

# Separable Convex Cost Flow Problems

Valentinas Sungaila and Eric Olberding

# Convex Objective Function

- Min Cost Flow: minimize  $\sum c_{ij}x_{ij}$ .
- Convex Cost Flow: minimize  $\sum C_{ij}(x_{ij})$
- The cost on each arc,  $C_{ij}(x_{ij})$  is a convex (bath tub shaped) function of flow on that arc. We assume  $C_{ij}(0) = 0$ .
- Convex function:  $\theta \in [0,1]$   
$$C_{ij}(\theta x_{ij} + (1 - \theta)x'_{ij}) \leq \theta C_{ij}(x_{ij}) + (1 - \theta)C_{ij}(x'_{ij})$$

# Separable Cost

- The total cost is the sum of the cost on each arc.

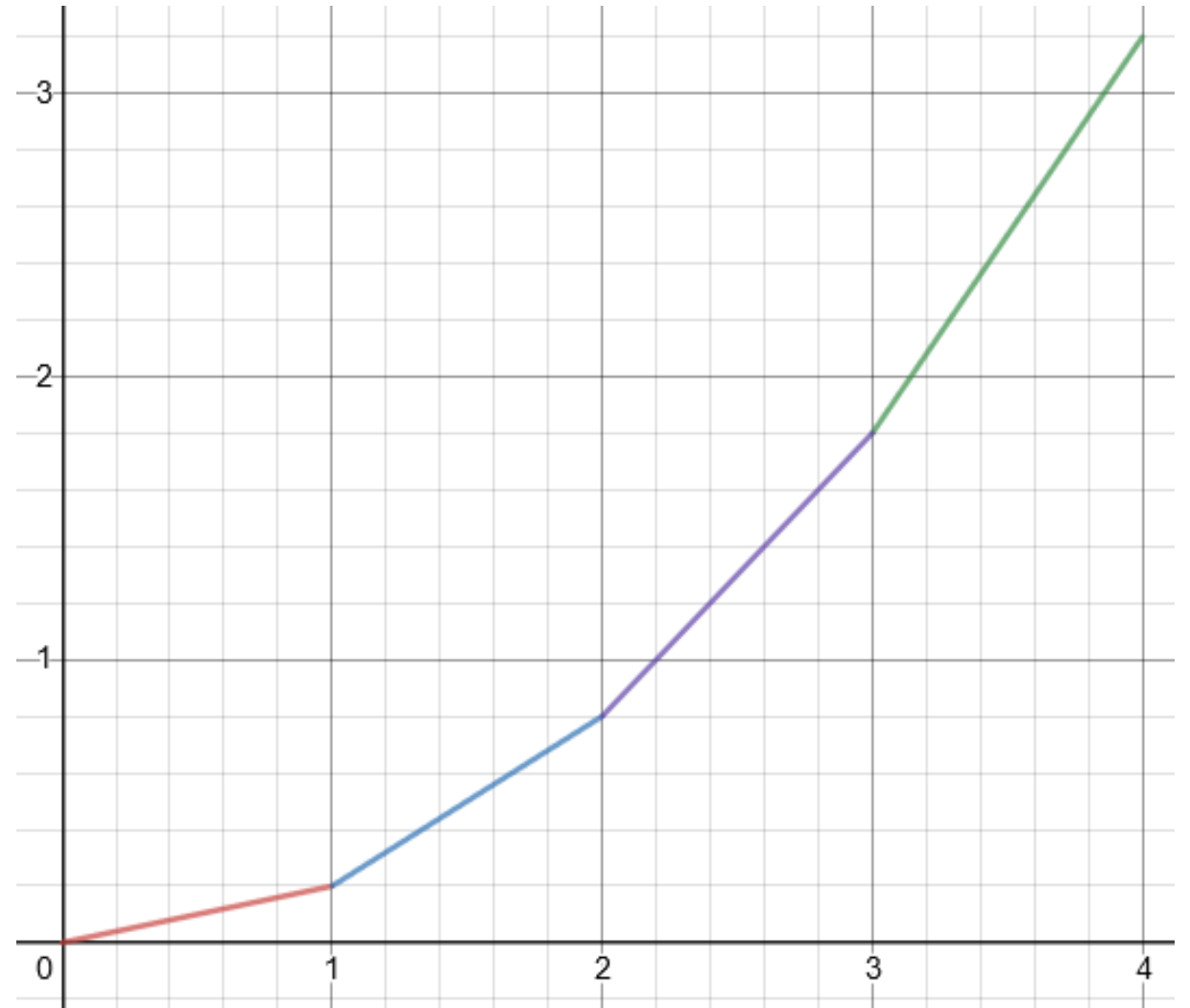
$$T = \sum C_{ij}(x_{ij})$$

- Example of inseparable cost for network with two arcs with cost  $c_1$  and  $c_2$  per unit flow.

