Network Flows

UPCOPENCOURSEWARE number 34414

Topic 7: Convex Costs Flows

F.-Javier Heredia

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Estadística
i Investigació Operativa

7.- Convex Cost Flows

- Source material:
  - R.K. Ahuja, Th.L. Magnanti, J. Orlin “Network Flows”, chap. 15.
  - R.K. Ahuja “Advanced Network Optimization” http://www.ise.ufl.edu/ANO/

Introduction to Convex Cost Flows

- Convex cost flow problems are minimum cost flow problems where the cost of flow is nonlinear.

$$\text{Minimize} \quad \sum_{(i,j)\in A} C_{ij}(x_{ij})$$

subject to

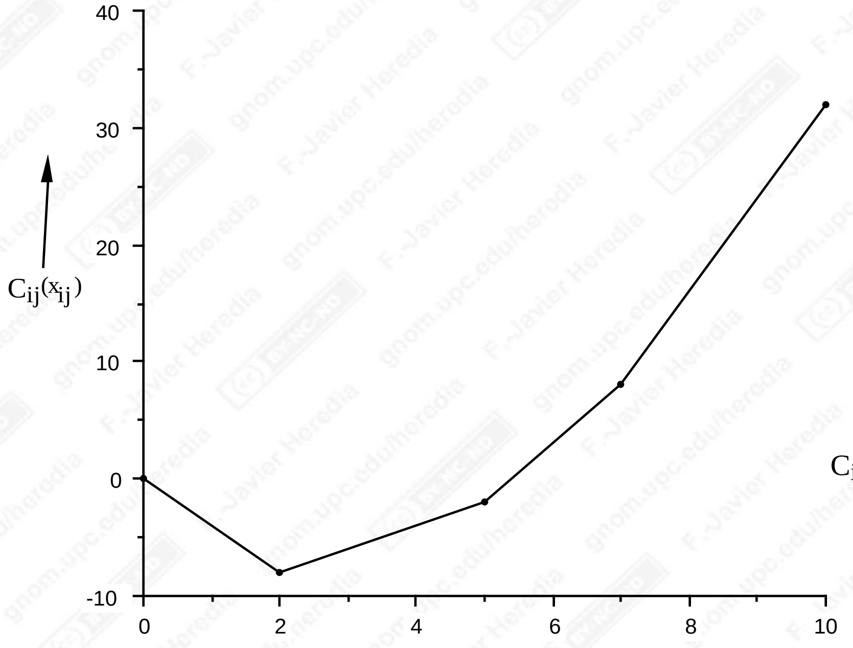
$$\sum_{\{j:(i,j)\in A\}} \mathbf{x}_{ij} - \sum_{\{j:(j,i)\in A\}} \mathbf{x}_{ji} = b(i) \quad \text{for each node } i \in N$$

$$0 \le x_{ij} \le u_{ij} \quad \text{and integer for each arc } (i,j) \in N$$

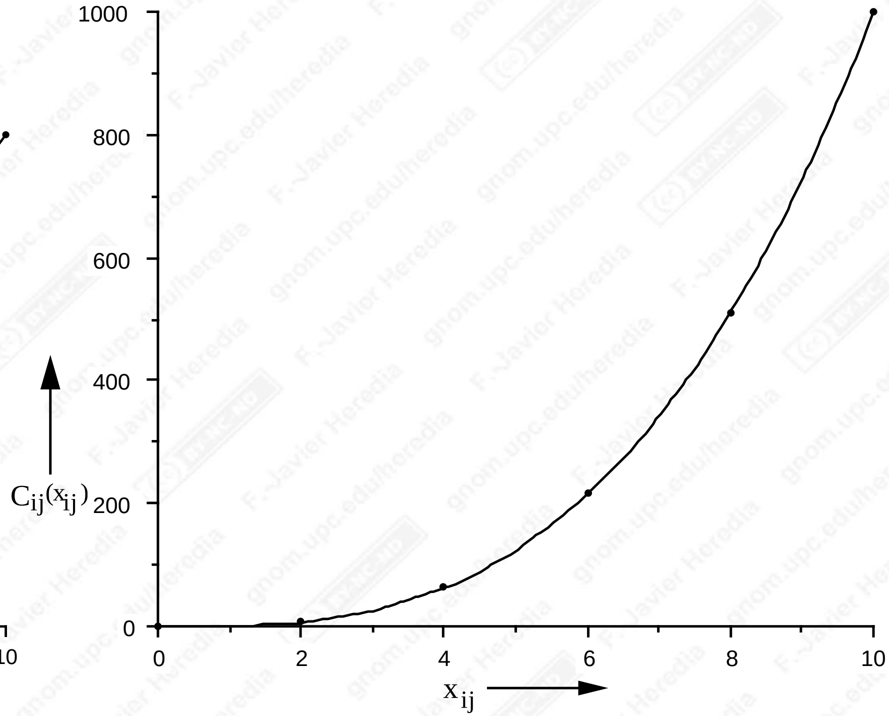
- Here we assume that **the total cost is separable.**
- The cost of flow $C_{ij}(x_{ij})$ is a **convex function** of x_{ij} instead of a linear function $c_{ij}x_{ij}$.

