Lecture 10: Minimum Cost Network Flows

Prof. Krishna R. Pattipati
Dept. of Electrical and Computer Engineering
University of Connecticut
Contact: krishna@engr.uconn.edu; (860) 486-2890

Minimum cost network flows

- Minimum cost network flows
  - LP formulation
  - Special case: shortest path problem
- **Minimum cost circulation problem** as a generalization
  - Special cases
    - Maximum flow problem
    - Transportation problem
    - Assignment problem
- Dual problem and optimality conditions
- Relaxation algorithms
  - RELAX
  - ϵ-RELAX

Minimum cost network flow problem

- Minimum cost network flow (MCNF) problem
  - Want to send a specified amount of flow v from s to t ∋ the total cost is a minimum
  - LP formulation
    - a_{ij} = cost per unit flow through edge (i, j)
    - x_{ij} = flow of commodity through edge (i, j)

$$
\begin{aligned}
&\min \textstyle\sum_{i,j} a_{ij} x_{ij} \\
&\text{s.t.} \quad -\textstyle\sum_j x_{ji} + \sum_k x_{ik} = 0, \quad \forall i \neq s,t \\
&\qquad -\textstyle\sum_j x_{js} + \sum_k x_{sk} - v = 0 \\
&\qquad -\textstyle\sum_j x_{jt} + \sum_k x_{tk} + v = 0 \\
&\qquad 0 \leq b_{ij} \leq x_{ij} \leq c_{ij}
\end{aligned}
$$

  - Special case: shortest path problem
    - $b_{ij} = 0, c_{ij} = 1, v = 1$

    $\Rightarrow$ Single source-single destination shortest path problem

Minimum cost circulation problem (MCCP)

- What if a commodity can enter or exit at any node?

$$\begin{aligned} \min \ & \textstyle\sum_{i,j} a_{ij} x_{ij} \\ \text{s.t.} \ & -\textstyle\sum_j x_{ji} + \sum_k x_{ik} = s_i, \forall i \\ & \textstyle\sum_i s_i = 0 \\ & 0 \le b_{ij} \le x_{ij} \le c_{ij} \end{aligned}$$

- Minimum cost circulation problem
  - Can view max. flow, min. cost and shortest path problems as special cases of the so-called **min. cost circulation problem**
  - LP formulation of minimum cost circulation problem

$$\begin{aligned} \min \ & \textstyle\sum_{i,j} a_{ij} x_{ij} \\ \text{s.t.} \ & -\textstyle\sum_j x_{ji} + \sum_k x_{ik} = 0, \quad \forall i \\ & 0 \le b_{ij} \le x_{ij} \le c_{ij} \end{aligned}$$

  - **Special case 1**: maximum flow problem
    - Add return arc $(t\ ,s) \ni \ b_{ts} = 0$, $c_{ts} = \infty$ and $a_{ts} = -1$
    - For all other arcs, set $a_{ij} = 0$
    - LP formulation

$$\begin{aligned} \min \ & -x_{ts} \ \text{ or } \ \max \ x_{st} \\ \text{s.t.} \ & -\textstyle\sum_j x_{ji} + \sum_k x_{ik} = 0 \\ & 0 \le b_{ij} \le x_{ij} \le c_{ij} \end{aligned}$$

Recall flow is skew symmetric

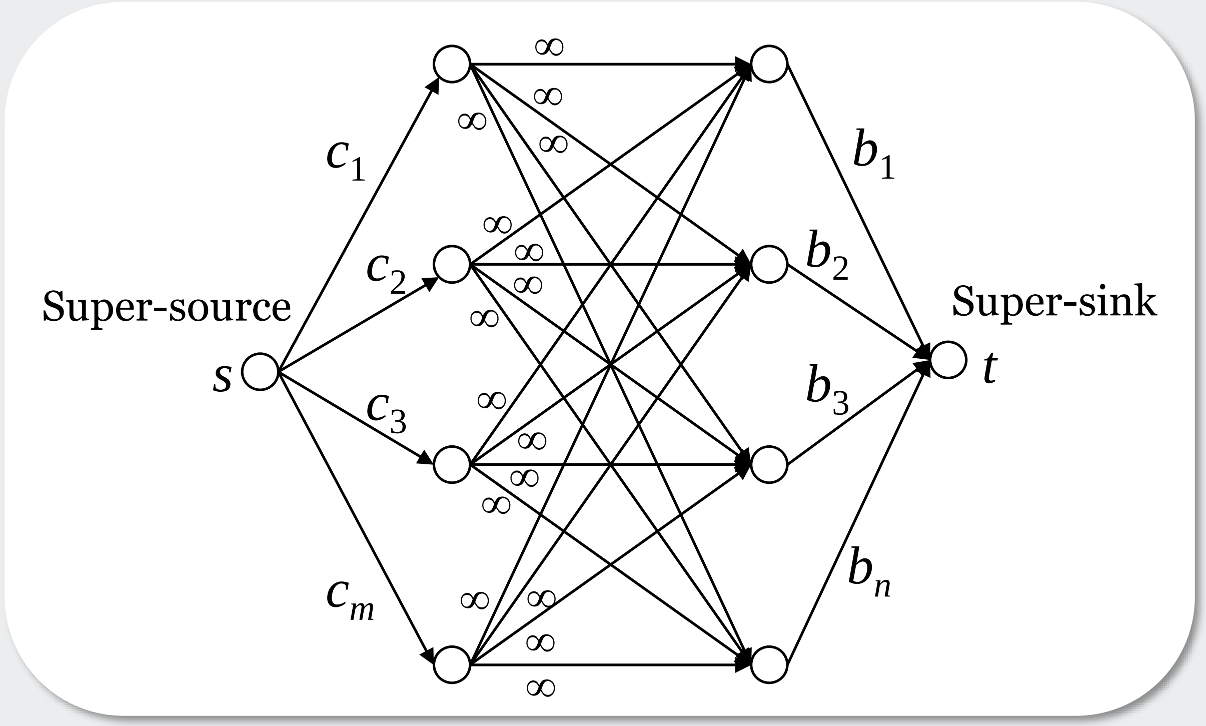
MCCP and special cases

- **Special case 2**: minimum cost network flow (MCNF) problem
  - Add a return *arc*$(t,s) \ni b_{ts} = 0,\ c_{ts} = v,\ \&\ a_{ts} = 0$
- **Special case 3**: general MCNF problem
  - add two nodes a and b $\ni$
    - arc (a,i) is added if $s_i > 0$
    - arc (i,b) is added if $s_i < 0$
  - Alternately, can add a single node ‘0’ with zero supply, with arcs $(i,0)$ if $s_i > 0$ and an arc $(0,i)$ if $s_i < 0$. The artificial arcs have cost $M > (n-1)C/2$ where C is the largest absolute arc cost in the original network (see Bertsekas, section 5.2)
- **Special case 4**: feasible circulation problem

  $\Rightarrow a_{ij} = 0, \forall i, j$
- **Special case 5**: shortest path problem
  - Add a return *arc*(t,s) with $b_{ts} = c_{ts} = 1$
  - For all other arcs, $b_{ij}=0,\ c_{ij}=\infty$ and a_{ij} is given
  - To find shortest paths to all nodes from s, add return arcs $(i,s),\ i \neq s$ $\ni b_{is} = c_{is} = 1$