$$T = (c_1x_1 + c_2x_2)^2$$

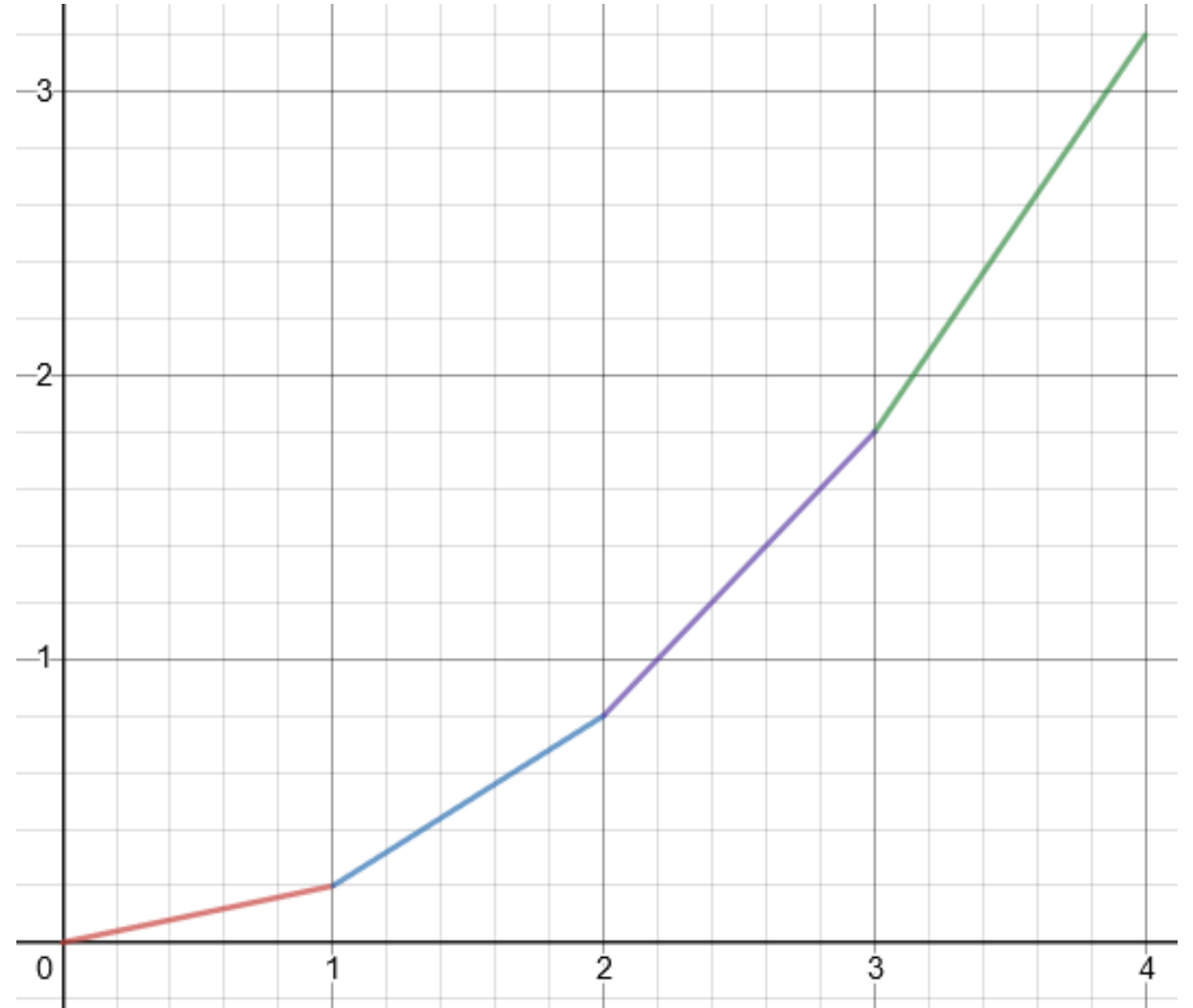
# Piecewise Continuous Linear Costs

- $y$  is cost,  $x$  is flow
- Continuous linear piece-wise functions.
  - arc cost specified with  $p$  pieces of information.
- Total information for cost will be total number of pieces for all arc cost functions
  - 4 pieces of information for this arc