Introduction to Convex Cost Flows

- Some sample cost functions:



(a)



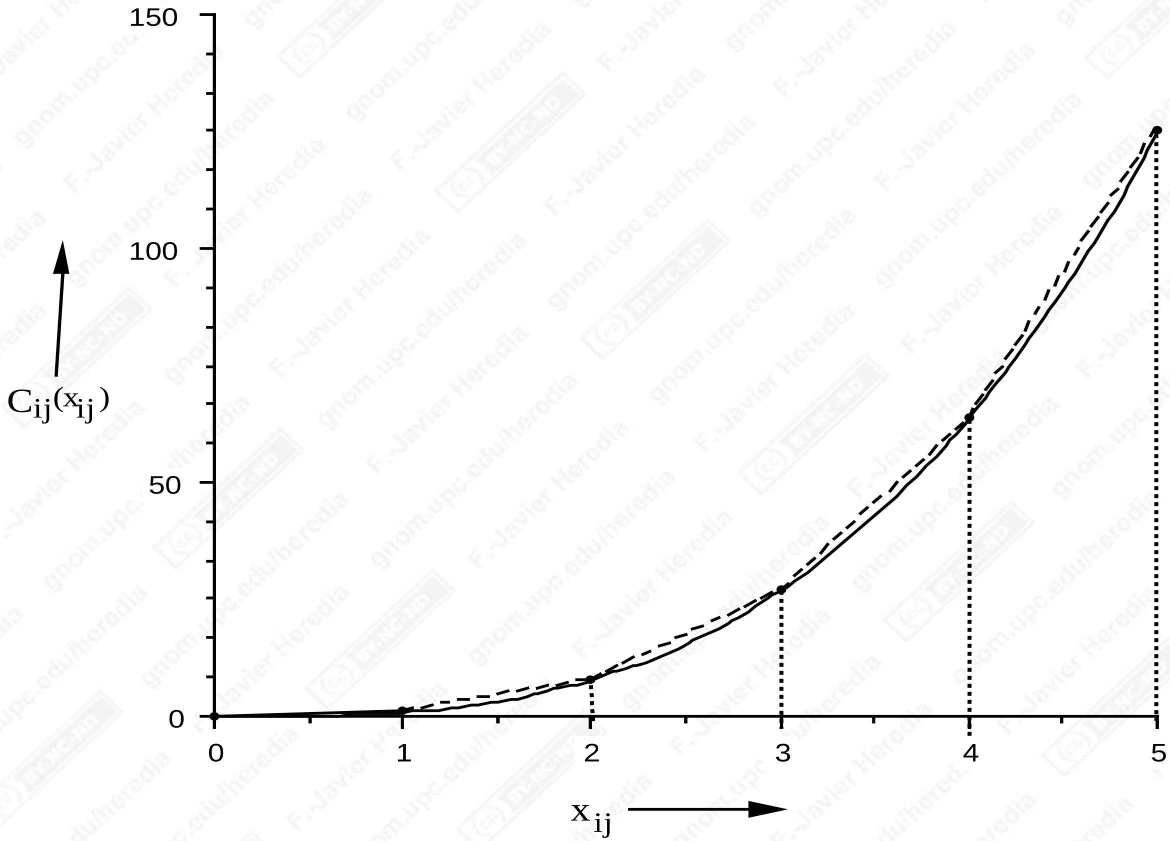
(b)

Applications

- Urban Traffic Flows.
- Area transfer in Communication Networks.
- Matrix Balancing.
- Stick Percolation Problem.