MCCP and Transportation Problem

- **Special case 6**: transportation or Hitchcock-Koopman's problem



$$\left.\begin{aligned} &\min \textstyle\sum_i \sum_j x_{ij} a_{ij} \\ &\text{s.t. } \textstyle\sum_{j=1}^{n} x_{ij} \le c_i; \quad i = 1, \ldots, m \\ &\qquad \textstyle\sum_{i=1}^{m} x_{ij} \le b_i; \quad j = 1, \ldots, n \\ &\qquad 0 \le x_{ij}; \quad \textstyle\sum b_j = \sum c_i \end{aligned}\right\} \text{special case of generalized min. cost flow problem}$$

Dual of the minimum cost circulation problem

- **Special case 7**: assignment problem (or) weighted bipartite matching problem … lecture 8
  - LP formulation

$$\begin{aligned} \min & \sum_i \sum_j x_{ij} a_{ij} \\ \text{s.t.} & \sum_j x_{ij} = 1 \\ & \sum_i x_{ij} = 1 \\ & 0 \le x_{ij} \le 1 \end{aligned}$$

- Dual of the minimum cost circulation problem

$$\underline{\textit{Primal}} \qquad \begin{aligned} \min & \sum_i \sum_j x_{ij} a_{ij} \\ \text{s.t.} & \sum_j x_{ij} - \sum_k x_{ik} = 0 \\ & -x_{ij} \ge -c_{ij} \\ & x_{ij} \ge b_{ij} \end{aligned}$$

$$\Rightarrow$$

$$\underline{\textit{Dual}} \qquad \begin{aligned} \max & \sum_i \sum_j b_{ij} \beta_{ij} - \sum_i \sum_j c_{ij} \mu_{ij} \\ \text{s.t.} & -\lambda_i + \lambda_j + \beta_{ij} - \mu_{ij} \le a_{ij} \\ & \beta_{ij} \ge 0 \\ & \mu_{ij} \ge 0 \end{aligned}$$

Optimality – KILTER conditions

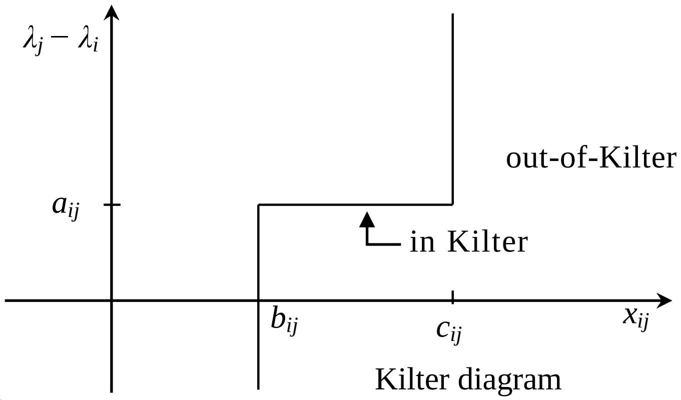
- Optimality conditions or complementary slackness or Karush-Kuhn-Tucker or **KILTER** conditions

$$
\begin{aligned}
&x_{ij} > 0 \;\Rightarrow -\lambda_i + \lambda_j + \beta_{ij} - \mu_{ij} = a_{ij} \\
&\beta_{ij} > 0 \;\Rightarrow x_{ij} = b_{ij} \\
&\mu_{ij} > 0 \;\Rightarrow x_{ij} = c_{ij}
\end{aligned}
$$

- Alternatively,

$$
\left.\begin{aligned}
&x_{ij} = b_{ij} \Rightarrow \mu_{ij} = 0 \;\&\; \beta_{ij} \ge 0 \Rightarrow \lambda_j - \lambda_i \le a_{ij} \\
&b_{ij} < x_{ij} < c_{ij} \Rightarrow \mu_{ij} = \beta_{ij} = 0 \Rightarrow \lambda_j - \lambda_i = a_{ij} \\
&x_{ij} = c_{ij} \Rightarrow \beta_{ij} = 0 \;\&\; \mu_{ij} \ge 0 \Rightarrow \lambda_j - \lambda_i \ge a_{ij}
\end{aligned}\right\}\text{known as KILTER conditions}
$$

  - λ_i = (–price of node *i*) or (–Lagrange multiplier) or dual variable



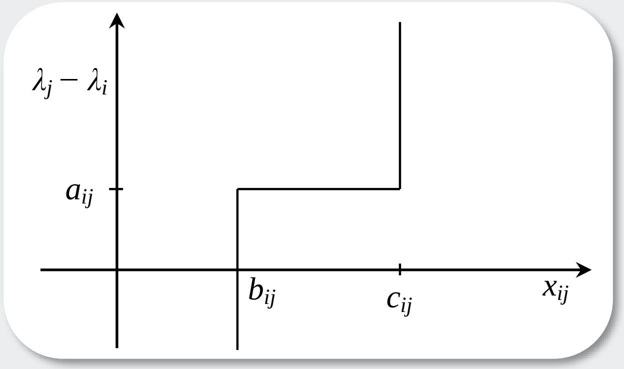
Kilter diagram

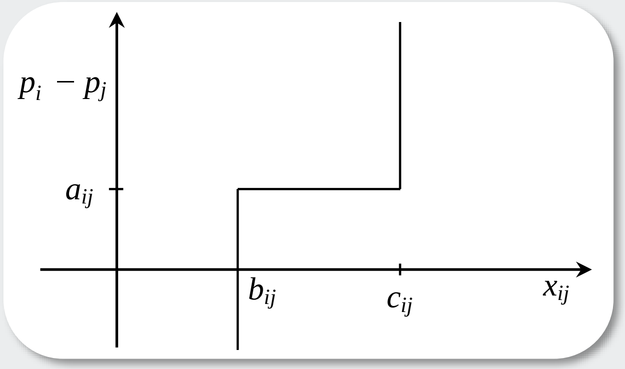
MCNF Algorithms

- **Primal simplex methods**
  - Non-zero feasible flow pattern (bfs) is a **spanning tree***
  - **Pivoting**: how to go from one bfs to another, while reducing primal cost?
    - Add an edge to the tree that creates a negative cost cycle
    - Remove the edge from the current tree to get back the spanning tree (next bfs)
  - Software: RNET: The Rutgers Minimum Cost Network Flow Subroutines (Grigoriadis, 1980); NETFLO: Algorithms for Network Programming (Kennington & Helgason, 1980)
- **Primal-dual methods**
  - Start with (x,p) pair satisfying CS conditions and maintains the CS property throughout
  - Out-of-kilter methods: shortest augmenting path methods
  - RELAX: Relaxation Algorithm for network flows (Bertsekas & Tseng, 1985-1991)
  - ϵ-RELAX ~ Auction-like algorithm (Bertsekas and Eckstein, 1988)
- For problems with lower and upper bound constraints, also need to keep track of arcs with flows at lower bound and arcs with flows at upper bound (see Bertsekas' book, section 5.3)