# Piecewise Continuous Linear Costs

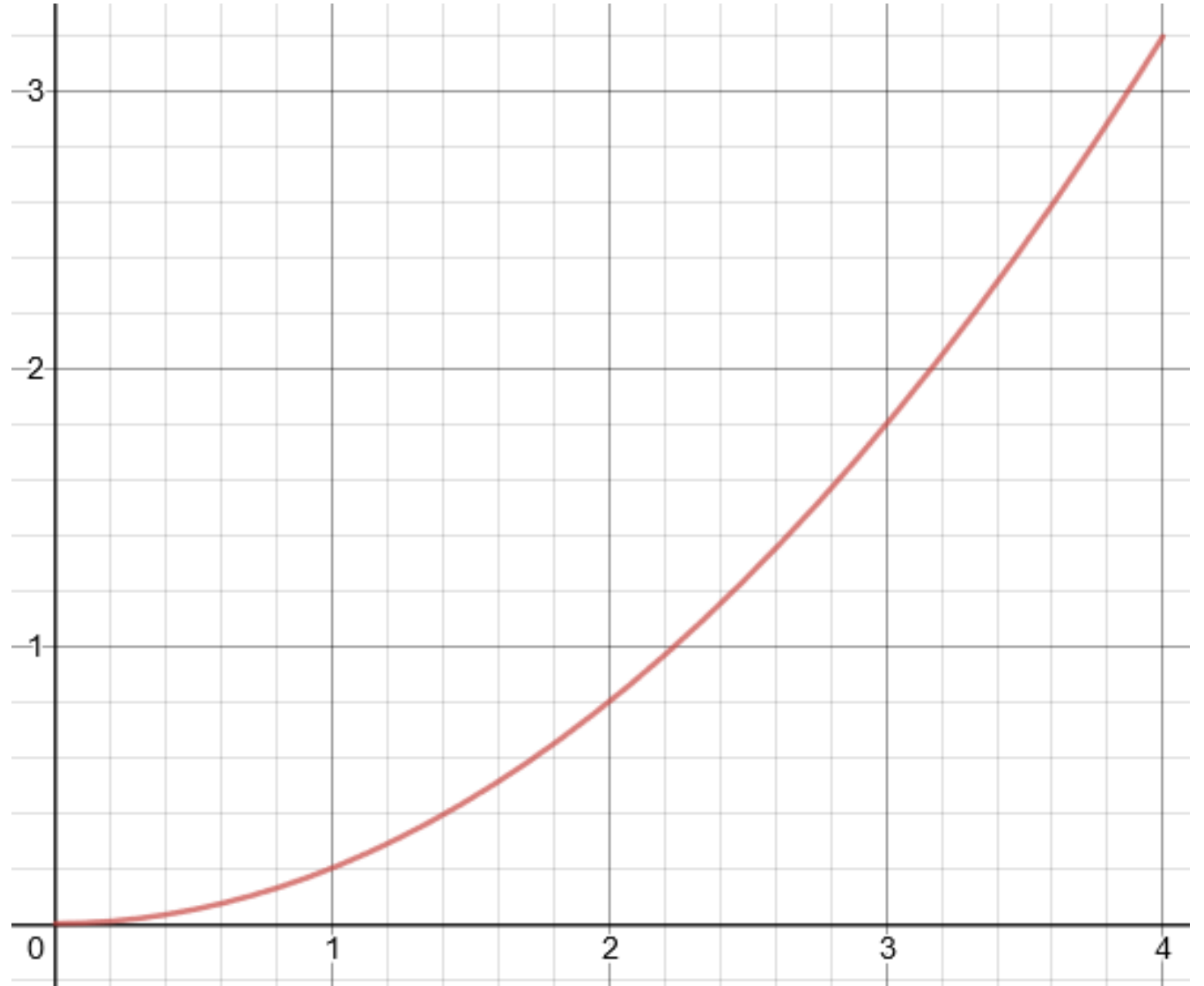
$$C(x) = \begin{cases} 0.2x, & \text{if } 0 \leq x < 1 \\ 0.6x - 0.4, & \text{if } 1 \leq x < 2 \\ x - 1.2, & \text{if } 2 \leq x < 3 \\ 1.4x - 2.4, & \text{if } 3 \leq x \leq 4 \end{cases}$$