Piecewise Linear Assumption

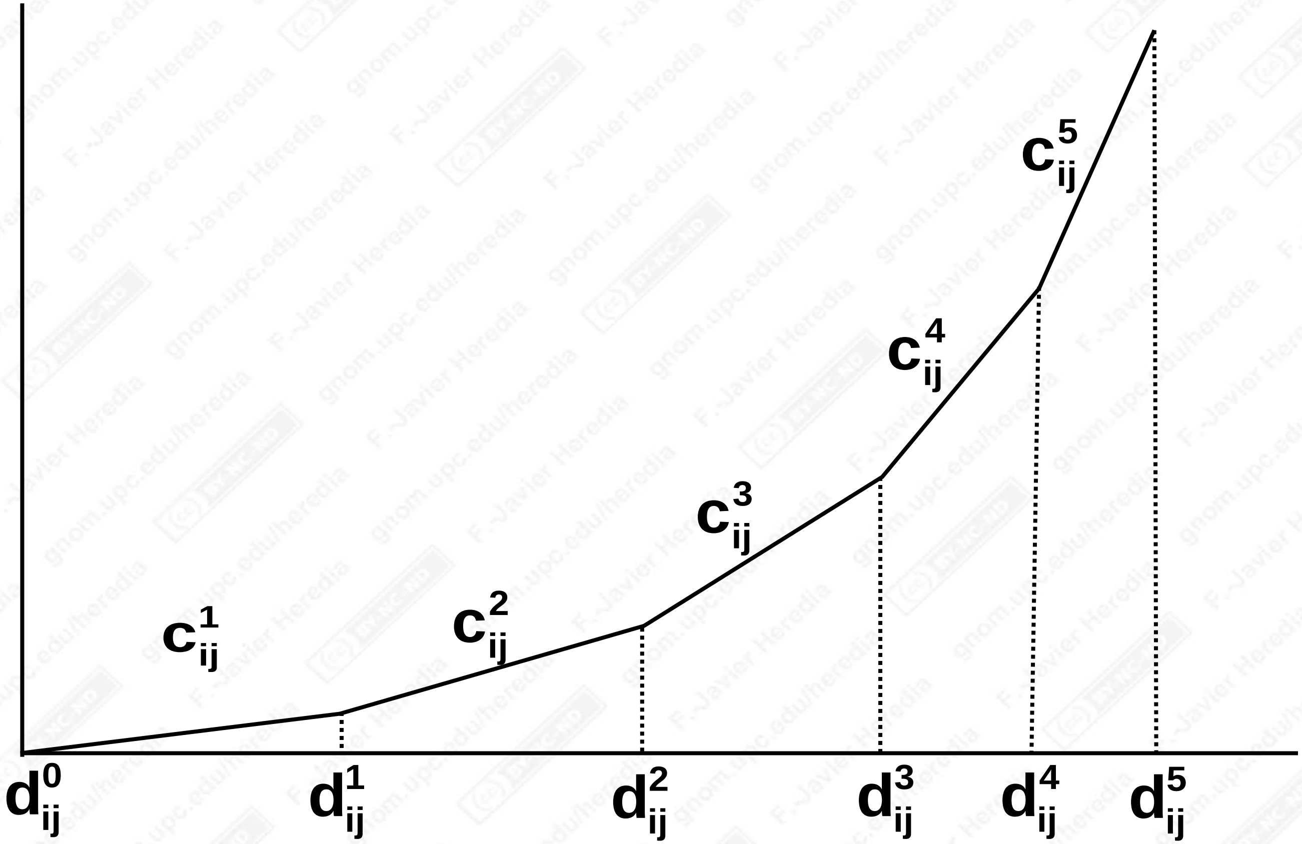
- We will assume that the cost of flow on any arc is a piecewise linear convex function:



UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH
Departament d'Estadística i Investigació Operativa

