In the name of God

Network Flows

5. Minimum Cost Flow Problem
5.1 Introduction

Fall 2010

Instructor: Dr. Masoud Yaghini

Outline

- Introduction
- Cycle Canceling Algorithm
- Successive Shortest Path Algorithm

Introduction

Introduction

- In the previous chapters, we have considered two special cases of ***minimum cost flow problem***
  - ***the shortest path problem***
  - ***the maximum flow problem***
- These problems address different components of the overall minimum cost flow problem
  - ***Shortest path problems*** consider arc flow costs but no flow capacities;
  - ***Maximum flow problems*** consider capacities but only the simplest cost structure.
- The ***minimum cost flow problem*** combines these problem ingredients

Introduction

- It is easy to understand the basic nature of ***shortest path*** and ***maximum flow problems*** and to develop core algorithms for solving them;
  - nevertheless, designing and analyzing efficient algorithms is a very challenging task, requiring considerable ingenuity and considerable insight concerning both basic algorithmic strategies and their implementations.
- The algorithms for solving the ***minimum cost flow problem*** are not as efficient as those for the ***shortest path*** and ***maximum flow problems***

Notation and Assumptions

- $G = (N, A)$: a directed network with a cost c_{ij} and a capacity u_{ij} associated with every arc $(i, j) \in A$.
- $b(i)$: a number $b(i)$, $i \in N$, indicates its supply or demand depending on whether $b(i) > 0$ or $b(i) < 0$
- x_{ij} : amount shipped on arc (i, j)
- C : the largest magnitude of any arc cost.
- U : the largest magnitude of any supply/demand or finite arc capacity.
- l_{ij} : the lower bounds on arc flows which are all zero.

Notation and Assumptions

- The ***minimum cost flow problem*** can be stated as follows:

$$\text{Minimize} \quad z(x) = \sum_{(i,j)\in A} c_{ij} x_{ij}$$

subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \text{for all } i \in N,$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for all } (i, j) \in A.$$

Notation and Assumptions

- We further make the following assumptions:
- ***Assumption 1. All data (cost, supply/demand, and capacity) are integral.***
  - this assumption is not really restrictive in practice because computers work with rational numbers which we can convert to integer numbers by multiplying by a suitably large number.

Notation and Assumptions

- ***Assumption 2. The network is directed.***
  - we can always fulfill this assumption by transforming any undirected network into a directed network.

- ***Assumption 3. The supplies/demands at the nodes satisfy the condition***

$$\sum_{i \in N} b(i) = 0$$

  - ***and the minimum cost flow problem has a feasible solution.***

Notation and Assumptions

- ***Assumption 4. We assume that the network G contains an uncapacitated directed path (i.e., each arc in the path has infinite capacity) between every pair of nodes.***
  - We impose this condition, if necessary, by adding artificial arcs $(1, j)$ and $(j, 1)$ for each $j \in N$ and assigning a large cost and infinite capacity to each of these arcs.
  - No such arc would appear in a minimum cost solution unless the problem contains no feasible solution without artificial arcs.

Notation and Assumptions

- ***Assumption 5. All arc costs are nonnegative.***
  - This assumption imposes no loss of generality since the arc reversal transformation

Introduction

- ***Residual Network***
  - The residual network $G(x)$ corresponding to a flow x is defined as follows.
  - We replace each arc $(i, j) \in A$ by two arcs (i, j) and (j, i).
  - The arc (i, j) has cost c_{ij} and residual capacity $r_{ij} = u_{ij} - x_{ij}$
  - the arc (j, i) has cost $c_{ji} = -c_{ij}$ and residual capacity $r_{ji} = x_{ij}$
  - The residual network consists only of arcs with positive residual capacity.

Cycle Canceling Algorithm

Cycle Canceling Algorithm

- ***Cycle Canceling Algorithm***
  - This algorithm maintains a feasible solution and at every iteration attempts to improve its objective function value.
  - The algorithm first establishes a feasible flow x in the network by solving a maximum flow problem
  - Then it iteratively finds ***negative cost-directed cycles*** in the residual network and augments flows on these cycles.
  - The algorithm terminates when the residual network contains no negative cost-directed cycle.
  - When the algorithm terminates, it has found a minimum cost flow.

- ***Theorem (Negative Cycle Optimality Conditions)***
  - A feasible solution x* is an optimal solution of the minimum cost flow problem if and only if it satisfies the negative cycle optimality conditions: namely, the residual network $G(x^*)$ contains no negative cost (directed) cycle.

Cycle Canceling Algorithm

- *Cycle Canceling Algorithm*

algorithm *cycle-canceling*;
begin
    establish a feasible flow x in the network;
    while $G(x)$ contains a negative cycle **do**
    begin
        use some algorithm to identify a negative cycle W;
        $\delta := \min\{r_{ij} : (i, j) \in W\}$;
        augment δ units of flow in the cycle W and update $G(x)$;
    end;
end;

Successive Shortest Path Algorithm

Successive Shortest Path Algorithm

- The cycle-canceling algorithm maintains feasibility of the solution at every step and attempts to achieve optimality.
- In contrast, the ***successive shortest path algorithm*** maintains optimality of the solution at every step and strives to attain feasibility.
- It maintains a solution x that satisfies the nonnegativity and capacity constraints, but violates the mass balance constraints of the nodes.

Successive Shortest Path Algorithm

- At each step, the algorithm selects a node *s* with excess supply (i.e., supply not yet sent to some demand node) and a node *t* with unfulfilled demand and sends flow from *s* to *t* along a shortest path in the residual network.
- The algorithm terminates when the current solution satisfies all the mass balance constraints.

Successive Shortest Path Algorithm

- To describe this algorithm as well as several later developments, we first introduce the concept of ***pseudoflows***.
- A pseudoflow is a function x: A -> R^+ satisfying only the capacity and nonnegativity constraints; it need not satisfy the mass balance constraints.
- For any pseudoflow x, we define the imbalance of node i as

$$e(i) = b(i) + \sum_{\{j:(j,i)\in A\}} x_{ji} - \sum_{\{j:(i,j)\in A\}} x_{ij} \qquad \text{for all } i \in N.$$

Successive Shortest Path Algorithm

- For some node i:
  - If $e(i) > 0$, we refer to $e(i)$ as the ***excess*** of node i;
  - if $e(i) < 0$, we call $-e$(i) the node's ***deficit***.
  - If $e(i) = 0$, we refer to a node i as ***balanced***.
- Let E and D denote the sets of excess and deficit nodes in the network.
- Notice that

$$\sum_{i \in N} e(i) = \sum_{i \in N} b(i) = 0,$$

- And hence

$$\sum_{i \in E} e(i) = -\sum_{i \in D} e(i).$$

Successive Shortest Path Algorithm

- Consequently, if the network contains an excess node, it must also contain a deficit node.
- The residual network corresponding to a pseudoflow is defined in the same way that we define the residual network for a flow.

The End