# Concise Function

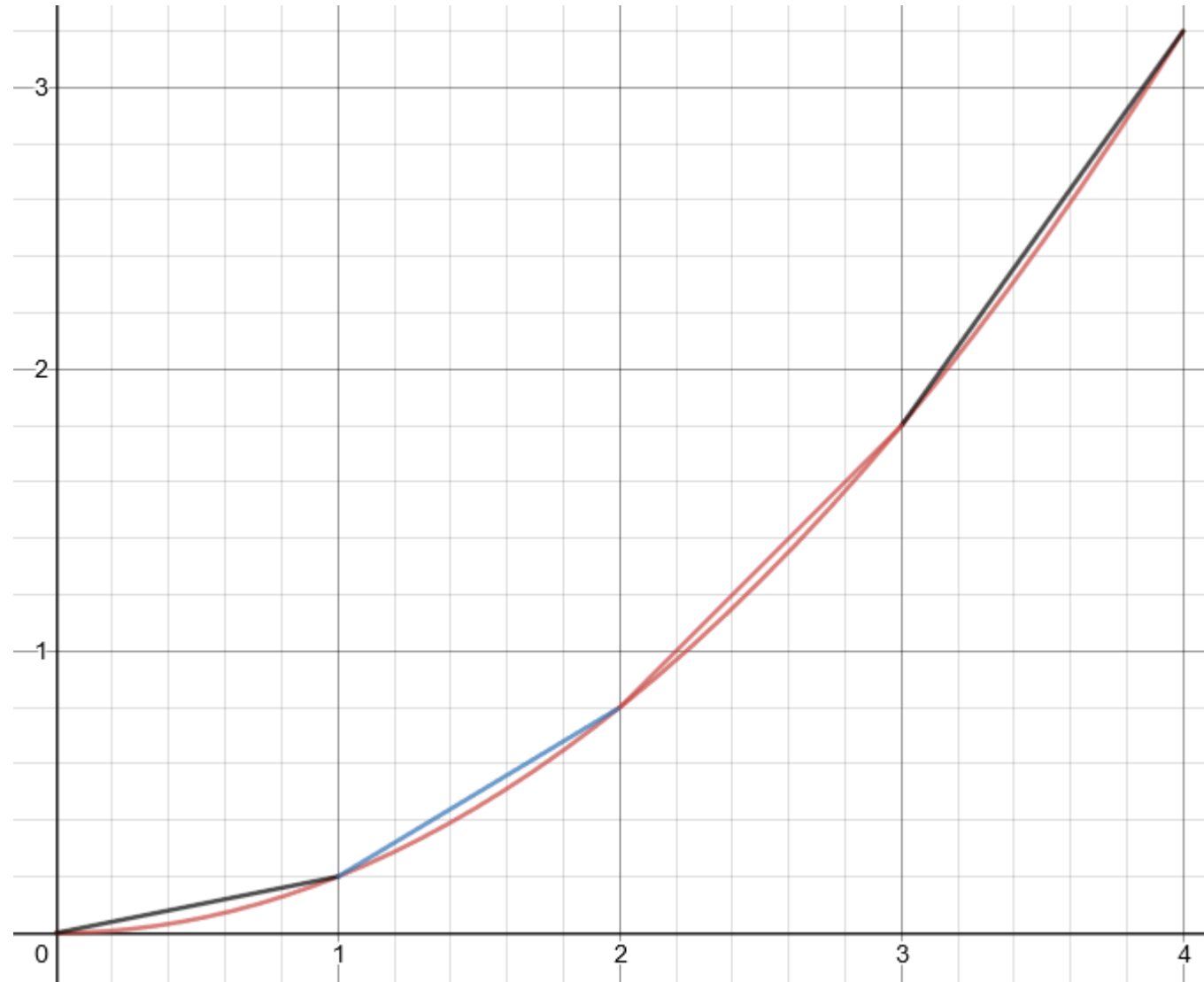
- Specified with  $O(1)$  pieces of information per arc

$$C(x) = \frac{1}{5}x^2 \quad \text{if } 0 \leq x \leq 4$$



# Approximation of Concise function with a Piecewise Linear function

- Assume feasible solutions for concise function are integers
- Equivalent to piecewise linear convex cost
- Optimal integer solution can be worse than continuous solution



# Scaling to improve accuracy

- Replace  $x_{ij}$  in original problem with  $\frac{y_{ij}}{M}$  for large  $M$
- Integer optimal solution  $y_{ij}^*$  gives solution to original problem is  $x_{ij}^* = \frac{y_{ij}^*}{M}$  within  $1/M$  of the optimal solution.
- Stretching the x-axis to include more integer breakpoints



# Issues with Approximation

- Algorithm running time on piecewise approximation may not be in terms of input data for concise cost graph
- Running time depends on number of segments used to approximate the concise function.