Primal-dual algorithms for MCNF

- **Most successful methods employ primal-dual concepts**
  - RELAX
  - ϵ-RELAX
    - **RELAX is at least 4 – 10 times faster than out-of-Kilter**
    - **ϵ-RELAX is parallelizable**
- RELAX and ϵ-RELAX algorithms are based on primal-dual concepts
  - Assume: a_{ij}, b_{ij} & c_{ij} are integers
    - No loss in generality, since can scale rational numbers so that they are integers
    - If irrational, scale numbers to machine precision
  - If $p_i = -\lambda_i$, then Kilter conditions are of the form:





Lagrangian function

- Primal-dual from classical Lagrangian framework
  - Let us consider the dual formulation from the classical duality framework
    - Define Lagrange multipliers p_i for each of the n conservation of flow constraints
    - But, leave the lower and upper bound constraints on x_{ij} alone $\Rightarrow$ partial dualization
    - Lagrangian function

$$\begin{aligned}
\min L(\underline{x}, \underline{p}) &= \textstyle\sum_{(i,j)\in E}(a_{ij} + p_j - p_i)x_{ij} = \sum_{(i,j)\in E}(a_{ij} - t_{ij})x_{ij} \\
\text{s.t.} \quad & b_{ij} \le x_{ij} \le c_{ij}, \forall (i,j) \in E \\
& t_{ij} = p_i - p_j \\
& \quad = \text{price of node } i - \text{price of node } j \\
& \quad = \text{tension of edge } (i,j)
\end{aligned}$$

Dual of MCNF

- Dual problem is:

$$\max_{\underline{p}} \quad q(\underline{p})$$

s.t. no constraint on $p_i \Rightarrow p_i \geq \text{ or } \leq 0$

where

$$\begin{aligned} q(\underline{p}) &= \min_{b_{ij} \leq x_{ij} \leq c_{ij}} L(\underline{x}, \underline{p}) \\ &= \sum_{(i,j)\in E} \min_{b_{ij} \leq x_{ij} \leq c_{ij}} (a_{ij} + p_j - p_i) x_{ij} \\ &= \sum_{(i,j)\in E} \min_{b_{ij} \leq x_{ij} \leq c_{ij}} (a_{ij} - t_{ij}) x_{ij} = \sum_{(i,j)\in E} q_{ij}(t_{ij}) \end{aligned}$$

where

$$q_{ij}(t_{ij}) = \min_{b_{ij} \leq x_{ij} \leq c_{ij}} \left(a_{ij} - t_{ij}\right) x_{ij}$$

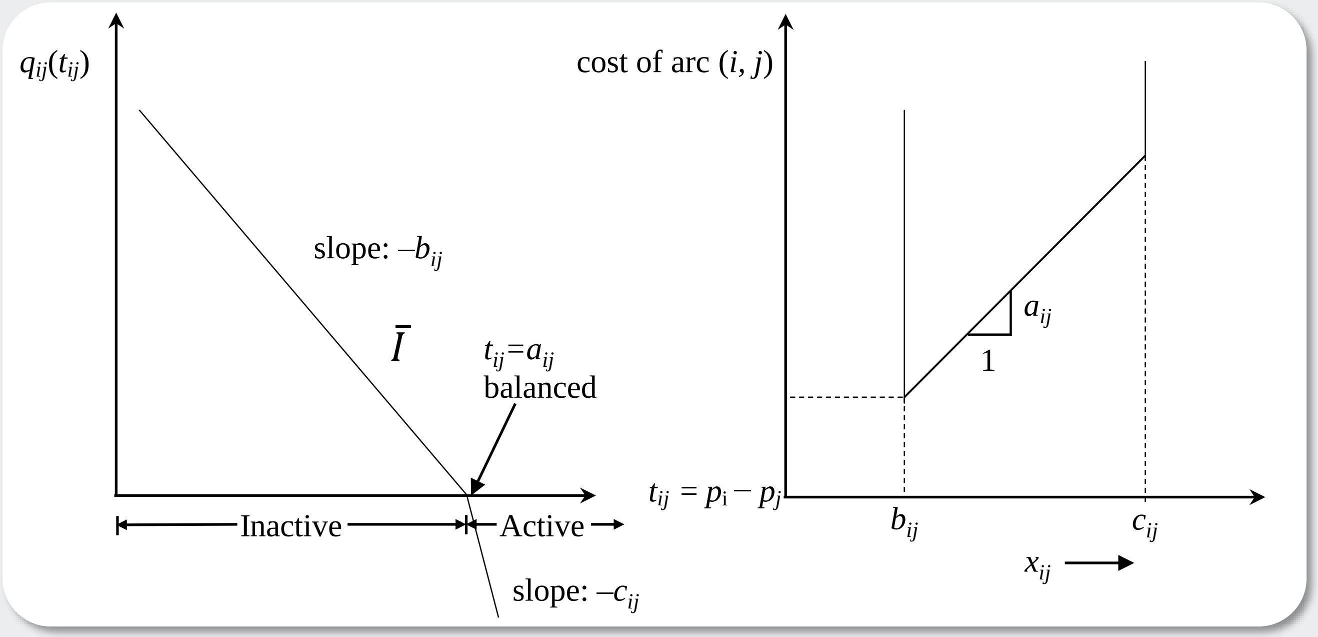
$$\Rightarrow x_{ij}^*(t_{ij}) = \begin{cases} b_{ij} & \text{if } a_{ij} > t_{ij} \\ \text{any } x_{ij} \in \{b_{ij}, c_{ij}\} & \text{if } a_{ij} = t_{ij} \\ c_{ij} & \text{if } a_{ij} < t_{ij} \end{cases}$$

- Observations
  - ❖ $L(\underline{x}, \underline{p})$ and $q(\underline{p})$ are separable in network edges

Inactive, Balanaced and Active arcs

- For a given p_i, p_j (or equivalently, t_{ij}), $q_{ij}(t_{ij})$ is easily evaluated via scalar minimization

$$q_{ij}(t_{ij}) = \begin{cases} (a_{ij} - t_{ij})b_{ij} & \text{if } t_{ij} \le a_{ij} \Rightarrow p_i \le p_j + a_{ij} \\ (a_{ij} - t_{ij})c_{ij} & \text{if } t_{ij} \ge a_{ij} \Rightarrow p_i \ge p_j + a_{ij} \end{cases}$$



- For a given p_i, p_j (or equivalently, $t_{ij} = p_i - p_j$), we say that arc (i, j) is:
  - Inactive if $t_{ij} < a_{ij}$ (or) $p_i < p_j + a_{ij}$
  - Balanced if $t_{ij} = a_{ij}$ (or) $p_i = p_j + a_{ij}$
  - Active if $t_{ij} > a_{ij}$ (or) $p_i > p_j + a_{ij}$

