Network Design and Optimization course

Lecture 6

Alberto Ceselli
alberto.ceselli@unimi.it

Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano

November 9, 2011

The problem

We are now able to cope with costs and capacities at the same time. Given

- a set of nodes,
- a set of links connecting them,
- a connection request between two nodes of the network,

(that is, an existing network).
I want to

- decide which links to use in the connection (route)
- maximizing the quality of service (e.g. minimizing delay time)

Assumptions

Some assumptions:

1. no *costs* involved: packets can also follow non-shortest paths,

3. the capacity of each link is enough for the whole connection request.

Assumptions

Some assumptions:

2. $\rightarrow$ *cost* matters!

4. $\rightarrow$ the capacity of links may not be enough for the whole connection request.

Recognizing a known problem ...

We observe

- when capacities are always large enough: Shortest Path Problems,
- when costs are not involved: Max Flow Problems.

we are facing a *Min Cost Flow (MCF) problem*.

Recognizing a known problem ...

We observe

- when capacities are always large enough: Shortest Path Problems,
- when costs are not involved: Max Flow Problems.

we are facing a *Min Cost Flow (MCF) problem*.
N.B. Min Cost Flows generalize both Shortest Path and Max Flow problems.

Graph model

Given a network, build a *directed* graph $G = (V, A)$ having

- one vertex $i \in V$ for each node of the network
- one arc $a \in A \subseteq V \times V$ for each link of the network
- capacities $u_{(i,j)}$ on each arc $(i, j) \in A$
- costs $c_{(i,j)}$ on each arc $(i, j) \in A$
- flow consumption b_i for each node $i \in V$

Mathematical Programming model

Let $x_{(i,j)}$ be *decision variables* representing the *amount of flow* sent on arc (i,j). Let v represent the total cost of routing packets in the network.

$$
\begin{aligned}
\text{maximize } v = \sum_{(i,j)\in A} c_{(i,j)} x_{(i,j)} & \\
\text{subject to } \sum_{j\in V} x_{(i,j)} = \sum_{k\in V} x_{(k,i)} + b_i & \qquad \forall i \in V, i \neq s,t \\
0 \leq x_{(i,j)} \leq u_{(i,j)} & \qquad \forall (i,j) \in A
\end{aligned}
$$

Assumptions

We assume that

- all data are integral,
- it is possible to send M units of flow from s to t (i.e. the problem is feasible),
- all arc costs are non-negative (but it can be shown that this is without loss of generality),
- $b_i = 0$ for all $i \in V$, except a special vertex $s \in V$ representing the origin of packets ($b_s = -M$ units) and a special vertex $t \in V$ representing the destination of packets ($b_t = M$ units) (this is also w.l.o.g).

and for technical reasons

- the graph G contains an *uncapacitated* directed path between every pair of nodes (but we can always add a suitable gadget).

Residual network

As in Max Flow problems, the *residual network* $G(X)$ corresponding to a flow x is defined as follows.

- We replace each arc $(i, j) \in A$ with 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 $r_{ji} = x_{ij}$
- only arcs with positive residual capacity are actually considered in $G(x)$.

Algorithms for MCF work rather on residual networks than on the starting graph.

Optimality conditions

Recall optimality conditions for Shortest Paths

- $d(j) \leq d(i) + c_{ij}$ for each $(i, j) \in A$

and optimality conditions for Max Flow

- (e.g.) no augmenting paths

$\rightarrow$ in both cases it is enough to "correct" a solution until they are satisfied, to obtain an exact algorithm! We look for something similar for MCF.

Optimality conditions

We will see three (equivalent) optimality conditions:

- negative cycle
- reduced cost
- complementary slackness

Negative Cycle optimality conditions

Theorem (negative cycle optimality conditions): A feasible solution x^* is an optimal solution of the MCF problem *if and only if* the residual network $G(x^*)$ contains no negative cost (directed) cycle.

Proof. Omitted.

Negative Cycle algorithm

Idea: maintain feasibility of the solution at every step, and attempt to achieve optimality.
Orlin's slides ahead!

put slides 20 .. 24

Negative cycles

Key issue: how to find a negative cost cycle?

-
-
-

Negative cycles

Key issue: how to find a negative cost cycle?