# Application: Urban Traffic Flow (1)

- As the number of cars increases, the road becomes more congested
  - delay in travel time
- Can be modeled as

$$Delay = \frac{\alpha x}{(u - x)}$$

- $x$  is the flow of traffic
  - $u$  theoretical road capacity
  - $\alpha$  a constant
- Finding flow that achieves a minimum overall delay for all roads is a convex cost network flow problem

# Application: Urban Traffic Flow (2)

- People travel from their starting point to their destination with minimum delay

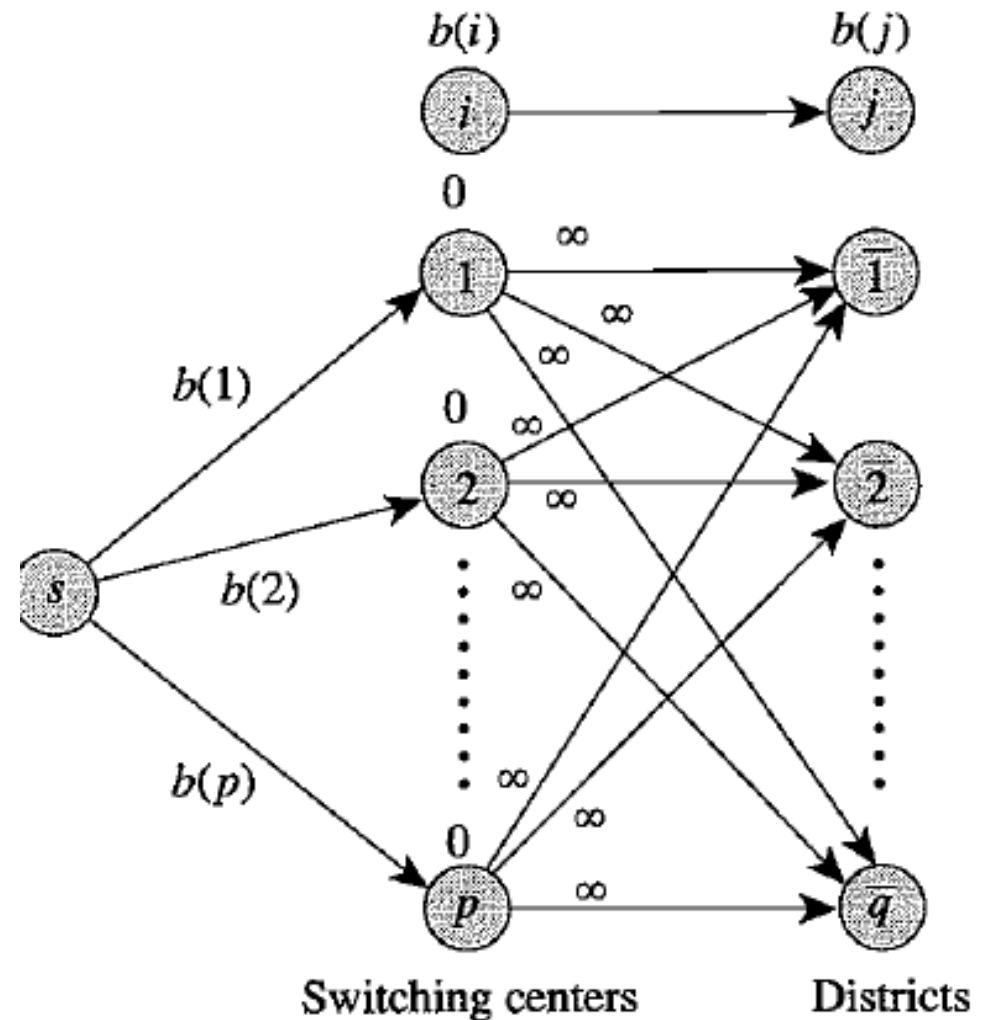
- The objective function is given by

$$\sum_{(i,j) \in A} \int_0^{x_{ij}} C_{ij}(y) dy$$

- $x_{ij}$  is the flow on arc  $(i,j)$
- $C_{ij}(y)$  is the delay cost on the arc
- The delay a person incurs depends on others
- If delay function is non decreasing then the objective function is convex

# Application: Area Transfers in Communication Networks

- In the past equipping every phone with a routing ability was expensive. Thus requiring switch centers to be used when routing phone calls.
- $d(j)$  current demand of district  $j$
- $b(i)$  represents the capacity of switch center  $i$