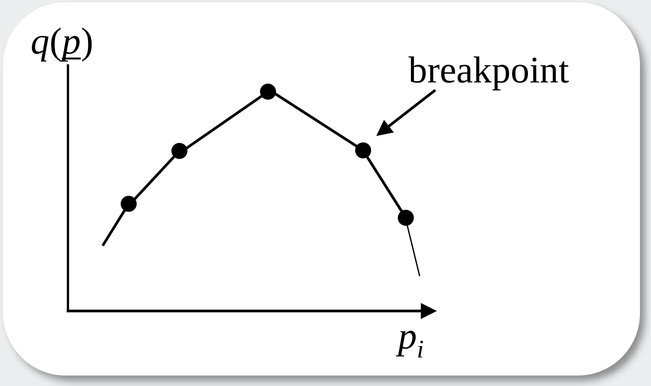
Arc status and CS conditions

- What do CS conditions mean in terms of inactive, balanced & active arcs?
  - Inactive $\Rightarrow x^*_{ij} = b_{ij}$
  - Balanced $\Rightarrow b_{ij} \leq x^* \leq c_{ij}$
  - Active $\Rightarrow x^*_{ij} = c_{ij}$

  $\Rightarrow$ and, of course, all conservation constraints are satisfied
- **Deficit of a node *i***: export – import

$$d_i = \underbrace{\sum_{(i,j)\in E} x_{ij}}_{\text{outflow}} - \underbrace{\sum_{(j,i)\in E} x_{ji}}_{\text{inflow}}, \quad \forall i \in V$$

  - At optimality: $d_i = 0, \forall i \in V$
  - Also, note that $\sum_{i\in V} d_i = 0, \forall$ flow
- Suppose we change p_i only
  - what does $q(\underline{p})$ look like?



- Breakpoints correspond to points where one or more edges incident to node *i* are balanced
- Only at breakpoints we can set x_{ij} in the range (b_{ij}, c_{ij})
- In the linear portion:

  $x_{ij} = c_{ij}$ if (i, j) is active

  $x_{ij} = b_{ij}$ if (i, j) is inactive

Idea: Adjust flows and prices to reduce deficits

- Idea: suppose have a flow-price pair ($\underline{x}$, $\underline{p}$) ∋
  - Complementary slackness conditions are satisfied, but
  - Conservation of flow constraints need not be met
- If conservation constraints are satisfied ⇒ $d_i = 0$

  ⇒ Solution ($\underline{x}$, $\underline{p}$) is optimal, since **simultaneous primal and dual feasibility implies optimality**
- Otherwise, ∃ nodes such that $d_i > 0$ for some nodes & $d_i < 0$ for some other nodes, since $\Sigma_i d_i = 0$
- So, we need to work towards $d_i = 0, \forall i \in V$
  - If $d_i < 0$ (⇒ export < import … U.S.A.), we can do the following:
    - Look at **balanced unsaturated outgoing edges**, i.e., those edges with $x_{ij} < c_{ij}$ (or unsaturated edges)
      - ❖ Increase the flow on outgoing edges

$$x_{ij} = x_{ij} + \overbrace{\min\{-d_i, d_j, c_{ij} - x_{ij}\}}^{\delta}$$
$$d_i = d_i + \delta, \quad d_j = d_j - \delta$$

  - Equivalently, look at **balanced incoming edges**, i.e., those with $x_{ji} > b_{ji}$, i.e., edges with flow > minimal flow,
    - Decrease the flow along these edges

$$x_{ji} = x_{ji} - \overbrace{\min\{-d_i, d_j, x_{ji} - b_{ji}\}}^{\delta}$$
$$d_i = d_i + \delta, \quad d_j = d_j - \delta$$

  - Note:
    - We perform increase and decrease operations ∋ capacity constraints are not violated
    - p_i did not change

Flow adjustment for surplus nodes

- If increase and decrease operations make $d_i = 0$, we are done
- Otherwise, the result is:
  - Active - **all outgoing balanced edges at their maximum flow**, c_{ij}
  - Inactive - **all incoming balanced edges at their minimum flow**, b_{ji}
- **If new d_i is still < 0, we must make an inactive outgoing edge balanced**
  - **This we can do by increasing p_i until the next breakpoint, i.e., until another edge becomes balanced**
  - Repeat the whole process again (up iteration, equivalently, $t_{ij} \uparrow$ or price adjustment step)
- If $d_i > 0$ ($\Rightarrow$ import < export or have surplus … China, Japan, Mexico), we can do the opposite

  $\Rightarrow$ Increase imports & decrease exports
  - Look at **inactive & balanced incoming edges** (j, i)
  - Those edges with $x_{ji} < c_{ji}$, and
  - Look at **active & balanced outgoing edges** $(i, j) \Rightarrow$ those edges with $x_{ij} > b_{ij}$
  - Find by how much we can increase imports & decrease exports by operating each incoming edge at its capacity (upper bound) & outgoing edge at its lower bound … let us denote this by ΔC
- If $\Delta C > 0$
  - Reduce price of $p_i \ni$ one of the active outgoing edges become balanced or an inactive incoming edge becomes balanced
  - Set flows on all incoming balanced arcs (including the new one) at maximum & set flows on all outgoing balanced arcs (including the new one) at minimum

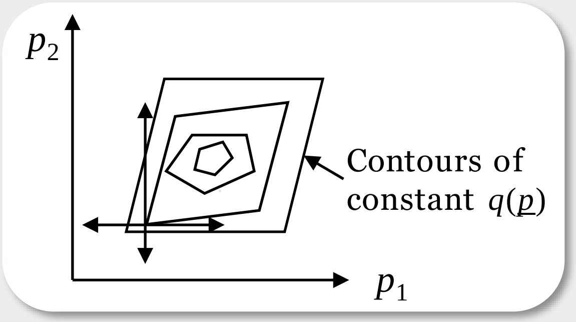
Jamming problem

- Otherwise if any adjacent node j has $d_j < 0$

$$\text{flow adjustment}\begin{cases}\text{if } (j,i) \text{ is an incoming arc}: & x_{ji} = x_{ji} + \overbrace{\min\{d_i, -d_j, c_{ji} - x_{ji}\}}^{\delta} \\ & d_i = d_i - \delta, \quad d_j = d_j + \delta \\ \text{if } (i,j) \text{ is an outgoing arc}: & x_{ij} = x_{ij} - \overbrace{\min\{d_i, -d_j, x_{ij} - b_{ij}\}}^{\delta} \\ & d_i = d_i - \delta, \quad d_j = d_j + \delta\end{cases}$$

$\Rightarrow$ **coordinate ascent methods (or one variable at a time adjustment)!!**

- Problem with the approach – jamming
  - Because $q(\underline{p})$ is sum of non-differentiable concave functions
  - Can't change p_1 or p_2 to improve
  - Use more complicated directions $\Rightarrow$ RELAX
  - Use ϵ-relaxation $\Rightarrow$ ϵ-relaxation algorithms
    - Move, even if dual cost decreases, but only by a small $\pm\,\epsilon$ beyond maximum in a direction
    - Similar to auction for the assignment problem



Basic ideas of RELAX

- Choose a starting flow - price pair $(\underline{x}, \underline{p})$ satisfying complementary slackness
  - Example: $\underline{p} = \underline{0}$ and $x_{ij} = b_{ij}$ for $a_{ij} > 0$
- Repeat until $\nexists$ a node with positive deficit
  - Obtain a new $(\underline{x}, \underline{p})$ by carrying out
    - Single node price adjustment (or)
    - Single node flow adjustment (or)
    - Multiple node price (or) flow adjustment
- Single node price adjustment
  - We will describe an algorithm for $d_i > 0$ … think of Japan, China or Mexico
  - The algorithm for $d_i < 0$ is similar … think of U.S.A.
  - Suppose have a node with $d_i < 0$
    - You must have one as long as $d_i \neq 0, \forall i$
    - Recall that $\Sigma_i \, d_i = 0 \; \forall$ flow
  - Q: what is the maximum price change I can have at node i w/o violating the complementary slackness conditions?