Nature of the Cost Function

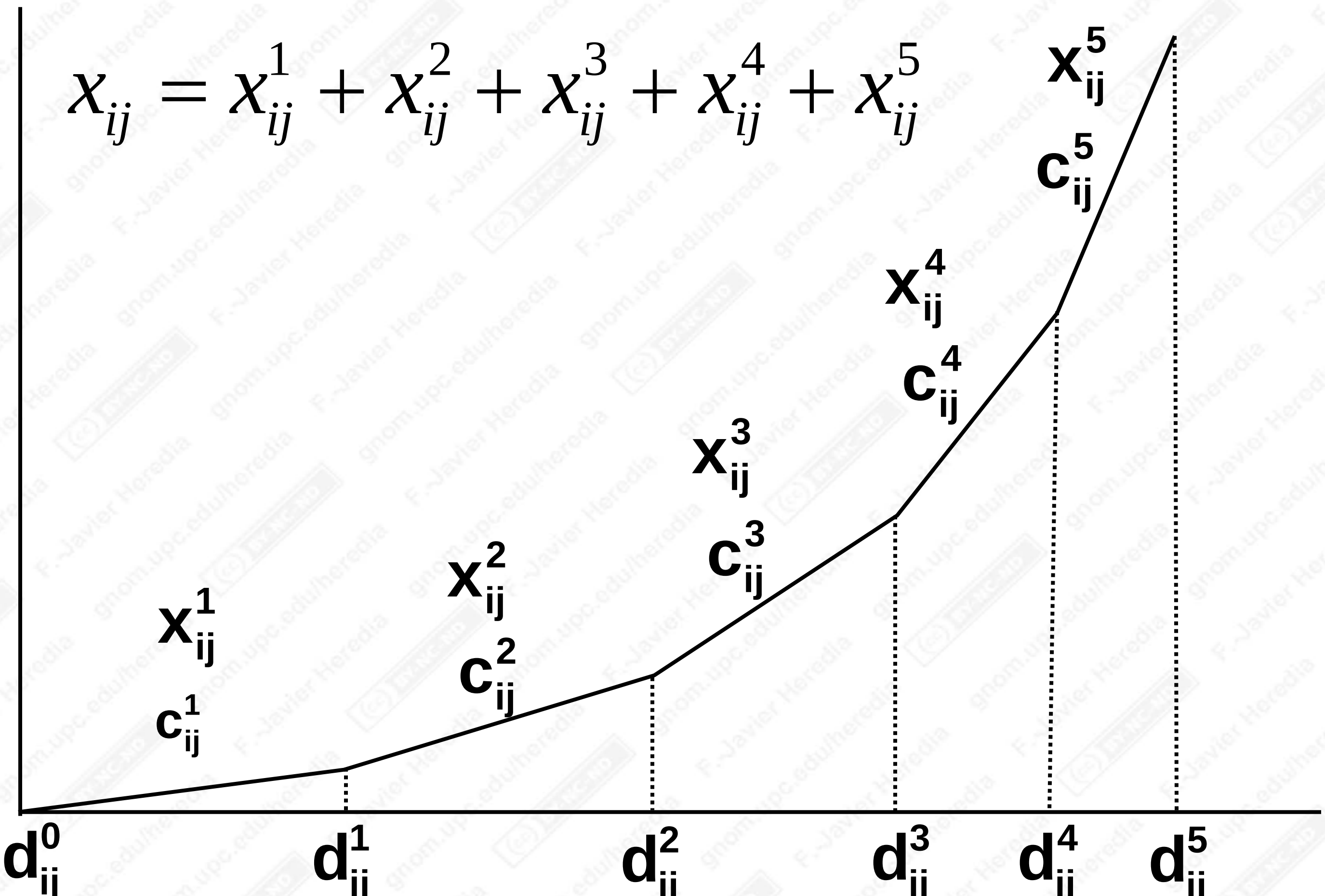
- We will assume that each arc has p linear segments associated with it. For example, if p = 5, then the cost function would look like:



Transformation to Min Cost Flows

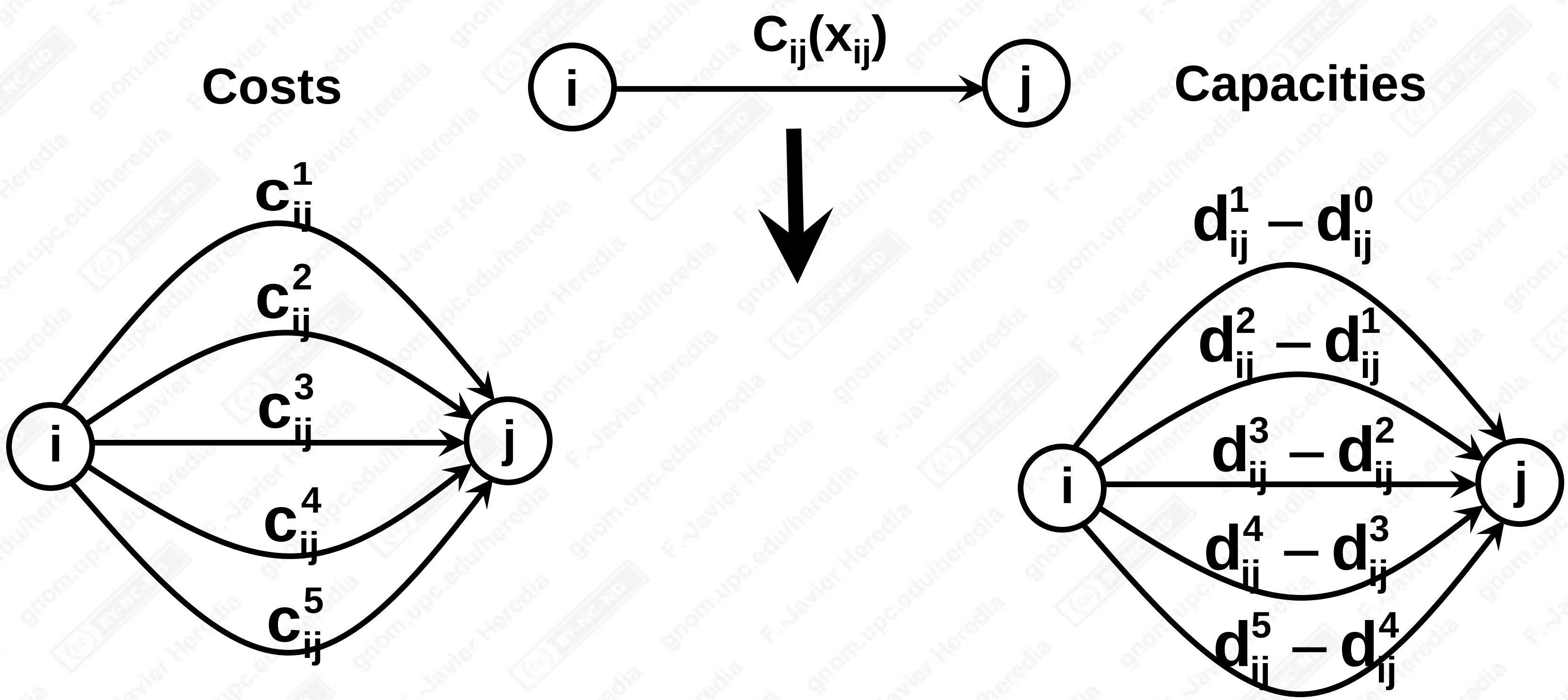
- The convex cost flow problem can be transformed to a minimum cost flow problem by increasing the number of variables:

$$x_{ij} = x_{ij}^1 + x_{ij}^2 + x_{ij}^3 + x_{ij}^4 + x_{ij}^5$$



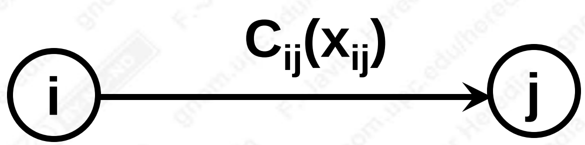
Transformation to Min Cost Flows (contd.)

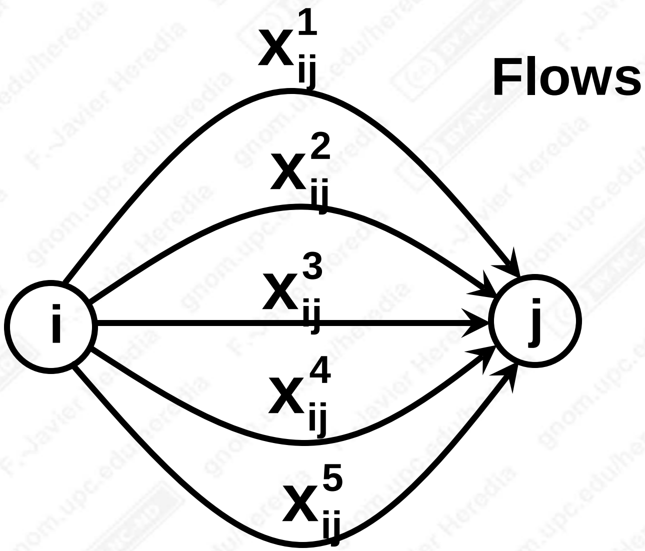
- We replace each arc with p linear segments by p arcs with linear costs.



Transformation to Min Cost Flows (contd.)

- We call a flow in the transformed network to be a **contiguous flow** if a segment with greater cost is used only if segments of lower costs are saturated.





- There is a one-to-one correspondence between flows in the original network and contiguous flows in the transformed network.