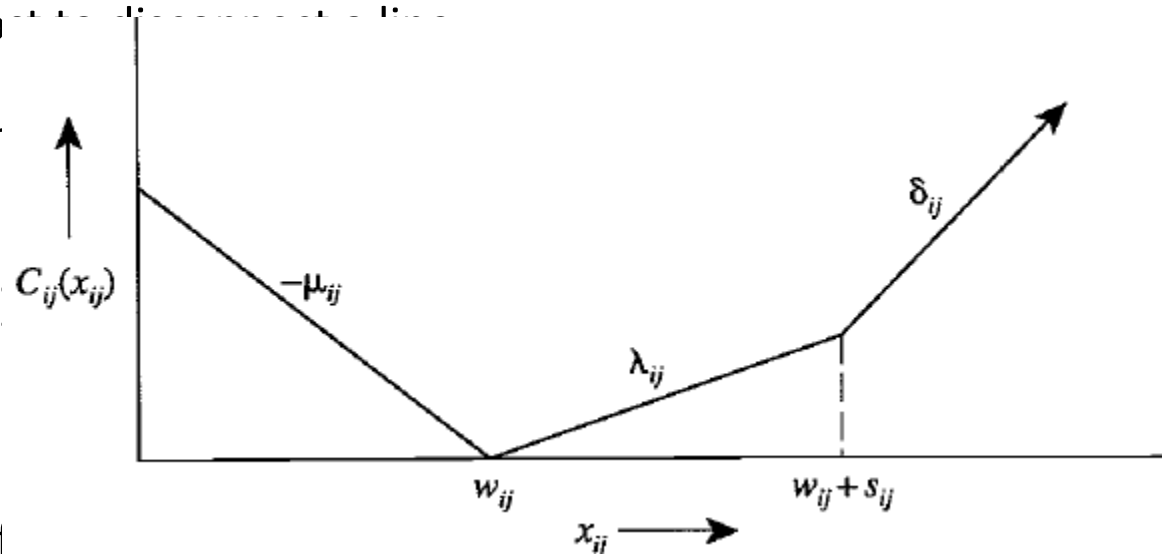


# Application: Area Transfers in Communication Networks

- $w_{ij}$  represents the number of lines currently in use between center  $i$  and district  $j$ .

- $\mu_{ij}$  represents the cost from switch center  $i$  to it to another switch center

- $s_{ij}$  represents the spare capacity of the switch center  $i$  to district  $j$  per line.



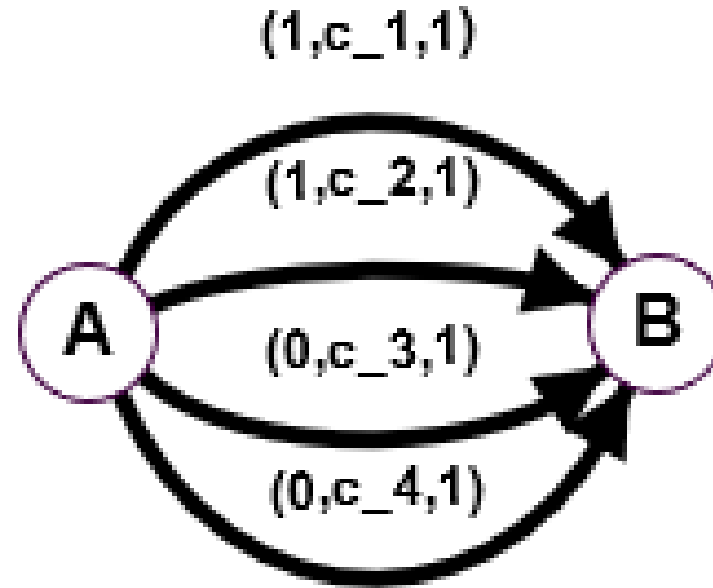
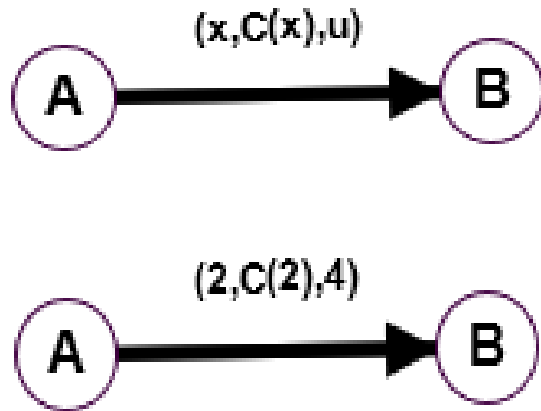
- $\delta_{ij}$  represents a cost beyond the spare ones and the assumption is  $\delta_{ij} > \lambda_{ij}$ .

# Network Transformation for Piecewise Convex Costs

- Transform convex cost flow problem into format where standard min cost flow algorithms work
- Idea: arc  $(i, j)$  with  $p$  pieces for its cost function becomes multiple arcs  $(i, j)_k$ , then we use the residual network to remove the multi-arc issue

# Network Transformation for Piecewise Convex Costs

- Arc with cost per unit flow of  $c_1$  when  $x_{ij} \in [0,1)$ ,  $c_2$  when  $x_{ij} \in [1,2)$ , and  $c_3, c_4$  are similar. Capacity is 4. We send 2 units of flow along this arc.