Formalization of RELAX

- $d_i > 0 \Rightarrow$ import <export … China, Japan, Mexico
- We can reduce d_i by increasing flows on incoming edges & reducing flows on outgoing edges
- Need outgoing edges (i, j) and incoming edges (j, i) ∋ $$x_{ij} > b_{ij} \Rightarrow p_i \geq a_{ij} + p_j$$ $$x_{ji} < c_{ji} \Rightarrow p_j \leq p_i + a_{ji}$$
- Let $$B_i^+ = \left\{ j : (i, j) \text{ is balanced, } x_{ij} > b_{ij} \right\}$$ $$B_i^- = \left\{ j : (j, i) \text{ is balanced, } x_{ji} < c_{ji} \right\}$$

⇒ So, for an outgoing edge $(i, j) \in B_i^+ \cup \{active\}$, we can reduce p_i to at most $(p_j + a_{ij})$

⇒ For an incoming edge $(j, i) \in B_i^- \cup \{inactive\}$, we can reduce p_i to at most $(p_j - a_{ji})$ without violating C-S conditions

- Formally
$$\Rightarrow p_{inew} = \max\left\{ (p_j + a_{ij}) : \overbrace{(i, j) \in E, x_{ij} > b_{ij}}^{j \in B_i^+ \cup (active)} ; (p_j - a_{ji}) : \overbrace{(j, i) \in E, x_{ji} > c_{ji}}^{j \in B_i^- \cup (inactive)} \right\}$$
- If $p_{inew} < p_i$ OK … what if $p_{inew} = p_i$?
    - We can still do it provided imports can be increased & exports decreased ∋ the residual capacity of all balanced incoming and outgoing edges are such that:
$$\delta_i = \sum_{j \in B_i^-} \left(c_{ji} - x_{ji} \right) + \sum_{j \in B_i^+} \left(x_{ij} - b_{ij} \right) \leq d_i$$

⇒ All incoming balanced edges at their capacity & all outgoing balanced edges at their minimum can't resolve deficit

Single node price adjustment algorithm

- Single node price adjustment algorithm
  - Start with a node with $d_i > 0$
  - Compute p_{inew} & set $p_i = p_{inew}$
  - If $\delta_i \le d_i$ then
    - $d_i = d_i - \delta_i$
    - $x_{ji} = c_{ji}, \ \forall j \in B_i^-$
    - $x_{ij} = b_{ij}, \ \forall j \in B_i^+$
  - Else terminate
- What does it all mean?
  - Consider

$$q(\underline{p}) = \sum_{(i,j)\in E} q_{ij}(t_{ij}) = \sum_{(i,j)\in E} q_{ij}(p_i - p_j)$$

  - Suppose want to reduce p_i

    $\Rightarrow \underline{p} = \underline{p} - \alpha \underline{e}_i; \ \underline{e}_i = i$th unit vector
  - What is the directional derivative?

Directional derivative

$$\Delta C(\underline{e}_i,\underline{p}) = \sum_{(k,j)\in E} \lim_{\alpha\to 0} \frac{\left[q_{kj}(p_k-\alpha\delta_{ik}-p_j)-q_{kj}(p_k-p_j)\right]}{\alpha}$$

$$= \sum_{(k,j)\in E} r_{kj}(\underline{e}_i,\underline{p})$$

$$r_{kj}(\underline{e}_i,\underline{p}) = \begin{cases} 0 & \text{if } k\neq i \text{ or } j\neq i \\ b_{ij} & \text{if } k=i \,\&\, (i,j) \text{ is balanced or inactive} \\ c_{ij} & \text{if } k=i \,\&\, (i,j) \text{ is active} \\ -b_{ki} & \text{if } j=i \,\&\, (k,i) \text{ is inactive} \\ -c_{ki} & \text{if } j=i \,\&\, (k,i) \text{ is balanced or active} \end{cases}$$

$$\Rightarrow \Delta C(\underline{e}_i,\underline{p}) = \underbrace{\sum_{\substack{(i,j)\\ \text{active}}} c_{ij} + \sum_{\substack{(i,j)\\ \text{balanced or}\\ \text{inactive}}} b_{ij}}_{\text{Outflow}} - \underbrace{\sum_{\substack{(k,i)\\ \text{balanced or}\\ \text{active}}} c_{ki} - \sum_{\substack{(k,i)\\ \text{inactive}}} b_{ki}}_{\text{Inflow}}$$

- When inactive edges are at LB, active edges at UB, incoming balanced edges at UB, and outgoing balanced edges at LB

$$= d_i - \sum_{k\in B_i^-}(c_{ki}-x_{ki}) - \sum_{k\in B_i^+}(x_{ik}-b_{ik}) = d_i - \delta_i$$

- Proof: use the facts that active $\Rightarrow x_{ij} = c_{ij}$, inactive $\Rightarrow x_{ij} = b_{ij}$
- So, we can improve $q(\underline{p})$ only if $d_i - \delta_i \geq 0$ … if $d_i = \delta_i$, dual value will be same, but deficit $d_{inew} = 0$
- We have used essentially this idea for the assignment problem
- We can extend this idea to multiple directions to avoid jamming

Single node flow adjustment

- Very simple
  - If $d_i > 0$, reduce outgoing flows or increase incoming flows
- Single node flow adjustment algorithm
  - Start with a node i having $d_i > 0$ …. surplus
  - Repeat
    - Choose a node $j \ni j \in B_i^-$ (incoming balanced edge) & $d_j < 0$ … deficit (or)
    - A node $j \ni j \in B_i^+$ (outgoing balanced edge) & $d_j < 0$
  - If $j \in B_i^-$
    - $\delta = \min\{-d_j, d_i, c_{ji} - x_{ji}\}$
    - $d_i = d_i - \delta$
    - $d_j = d_j + \delta$
    - $x_{ji} = x_{ji} + \delta$
    - If $d_i = 0$ terminate, else go to repeat
  - Else if $j \in B_i^+$
    - $\delta = \min\{-d_j, d_i, x_{ij} - b_{ij}\}$
    - $d_i = d_i - \delta$
    - $d_j = d_j + \delta$
    - $x_{ij} = x_{ij} - \delta$
    - if $d_i = 0$ terminate, else go to repeat
  - Else terminate
- p_i^s do not change in a single node flow adjustment step
- What can go wrong with single node price & flow adjustment procedures?