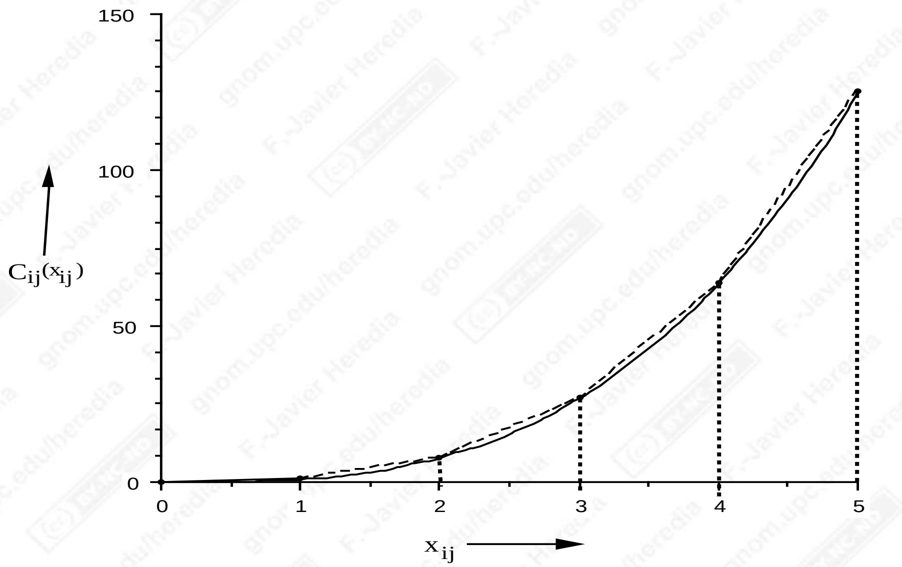
Transformation to Min Cost Flows (contd.)

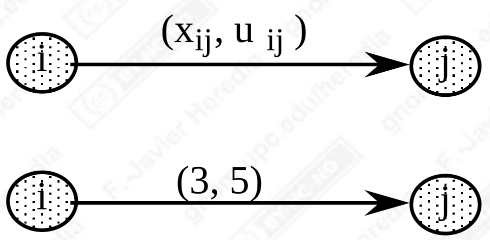
- A feasible flow in the transformed network may or may not be contiguous.
- The optimal solution in the transformed network is always a contiguous flow.
- We can solve the convex cost flow problem by solving a minimum cost flow problem.

Pseudopolynomial-Time algorithms

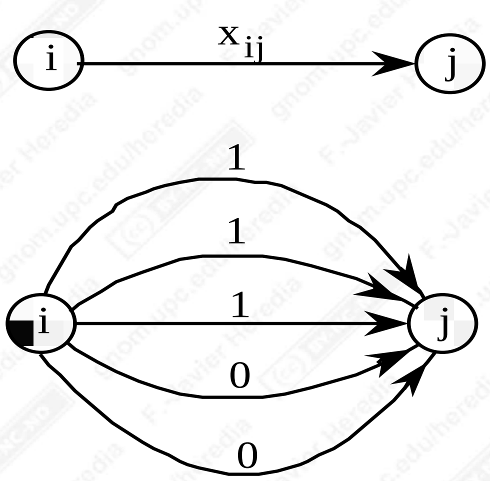
- We can transform a convex cost flow problem into a minimum cost flow problem and then solve it using any minimum cost flow algorithm.
  - The cycle canceling algorithm
  - The successive shortest path algorithm
- However, the resulting minimum cost flow problem may have too many arcs which might slow down the algorithm.
- We will show that if we define the residual network appropriately, then the network size will not increase at all (that is, extra arcs can be handled implicitly).

Constructing the Residual Network

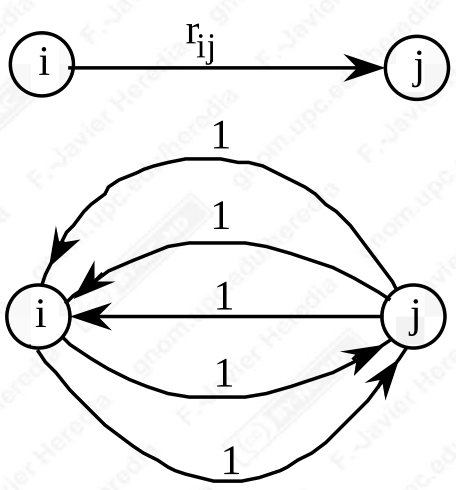




(a)



(b)