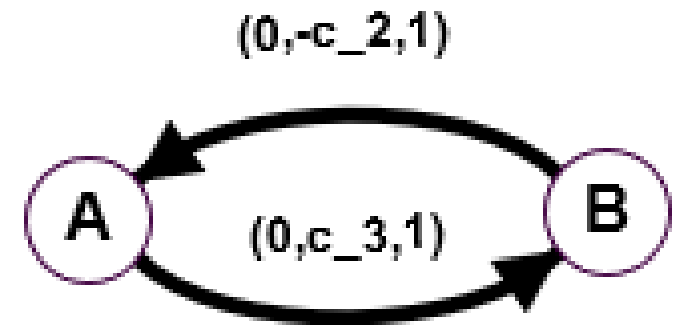
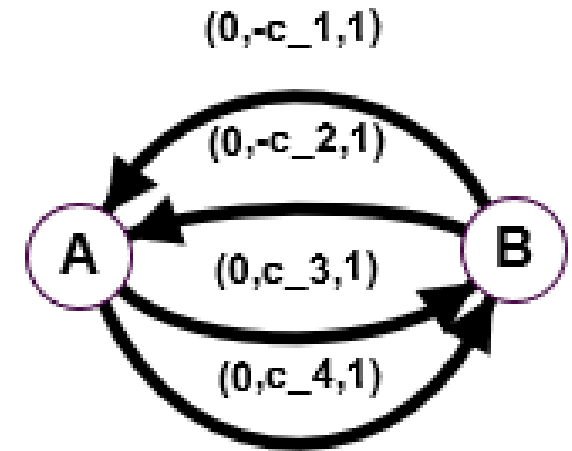
# Issues with Network Transformation

- Now we have a graph with multi-arcs!
- Same amount of information if cost was originally piecewise.
- Added information if cost was concise.



# Residual Network Modification

- Since convex,  $c_1 < c_2 < c_3 < c_4$ .
- If we sent a flow of 2, only need backward arc that saves that most and forward arc that costs the least per unit flow.
- Now usual min cost algorithms work



# General Residual Network Transformation

- Cost per unit flow may not change for each unit of flow. Let  $c_1 = 1$  for  $x_{ij} \in [0,2)$ ,  $c_2 = 4$  when  $x_{ij} \in [2,6)$ , and  $c_3 = 10$  when  $x_{ij} \in [6,13]$ .
- Send 4 units of flow, cost per unit is  $c_2$ . 2 capacity left until cost per unit changes either way. Now send another 3 units of flow so  $x_{ij} = 7$ .



# Pseudopolynomial Algorithms

- Cyclecanceling Algorithm: Similar to before but with modified residual network. When augmenting, may not be able to augment full amount along a negative cycle in the residual network. ( $-c_2$  only present in residual network)
- To fix this, send flow along the same negative cycle until its cost becomes nonnegative. This avoids unnecessarily searching for negative cost cycles.

# Pseudopolynomial Algorithms

- Successive shortest path algorithm: Again, use modified residual network.
- However, we may augment along the same path multiple times as the cost per unit flow of an arc changes.
- Might send only 1 unit of flow per augmentation with concise function approximation

# Polynomial Time Algorithm

- Capacity Scaling Algorithm: Modification of successive shortest paths. Instead of using integer breaks of 1 unit from the start, we start with larger intervals and then scale downward.
- Let  $C_{ij}(x_{ij}) = x_{ij}^4$  with  $u_{ij} = 12$ .
- Model function with breakpoints 8 apart, then 4 apart. Do this until we have breakpoints that are 1 apart.
- First phase can augment 8 units of flow, then 4, etc. Faster decrease of excess.