UCONN

Price adjustment procedure

- Price adjustment
  - $p_i \longrightarrow p_{inew}$ & $d_{inew} = 0$
    $\Rightarrow$ Can't work with node i
  - $p_i \longrightarrow p_{inew}$ & $d_{inew} > 0$, but directional derivative $d_i - \delta_i < 0$
    $\Rightarrow$ Can't adjust p_{inew} $\Rightarrow$ jammed
  - In the latter case, we can attempt a flow adjustment procedure

    . . . then, the resulting possibilities are:
    - $d_i = 0 \Rightarrow$ can't work with node i
    - $d_i > 0$ & $d_j > 0,\ \forall j \in B_i^- \cup B_i^+$

      $\Rightarrow$ "All rich neighbors"

      $\Rightarrow$ We are stuck and can't move
    - So, need alternate mechanisms in case when:

      $d_i > 0,\ d_j > 0,\ \forall j \in B_i^- \cup B_i^+$ & $d_i - d_j < 0$

Directional derivative for complex price adjustments

- Idea: Employ more complex ascent procedures than the simple coordinate ascent scheme
  - Constructing complex ascent steps
    - Suppose want to change prices at a subset S of nodes ($S \neq V$)
    - Consider the cut separating S and $V - S$
    - These are the incoming and outgoing edges from $V-S$ to S and S to $V-S$, respectively
    - Suppose want to decrease prices at all nodes $i \in S \Rightarrow$ $\underline{p}_{new} = \underline{p} - \alpha \sum \underline{e}_i = \underline{p} + \alpha \underline{v}$
    - Then, the directional derivative of $q(\underline{p})$ along v is:

$$\boxed{\Delta C(\underline{v}, \underline{p}) = \sum_{(k,j)\in E} r_{kj}(\underline{v}, \underline{p}) = \sum_{(k,j)\in E} r_{kj}(S, \underline{p}) = \Delta C(S, \underline{p})}$$

    - Since $\exists$ a one-to-one relationship between S and $\underline{v}$

$$r_{kj}(S, \underline{p}) = \begin{cases} 0 & \text{if } k, j \in S \text{ or } k, j \in V - S \\ b_{ij} & \text{if } k = i, i \in S \;\&\; j \in V - S \;\&\; (i, j) \text{ is balanced or inactive} \\ c_{ij} & \text{if } k = i, i \in S \;\&\; j \in V - S \;\&\; (i, j) \text{ is active} \\ -b_{ki} & \text{if } j = i, i \in S \;\&\; k \in V - S \;\&\; (k, i) \text{ is active} \\ -c_{ki} & \text{if } j = i, j \in S \;\&\; k \in V - S \;\&\; (k, i) \text{ is balanced or active} \end{cases}$$

RELAX steps -1

$$\Delta C(S,\underline{p}) = \sum_{\substack{i\in S\\ j\in V-S\\ (i,j)\text{ active}}} c_{ij} + \sum_{\substack{i\in S\\ j\in V-S\\ (i,j)\text{ balanced}\\ \text{or inactive}}} b_{ij} - \sum_{\substack{k\in V-S\\ i\in S\\ (k,i)\text{ inactive}}} b_{ki} - \sum_{\substack{k\in V-S\\ i\in S\\ (k,i)\text{ balanced}\\ \text{or active}}} c_{ki}$$

$$= \sum_{i\in S} d_i - \sum_{\substack{i\in S\\ j\in V-S\\ (i,j)\text{ balanced}}} (x_{ij} - b_{ij}) - \sum_{\substack{i\in V-S\\ j\in S\\ (j,i)\text{ balanced}}} (c_{ji} - x_{ji})$$

- Again if $\Delta C(S, \underline{p}) < 0 \Rightarrow$ can't perform ascent step
  - ❖ But can perform flow adjustment if $\exists$ a node $j \ni d_j < 0$ & $j \in S$
- In fact, we can embed the single node & multiple node price/flow adjustments into an overall algorithm as follows:
- One iteration of overall RELAX
  - ❖ Inputs $(\underline{x}, \underline{p})$; outputs $(\underline{x}, \underline{p})_{new}$

  <u>Step 1</u>: Choose a node i with $d_i > 0$ (a similar procedure for $d_i < 0$)

  - ❖ If no such node, terminate ($\Rightarrow$ optimum)
  - ❖ Label node i by $'O'$ . . . set $S = \emptyset$

  <u>Step 2</u>: Choose a labeled but unscanned node k (initially, $k = i$) . . . $S = S \cup \{k\}$

RELAX steps - 2

<u>Step 3</u>: Scan the label of node k as follows:

- Give the label "k" to all unlabeled nodes $m \ni$:

  $B_k^- = \{m : (m,k)$ is balanced & $x_{mk} < c_{mk}\}$

  $B_k^+ = \{m : (k,m)$ is balanced & $x_{mk} > b_{mk}\}$

- If $\underline{v}$ is the corresponding vector to S and $\Delta C(\underline{v}, \underline{p}) > 0$

  Go to Step 5

- Else if any of the labeled nodes m is $\ni d_m < 0$

  Go to Step 4

- Else

  Go to Step 2

<u>Step 4</u>: (flow augmentation step)

- A directed path p from i with $d_i > 0$ to a node m with $d_m < 0$ has been found
- What we want to do is either increase incoming flow to i from m or reduce outgoing flow from i to m; To do this define:

  $P^+ = \{(k, n) \in P : (k, n)$ is oriented in the direction from i to $m\}$

  $P^- = \{(k, n) \in P : (k, n)$ is oriented in the direction from m to $i\}$

RELAX steps - 3

- If $b_{kn} < x_{kn}$, we can reduce outgoing flow
- If $x_{kn} < c_{kn}$, we can increase incoming flow so,

$$\delta = \min\left\{d_i, -d_m, \{x_{kn} - b_{kn} : (k,n) \in P^+\}, \{c_{kn} - x_{kn} : (k,n) \in P^-\}\right\}$$

$$\Rightarrow x_{kn} = \begin{cases} x_{kn} - \delta \text{ if } (k,n) \in P^+ \\ x_{kn} + \delta \text{ if } (k,n) \in P^- \end{cases}$$

- Go to next iteration

<u>Step 5</u>: Let

$$\delta = \max\begin{Bmatrix} \{p_k - p_m - a_{km} \ : k \in S; m \in V - S; (k,m) \text{ active}\}, \\ \{p_k + a_{mk} - p_m \ : k \in S, \ m \in V - S; \ (m,k) \text{ inactive}\} \end{Bmatrix}$$

Set

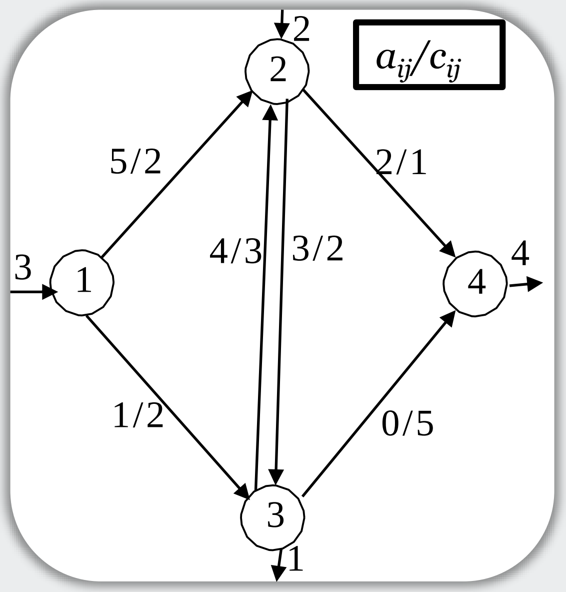
$$x_{km} = b_{km}, \forall \text{balanced arcs} \ni k \in S, \ m \in L, \ m \in V - S$$
$$x_{km} = c_{km}, \forall \text{balanced arcs} \ni k \in S, \ m \in L, \ m \in V - S$$