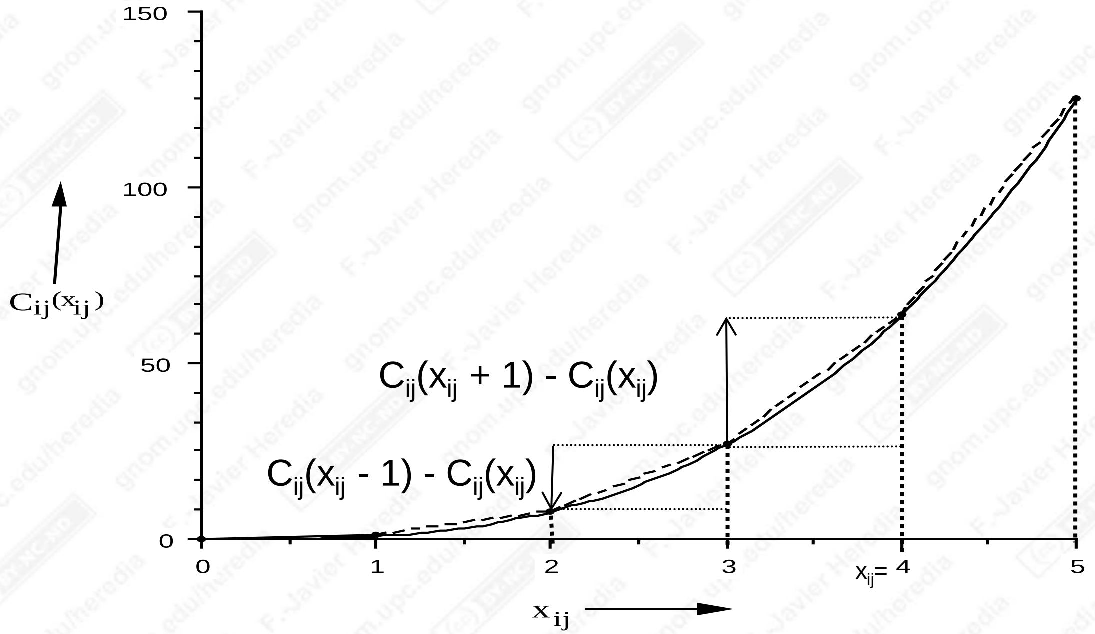


(c)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Estadística
i Investigació Operativa

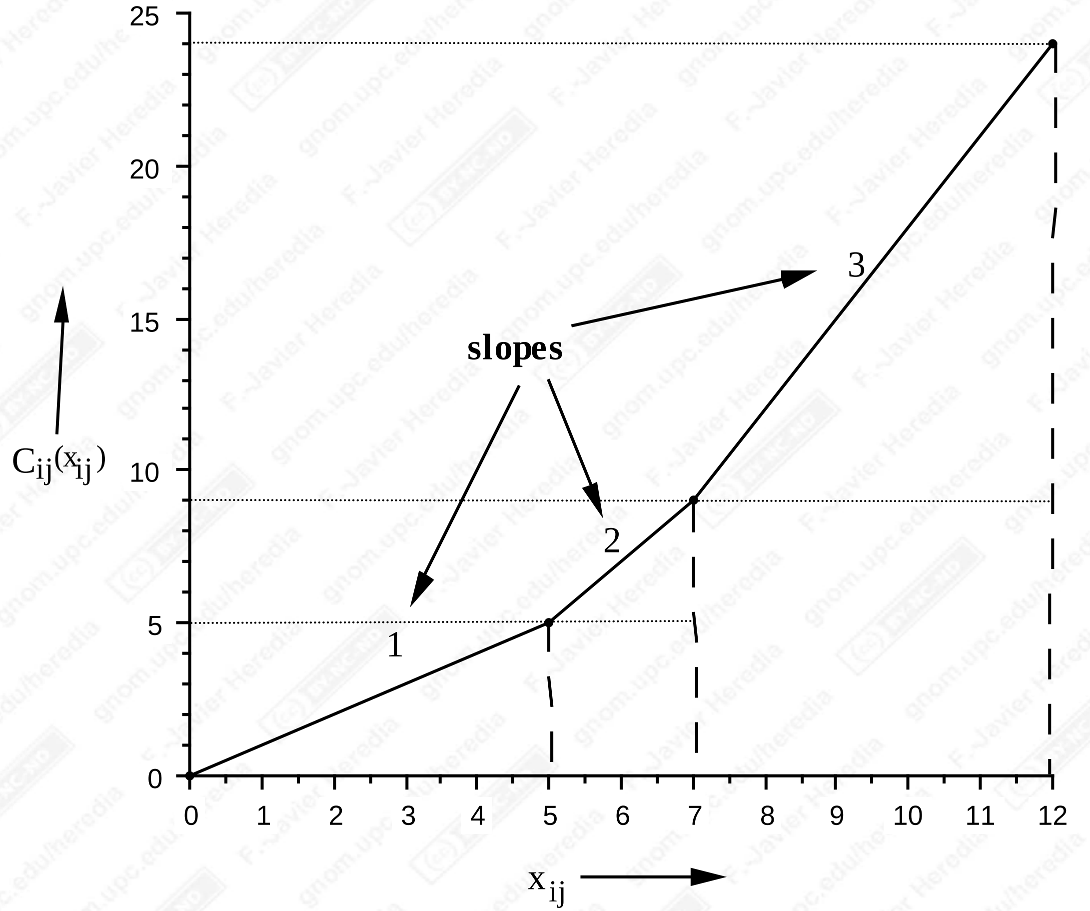
Constructing the Residual Network (contd.)