# Algorithm

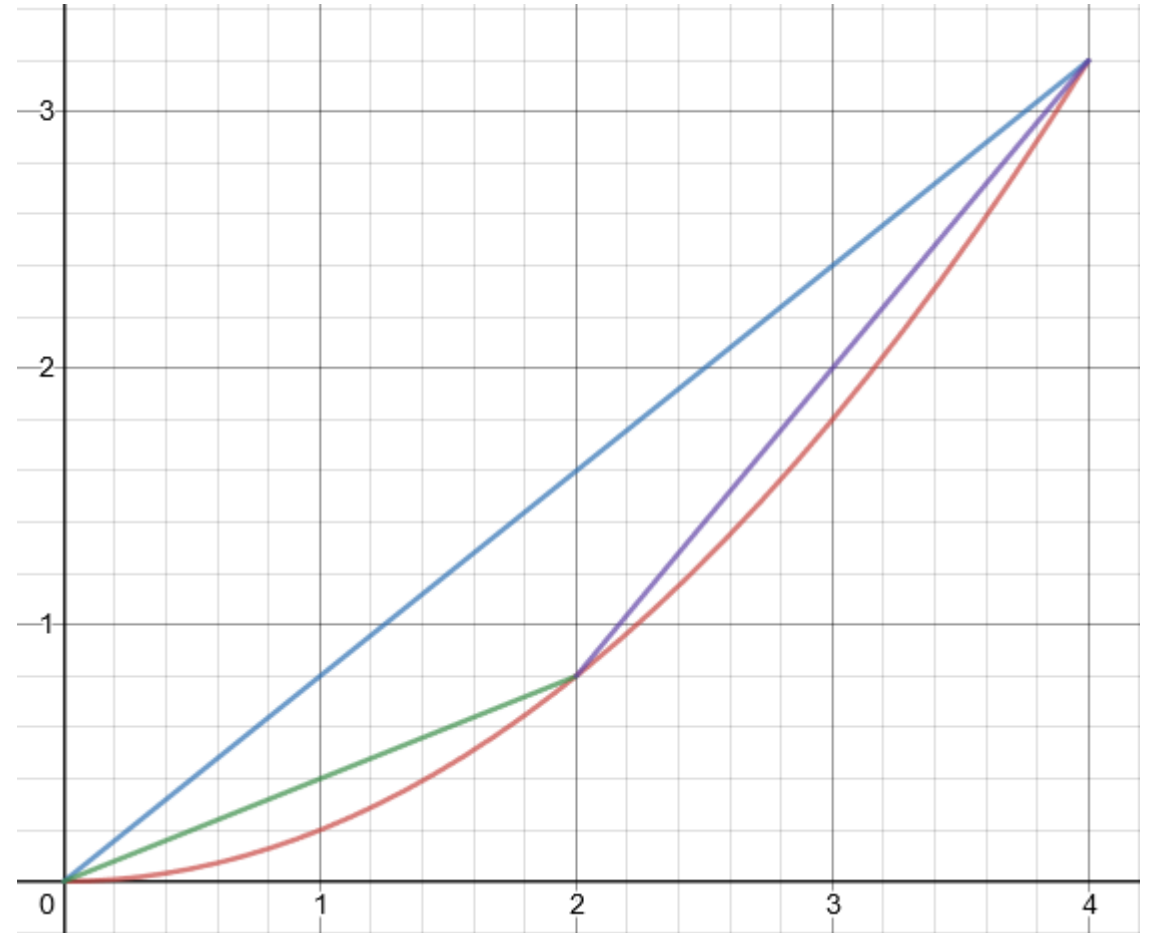
- Initially: 0 pseudoflow and 0 node potentials
- Let  $\Delta$  be the interval between breakpoints. We have a phase for each  $\Delta$ . During each phase, we create a  $\Delta$  residual network, constructed as we did before. (Different approximations require different modified residual networks)
- If arc  $(i, j)$  or  $(j, i)$  violates the reduced cost optimality conditions, increase or decrease flow  $x_{ij}$  by  $\Delta$  to correct this.

# Algorithm Cont.

- Define sets  $E(\Delta)$  and  $D(\Delta)$  for nodes with excesses and deficits of at least  $\Delta$ .
- We reduce excess by  $\Delta$ , then  $\frac{\Delta}{2}$ , then  $\frac{\Delta}{4}$  until we get to breakpoints that are 1 apart.
- Essentially do successive shortest path algorithm until either  $E(\Delta)$  or  $D(\Delta)$  is empty, then reduce  $\Delta$  by half and do it again.

# Capacity Scaling Algorithm - Optimality

- Flow  $x_{ij}$  satisfies reduced cost optimality within each phase
- Only needs to be adjusted by at most  $\Delta$  at start of next phase to achieve optimality
- $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j)$
- $c_{ij}$  is slope of segment to the right of  $x_{ij}$





# Runtime

- $S(n, m, C)$  time to solve shortest path problem with nonnegative arc lengths
- At end of  $2\Delta$  phase,  $E(2\Delta) = \emptyset$  or  $D(2\Delta) = \emptyset$
- Less than  $n * 2\Delta$  positive imbalance.

# Runtime

- Start of phase, potentially adjust each arc flow by  $\Delta$ .
- $(m + n) * 2\Delta$  is a bound on positive imbalance. Augment  $\Delta$  each iteration within a scaling phase.
- $O(m)$  augmentations within a phase and a  $\log(U)$  bound on phases. Also need to find shortest path for each augmentation.
- $O(m * \log(U) S(n, m, C))$

# References

- Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. 1993. Network flows: theory, algorithms, and applications. Prentice-Hall, Inc., USA.