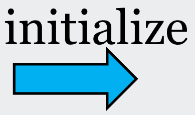
where L is the set of labeled nodes

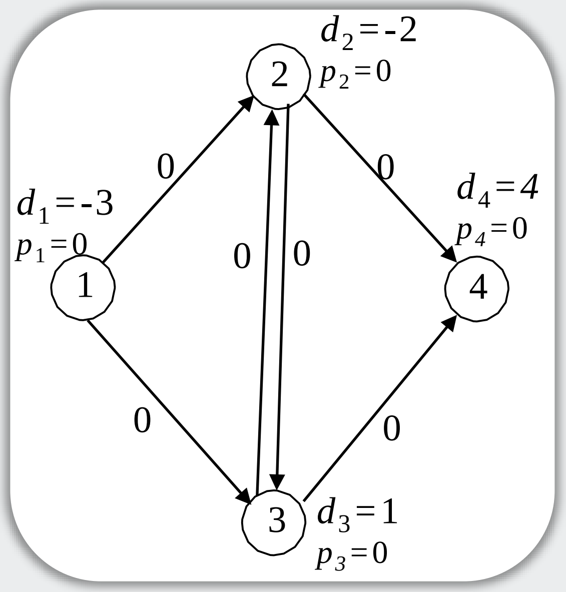
- Set
$$p_{knew} = \begin{cases} p_k - \delta & \text{if } k \in S \\ p_k & \text{if } k \in V - S \end{cases}$$
- Go to next iteration

Note: Need to initiate iterations with both $d_i > 0$ & $d_i < 0$

Illustration of RELAX - 1







- Select node 1: step 5: δ = max(-1,-5)= *-1; $p_1 = p_1 - \delta = 1$; $p_2 = p_3 = p_4 = 0$*
- Select node 1 again: *step 5: $x_{13}=2$; d_1*=-1; δ = max(1-5)= *-4; $p_1 = p_1 - \delta = 5$; $p_2 = p_3 = p_4 = 0$*
- Nodes 1 and 2 will give ascent direction: δ =-2*; $p_1 = p_1 - \delta = 7$; $p_2 = p_2 - \delta = 2$; $p_3 = p_4 = 0$*
  - Easier to see δ from the ΔC equation on slide 25
- Flow augmentation along path 1-2-4: *$x_{12}=1$; $x_{24}=1$; $d_1=0$; d_4*=-3
- Nodes 1 and 2 will give ascent direction: δ =-2*; $p_1 = p_1 - \delta = 9$; $p_2 = p_2 - \delta = 4$; $p_3 = p_4 = 0$*
- Flow augmentation along path (2,3,4): *$x_{23}=2$; $x_{34}=2$; $d_2=0$; d_4*=-1
- Flow augmentation along edge (3,4): *$x_{34}=3$; $d_3=0$, $d_4=0$; Done !*

Performance of RELAX

- Practical details
  - Need to initiate iterations with both $d_i > 0$ & $d_i < 0$
  - Can convert problem $\ni b_{ij} = 0$
    - Simply use:

$$\hat{x}_{ij} = x_{ij} - b_{ij} \;\&\; \hat{c}_{ij} = c_{ij} - b_{ij} \quad \Rightarrow \min \sum_i \sum_j \hat{x}_{ij} a_{ij}$$

$$\text{s.t.} \quad \sum_{(i,j)} \hat{x}_{ij} - \sum_{(j,i)} \hat{x}_{ji} = \hat{s}_i, \quad \forall i \in V$$

$$0 \le \hat{x}_{ij} \le c_{ij} - b_{ij}$$

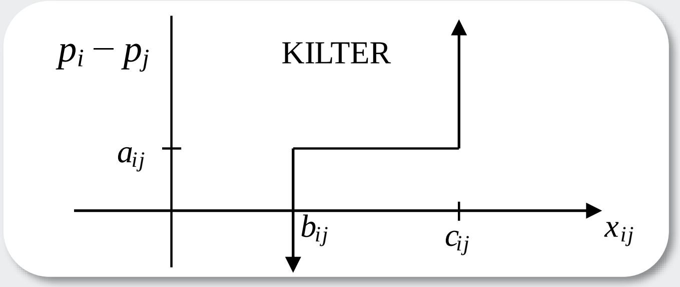
  - Use linked lists to represent data
    - [start node, end node, cap., $a_{ij} + p_j - p_i$, x_{ij} , next arc with same start node, next arc with the same end node]

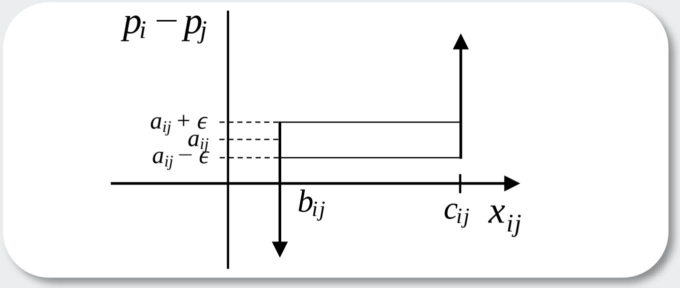
- Computational comparison

Problem	*RELAX*	*Out-of-Kilter*	*RNET*
Transportation	37.32	251.85	96.22
Assignment	9.31	56.89	40.37
Uncapacitated & Highly Cap. Transportation	32.09	192.88	41.94
Large Uncap. Problems	295	2217	882

- ϵ-relaxation algorithm
  - Satisfy complementary slackness only approximately
  - Essentially to avoid executing multi-node price and flow adjustments of RELAX $\Rightarrow$ well-suited for distributed implementation

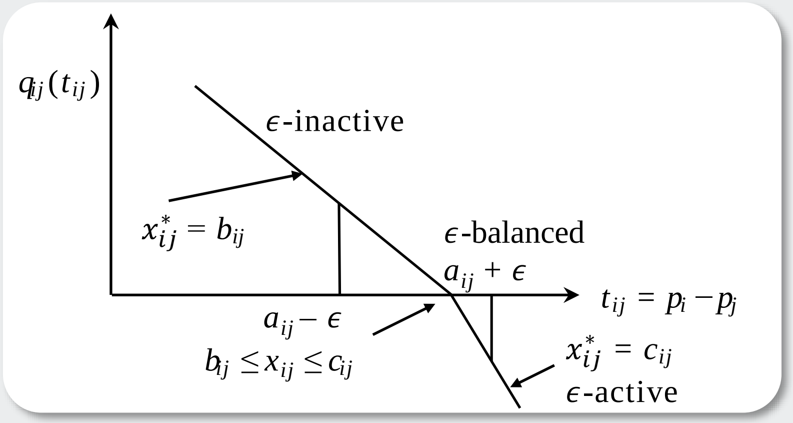
What does ε-relaxation mean?