- Run a shortest path algorithm (e.g. label correcting), and keep the first negative cycle
-
-

Negative cycles

Key issue: how to find a negative cost cycle?

-
- Augment flow in a negative cycle with maximum improvement:
  find W s.t. $-(\sum_{(i,j)\in W} c_{ij}) \cdot (\min\{r_{ij} : (i,j) \in W\})$ is maximum
  $\rightarrow$ NP-Hard problem (but polynomially solvable with a slight modification)
  $\rightarrow$ at most $O(mlog(mCU))$ iterations
-

Negative cycles

Key issue: how to find a negative cost cycle?

-
-
- Augment flow in a negative cycle with maximum average arc improvement:
  find W s.t. $-(\sum_{(i,j)\in W} c_{ij}) \cdot (\min\{r_{ij} : (i,j) \in W\})/|W|$ is maximum
  $\rightarrow$ such a cycle can be found in $O(nm)$ or $O(\sqrt{n}m\log(nC))$
  $\rightarrow$ at most $O(\min\{nm\log(nC), nm^2\log n\})$ iterations

Negative cycles

Key issue: how to find a negative cost cycle?

-
-
- Augment flow in a negative cycle with maximum average arc improvement:
  find W s.t. $-(\sum_{(i,j)\in W} c_{ij}) \cdot (\min\{r_{ij} : (i,j) \in W\})/|W|$ is maximum
  $\rightarrow$ such a cycle can be found in $O(nm)$ or $O(\sqrt{n}m\log(nC))$
  $\rightarrow$ at most $O(\min\{nm\log(nC), nm^2\log n\})$ iterations
- Network simplex algorithm

Università degli Studi di Milano

Reduced Cost optimality conditions

Recall from the Johnson's algorithm for Shortest Paths:

- we can assign *node potentials* π_i to each $i \in V$
- $c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j$ is the *reduced cost* of arc (i, j)
- for any path P from s to t,
  $\sum_{(i,j)\in P} c^{\pi}_{ij} = \sum_{(i,j)\in P} c_{ij} - \pi_s + \pi_t$
- in a similar way, for any cycle W, $\sum_{(i,j)\in W} c^{\pi}_{ij} = \sum_{(i,j)\in W} c_{ij}$

.

Reduced Cost optimality conditions

Recall from the Johnson's algorithm for Shortest Paths:

- we can assign *node potentials* π_i to each $i \in V$
- $c^{\pi}_{ij} = c_{ij} - \pi_i + \pi_j$ is the *reduced cost* of arc (i, j)
- for any path P from s to t, $\sum_{(i,j)\in P} c^{\pi}_{ij} = \sum_{(i,j)\in P} c_{ij} - \pi_s + \pi_t$
- in a similar way, for any cycle W, $\sum_{(i,j)\in W} c^{\pi}_{ij} = \sum_{(i,j)\in W} c_{ij}$
- if we compute optimal $d()$ labels, and set $\pi_i = -d(i)$, then $c^{d}_{ij} = c_{ij} - d(i) + d(j) \geq 0$ for each (i, j).

Reduced Cost optimality conditions

Theorem (reduced cost optimality conditions): A feasible solution x^* is an optimal solution of the MCF problem *if and only if* some set of node potentials π exists, that satisfy the following condition:

$$c_{ij}^{\pi} \geq 0 \quad \forall (i,j) \text{ in } G(x^*),$$

Reduced Cost optimality conditions

Proof.

Reduced Cost optimality conditions

Proof.

- **(if):** suppose that

$$c_{ij}^{\pi} \geq 0 \forall (i,j);$$

Reduced Cost optimality conditions

Proof.

- **(if):** suppose that

$$c_{ij}^{\pi} \geq 0 \forall (i,j);$$

- then, any cycle W has

$$\sum_{(i,j) \in W} c_{ij}^{\pi} \geq 0$$

Reduced Cost optimality conditions

Proof.

- **(if):** suppose that

$$c_{ij}^{\pi} \geq 0 \forall (i,j);$$

- then, any cycle W has

$$\sum_{(i,j) \in W} c_{ij}^{\pi} \geq 0$$

- but since

$$\sum_{(i,j) \in W} c_{ij}^{\pi} = \sum_{(i,j) \in W} c_{ij}$$

, then $\sum_{(i,j) \in W} c_{ij} \geq 0$ and therefore no negative cost cycle exists (i.e. x^* is optimal).

