i = 0$ (wlog)
- There is a feasible flow (wlog)

2 Applications/ Examples

Shortest Path and Max Flow can be seen as examples of min cost flow.

2.1 Transportation Problem

Consider K production facilities, L warehouses, and M demand centers. Let c_{ij} be the transportation cost from i to j, and u_{ij} the maximum amount that can be transported in one week. Decide the weekly minimum cost transportation schedule that meets demand.

2.2 Scheduling with Deferral Costs

You want to schedule p jobs on q identical machines. Assume that each job has a release time r_j, a processing time α_j, and a deferral cost $c_j(\tau)$, which is a monotonically increasing function of its completion time τ. The jobs do not have a fixed due date, and assume there is no job preemption. Find a schedule of jobs that minimizes the total deferral cost. This problem is tractable if we add the assumption that $\alpha_j = \alpha$ for all j.

2.3 Optimal Capacity Scheduling

You need to contract $d(i)$ units of warehouse capacity for periods $i = 1, \ldots, n$. Let c_{ij} be the price of contracting 1 unit of capacity in period i that will be available from period i to $j - 1$. Find the minimum cost capacity schedule that meets your demand needs.

2.4 The caterer problem

Assume there is a demand d_i for napkins on day $i = 1, \ldots, 7$. Cost of
a new napkin is a cents
2-day laundry is b cents per napkin Minimize the cost of meeting the demand.
1-day laundry is c cents per napkin

3 Optimality Conditions

Theorem 1. *(Negative Cycle Optimality Conditions). A feasible flow x^* is optimal for the minimum cost flow problem if and only if the residual network $G(x^*)$ contains no negative cost directed cycle*

proof:

Reduced Costs

Let π_i denote node potentials (dual prices) for each node i. Define $c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j$. Idea is to reward exports and penalize imports.

Recall that for any directed path P from k to l $\quad \sum_{(i,j)\in P} c_{ij}^{\pi} = \sum_{(i,j)\in P} c_{ij} - \pi_k + \pi_l$.

Theorem 2. *(Reduced Cost Optimality Conditions). A feasible flow x^* is optimal min cost flow if and only if some node potentials π satisfy $c_{ij}^{\pi} \geq 0$ for every $(i,j) \in G(x^*)$.*

proof:

Reduced cost optimality in terms of the original network:

A feasible flow x^* is optimal min cost flow iff there exist node potentials π such that

If $c_{ij}^{\pi} > 0$ then $x_{ij}^* = 0$

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

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

4 Negative Cycle Algorithm

Set x a feasible flow
while $G(x)$ contains a negative cost cycle do
    Find a negative cost cycle C in $G(x)$
    Let $\delta = \min\{r_{ij} \mid (i,j) \in C\}$
    Augment the flow x by δ on cycle C, update $G(x)$
endwhile

If supplies/demands and capacities are integral we maintain integral flow solutions.

The Negative Cycle Algorithm is finite if all data are finite and integral.

Work per iteration

Total complexity

Room for improvement

If look for minimum mean average cost cycle (i.e. minimizes $\frac{1}{|W|}\sum_{(i,j)\in W} c_{ij}$ over all cycles W), The algorithm is strongly polynomial, running in $O(n^2m^3 \log n)$ time.

5 Successive Shortest Path Algorithm

A flow x such that $0 \leq x \leq u$ is a **pseudo-flow**. We denote the excess (or deficit) of this flow at each node by $e(i)$, that is

$$e(i) = b(i) + \sum_{\{j \mid (j,i) \in A\}} x_{ji} - \sum_{\{j \mid (i,j) \in A\}} x_{ij} \ .$$

The infeasibility of the pseudo-flow x is $\sum_{e(i)>0} e(i)$.

Example:

This algorithm will maintain
node potentials π
a pseudo-flow x

The pair (x, π) is dual feasible if x is a pseudo-flow and $c^\pi \geq 0$ for every arc in $G(x)$.

Idea: Keep $c^\pi \geq 0$ for all arcs in $G(x)$ throughout the algorithm, while improving the feasibility of the pseudo-flow x.

Example What happens when we send flow from an excess to deficit node

Need to construct paths of reduced cost 0. How do we update the node potentials to make paths of cost 0?

Summary: We showed that if we push flow along a path of zero reduced cost, then we keep $c^\pi \geq 0$ on the new residual network. Consider the following example, and try to set node potentials that will create paths of 0 reduced cost:

Algorithm:

Set (x, π) dual feasible
While x is infeasible ($\sum_{e(i)>0} e(i) > 0$) do
    Select s such that $e(s) > 0$, t such that $e(t) < 0$
    For all i determine $d(i)$ the shortest distance from s to i, with costs c^π on $G(x)$
    Update $\pi = \pi - d$
    Send as much flow on the 0 reduced cost path from s to t
    Update x
endwhile

Complexity of Successive Shortest Path Algorithm
Each augmentation reduces the infeasibility by at least 1:

Finite number of iterations:

Work per iteration (the hard part is running Dijkstra):

6 Some efficient variations

6.1 Primal-Dual Algorithm

Same algorithm, but now send flow from many excess nodes to many deficit nodes at the same time.