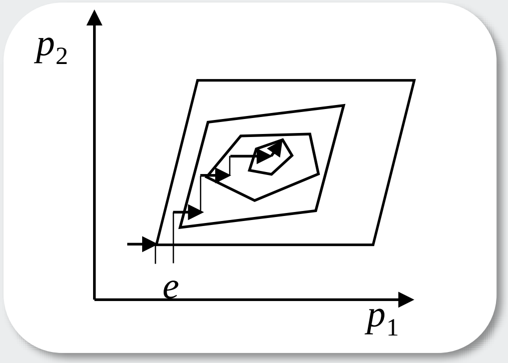
$\underline{CS}$		$\underline{\epsilon - CS}$
$x_{ij} < c_{ij} \Rightarrow p_i \le p_j + a_{ij}$	$\Leftrightarrow$	$p_i \le p_j + a_{ij} + \epsilon$
$x_{ij} > b_{ij} \Rightarrow p_i \ge p_j + a_{ij}$	$\Leftrightarrow$	$p_i \ge p_j + a_{ij} + \epsilon$
inactive $p_i < p_j + a_{ij}$	$\Leftrightarrow$	$p_i < p_j + a_{ij} - \epsilon$ ($\epsilon -$inactive)
balanced $p_i < p_j + a_{ij}$	$\Leftrightarrow$	$p_i = p_j + a_{ij} - \epsilon$ ($\epsilon^- -$balanced)
	$\Leftrightarrow$	$p_i = p_j + a_{ij} + \epsilon$ ($\epsilon^+ -$balanced)
active arc $\Rightarrow p_i > p_j + a_{ij}$	$\Leftrightarrow$	$p_i > p_j + a_{ij} + \epsilon$ ($\epsilon -$active)

so, $\epsilon -$balanced arc $\Rightarrow p_j + a_{ij} - \epsilon \le p_i \le p_j + a_{ij} + \epsilon$



Idea of ϵ-Relaxation Algorithm

- **Algorithm uses single node price and flow adjustment**



- **Basic result:** if $\epsilon < \frac{1}{n}$, (x_{ij}) is primal feasible and $(\underline{x}, \underline{p})$ satisfy ϵ-CS conditions, then (x_{ij}) is optimal (assuming a_{ij}, b_{ij}, & c_{ij} are integers $\Rightarrow x_{ij}$ is also integer)
- To be different, let us consider $d_i < 0$ case
  - $d_i < 0 \Rightarrow$ export < import $\Rightarrow$ USA
    - Increase flows on ϵ^+–balanced outgoing edges
    - Decrease flows on ϵ^-–balanced incoming edges
    - Increase price of node i . . . this is where ϵ-relaxation comes in

ϵ-Relaxation Algorithm

- Also:
  - ❖ $x_{ij} < c_{ij} \Rightarrow \epsilon$-balanced + ϵ-inactive
  - ❖ $x_{ij} > b_{ij} \Rightarrow \epsilon$-balanced + ϵ-active

▪ We use ϵ^+-balanced outgoing arcs $\Rightarrow (x_{ij} < c_{ij})$ & ϵ^--balanced incoming edges $(x_{ij} > b_{ij})$ . . . these are admissible arcs

$$\left.\begin{aligned} \epsilon^+\text{-balanced} &\Rightarrow x_{ij} < c_{ij} \\ \epsilon^-\text{-balanced} &\Rightarrow x_{ji} > b_{ji} \end{aligned}\right\} \quad \text{admissible arcs}$$

▪ Just as in max. flow, we can construct an admissible graph
  - (Residual graph) $G_x \ni \exists$ an (i, j), $\forall\epsilon^+$-balanced edge (i, j) with $x_{ij} < c_{ij}$ & a reverse edge (j, i), $\forall\epsilon^-$-balanced edge (j, i) with $x_{ji} > b_{ji}$
  - Store admissible graph as a forward star . . . outlist (or push list)

▪ Mechanization
  - Suppose i is a node with $d_i < 0$

    <u>Step 1</u>: Remove arcs from the top of the i's push list
    - ❖ If $d_i < 0$ and $\exists$ an edge (i, j) go to Step 2 (i.e., increase outgoing flow or exports)
    - ❖ If $d_i < 0$ and $\exists$ an edge (j, i) go to Step 3 (i.e., decrease incoming flow or imports)
    - ❖ If the push list is empty, go to Step 4 (price change)
    - ❖ If $d_i = 0$, but an edge was found, stop

Mechanization

<u>Step 2</u>: (Decrease deficit by increasing outflows)

- $\delta = \min\{-d_i, c_{ij} - x_{ij}\}, x_{ij} = x_{ij} + \delta, \ d_i = d_i + \delta \ , \ d_j = d_j - \ \delta$
- (note: d_j is not involved $\Rightarrow$ local computation)
- If $\delta = c_{ij} - x_{ij}$, delete (i, j) from the push list
- Go to Step 1

<u>Step 3</u>: (Decrease deficit by decreasing inflows)

- $\delta = \min\{-d_i, x_{ji} - b_{ji}\}, \ x_{ji} = x_{ji} - \delta, \ d_i = d_i + \delta, \ d_j = d_j - \delta$
- If $d = x_{ji} - b_{ji}$, delete (i, j) from the push list
- Go to Step 1

<u>Step 4</u>: (Scan / price increase)

- By scanning all vertices incident to i, set
  $p_i = \min\{\{p_j + a_{ij} + \epsilon: (i, j) \in E \ \& \ x_{ij} < c_{ij}\} \cup \{p_j - a_{ji} + \epsilon: (j, i) \in E \ \& \ x_{ji} > b_{ji}\}\}$
- Construct a new push list for i, containing exactly only those incident arcs which are admissible with the new value of p_i
- Go to Step 1

Mechanization

- Why node prices must increase?
  - When we enter Step 4 of the algorithm
    - $x_{ij} = c_{ij} \Rightarrow p_i \geq p_j + a_{ij} + \epsilon$
    - $x_{ji} = b_{ji} \Rightarrow p_i \geq p_j - a_{ji} + \epsilon$
    - When Step 4 is entered

      $p_i < \min\{p_j + a_{ij} + \epsilon: (i, j) \in E, x_{ij} < c_{ij}\}$

      $p_i < \min\{p_j - a_{ji} + \epsilon: (j, i) \in E, x_{ji} < b_{ji}\}$

      $\Rightarrow p_i$ must increase in Step 4
- If set is empty $\Rightarrow$ outgoing $x_{ij} = c_{ij}$, incoming $x_{ji} = b_{ji} \Rightarrow$ infeasible since $d_i < 0$
- If $d_k > 0$ at the beginning, p_k does not change throughout . . . this is because a node with $d_i < 0$ will continue to be part of up iterations

  $\Rightarrow d_i \uparrow 0 \leftrightarrow d_k \downarrow$
- Finite termination with complexity $O(n^3 \log nc)$; $c = \max_{(i,j)\in A} a_{ij}$
- Extremely slow compared to RELAX on sequential computers
- Asynchronous version of the algorithm converges
  - Proof similar to distributed Bellman-Ford's version of shortest path algorithm

Summary

- Minimum cost network flow problem
  - Special cases
  - Kilter Conditions
- Best algorithms based on primal-dual concepts
  - RELAX
  - ϵ-RELAX, slow but parallelizable