Reduced Cost optimality conditions

- **(only if):** if x^* is optimal, then it contains no negative cost cycle

Reduced Cost optimality conditions

- **(only if):** if x^* is optimal, then it contains no negative cost cycle
- but then I can create distance labels $d()$ all satisfying $d(j) \leq d(i) + c_{ij}$ (SP optimality conditions)

Reduced Cost optimality conditions

- **(only if):** if x^* is optimal, then it contains no negative cost cycle
- but then I can create distance labels $d()$ all satisfying $d(j) \leq d(i) + c_{ij}$ (SP optimality conditions)
- and setting $\pi_i = -d(i)$ I have $-\pi_j \leq -\pi_i + c_{ij} \rightarrow 0 \leq \pi_j - \pi_i + c_{ij} = c^{\pi}_{ij}$.

Successive Shortest Path algorithm

Idea: maintain optimality of the solution (with respect to reduced costs) at every step, and attempt to achieve feasibility.

- Keep a solution x satisfying non-negativity and capacity constraints
- allow (temporary) violation of flow balance constraints

Successive Shortest Path algorithm

Idea: maintain optimality of the solution (with respect to reduced costs) at every step, and attempt to achieve feasibility.

- Keep a solution x satisfying non-negativity and capacity constraints
- allow (temporary) violation of flow balance constraints
- x is a *pseudoflow*
- we define as *imbalance* in i the value

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

Successive Shortest Path algorithm

BEGIN
$x := 0; \pi := 0$
$e(i) := b_i$ for all $i \in V$
initialize sets $E := \{i : e(i) > 0\}$ and $D := \{i : e(i) < 0\}$
while $E \neq \emptyset$ do
begin
    select a node $k \in E$ and a node $\ell \in D$
    find shortest path distances $d(j)$ from k to each j in $G(x)$
        using reduced costs c^π
    let P be a shortest path from k to ℓ
    update $\pi := \pi - d()$
    let $\delta := \min\{e(k), -e(l), \min\{r_{ij} : (i,j) \in P\}\}$
    augment δ units of flow along P
    update $G(x), x, E, D$ and the reduced costs
end
END

Successive Shortest Path algorithm

Example: page 322 of Network Flows.

Complementary Slackness optimality conditions

Theorem (complementary slackness optimality conditions): A feasible solution x^* is an optimal solution of the MCF problem *if and only if* some set of node potentials π exists, such that the reduced costs and flow values satisfy the following complementary slackness conditions for every arc $(i,j) \in A$:

1. if $c_{ij}^{\pi} > 0$ then $x_{ij}^* = 0$
2. if $c_{ij}^{\pi} < 0$ then $x_{ij}^* = u_{ij}$
3. if $0 < x_{ij}^* < u_{ij}$ then $c_{ij}^{\pi} = 0$

Proof.

On the blackboard (Network flows, page 310).

Primal-Dual algorithm

Idea: maintain pseudoflows as in the Successive Shortest Path algorithm, but instead of iteratively sending flow on shortest paths solve a maximum flow problem.

- consider all elements with excess or deficit at once by introducing artifical sources and sinks
- build an *admissible network*, obtained from $G(x)$ by removing all arcs having reduced cost different than 0,
- observe that *any* path in the admissible network is a shortest path in the residual network.

Successive Shortest Path algorithm

BEGIN
$x := 0; \pi := 0$
$e(s) := b_s;\ e(t) := b_t$
while $e(s) > 0$ do
begin
    find shortest path distances $d(j)$ from s to each j in $G(x)$
        using reduced costs c^π
    update $\pi := \pi - d()$
    define the admissible network $\widetilde{G}(x)$, including only arcs
        of $G(x)$ having $c^\pi_{ij} = 0$
    find a maximum flow from s to t in $\widetilde{G}(x)$
    update $G(x), e(s), e(t)$ and the reduced costs
end
END

Primal-Dual algorithm

Example: page 325 of Network Flows.

MCF algorithm implementation

Lab session: implementing MCF algorithms in AMPL.