How would you do this? Hint consider the subgraph of $G(x)$ only with arcs where $c_{ij}^{\pi} = 0$, and formulate a max flow problem.

6.2 Capacity Scaling Successive Shortest Path

Just like successive shortest path, but try to send at least Δ units of flow from s to t, where $e(s) \geq \Delta$ and $e(t) \leq -\Delta$. If no such path exists reduce Δ.

Algorithm:

Set (x, π) dual feasible, Δ large

While $\Delta \geq 1$ do

For all $(i, j) \in G(x)$ such that $r_{ij} \geq \Delta$ and $c_{ij} < 0$, send r_{ij} from i to j.

While there is $e(s) \geq \Delta$, $e(t) \leq \Delta$, and a path from s to t in $G(x, \Delta)$ do

For all i determine $d(i)$ the shortest distance from s to i, with costs c^{π} on $G(x, \Delta)$

Update $\pi = \pi - d$

Send as much flow on the 0 reduced cost path from s to t

Update x

endwhile

endwhile

Complexity

Change in total number of iterations. Let B be an upper bound on the infeasibility at a node.

Total infeasibility at the end of a 2Δ-scaling phase is $n2\Delta$:

At the beginning of a Δ-scaling phase, infeasibility can increase by $m\Delta$:

The total infeasibility at the beginning of Δ-scaling phase is $2n\Delta + m\Delta = O(m\Delta)$.

Initial infeasibility is $O(mB)$, which implies there will be $O(\log B)$ scaling phases. The number of augmentations per scaling phase is $O(m)$, Total complexity $O(m \log B(m + n \log nC))$.

With modifications this algorithm can be made strongly polynomial, with best complexity of $O((m \log n)(m + n \log n))$.

7 Linear Programming and Min Cost Flow

Optimization problem with all linear functions. The Simplex method introduced in 1947 is still one of the methods to solve LPs. In vector notation, a linear program in standard form is the following:

$$\begin{aligned} z_* = \quad & \min \quad c^T x \\ & \text{s.t.} \quad Ax = b \\ & \qquad x \geq 0 \end{aligned}$$

7.1 Graphic Example

Lets solve:

$$\begin{aligned} z = \quad & \max \quad x_1 + x_2 \\ & \text{s.t.} \quad 2x_1 + 3x_2 \leq 12 \\ & \qquad x_1 \leq 4 \\ & \qquad -x_1 + x_2 \leq 2 \\ & \qquad x_1, x_2 \geq 0 \end{aligned}$$

The feasible region is a polyhedron, the corners of a polyhedron are *extreme points*. Extreme points can not be written as a convex combination of points in the feasible region.

7.2 Dual Problem

Associated to any LP there is a related dual linear program. For the LP in standard form, the dual is

$$\begin{aligned} z^* = \quad & \max \quad b^T y \\ & \text{s.t.} \quad A^T y \leq c \\ & \qquad y \text{ free} \end{aligned}$$

Theorem 3. *Weak duality: If x and y are primal and dual feasible solutions, then $c^T x \geq b^T y$.*
proof:

Implications.

Theorem 4. *Strong duality: If either the primal or the dual have a finite optimal solution, so does the other and both have the same optimal objective function value.*

Complimentary slackness condition:

$$x_i \left(A^T y - c\right)_i = 0 \qquad \text{for any } i = 1, \ldots, n \ .$$

Theorem 5. *Complimentary slackness optimality conditions: The primal and dual feasible solutions x and y are optimal solutions if and only if they satisfy complimentary slackness.*
proof:

7.3 Dual of a min-cost flow

Consider the dual of the shortest path problem as a min cost flow:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Nx = b \\ & x \geq 0 \end{aligned}$$

where b is such that $b_s = 1$, $b_t = -1$, and $b_i = 0$ for $i \neq s, t$.

Derive the dual for the general min-cost flow problem, with arbitrary b:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Nx = b \\ & 0 \leq x \leq u \ . \end{aligned}$$

What do the complementarity slackness conditions look like for a min-cost flow problem?

7.4 Network Simplex Algorithm

Simplex algorithm visits adjacent *extreme points* until it is can prove its optimal. For a min-cost network flow, the algorithm does the same, it visits adjacent *tree* solutions until its optimal.

Algorithm 1 Network Simplex Algorithm

1: determine an initial feasible tree (T, L, U)
2: let x be the flow and π be the node potentials
3: **while** some non-tree arc violates optimality conditions **do**
4: select an arc (k, l) violating its optimality conditions
5: add (k, l) to T and determine the leaving arc (p, q)
6: update the tree and solutions x and π
7: **STOP**. The solution x, π are a primal-dual optimal pair.