- We need to maintain only two arcs for any arc (i, j): one from node i to node j and another from node j to node i.
- For the exemple:
  - Cost of the arc (i, j): $C_{ij}(x_{ij} + 1) - C_{ij}(x_{ij})$
  - Cost of the arc (j, i): $C_{ij}(x_{ij} - 1) - C_{ij}(x_{ij})$



Constructing the Residual Network (contd.)

- What will be the costs and residual capacity of arcs in the residual networks if x_{ij} = 5, 6, 10.

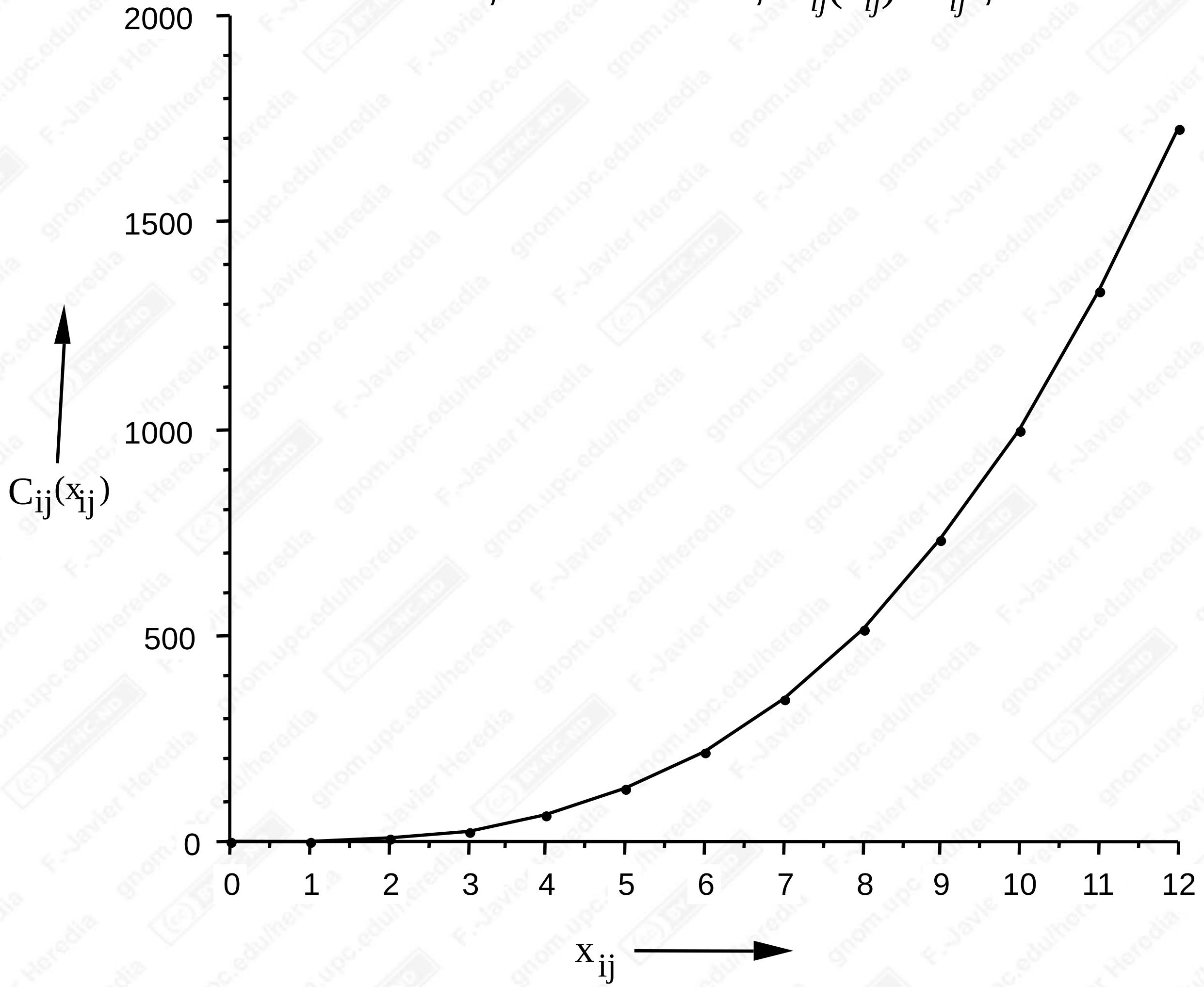


Polynomial Time Capacity Scaling Algorithm

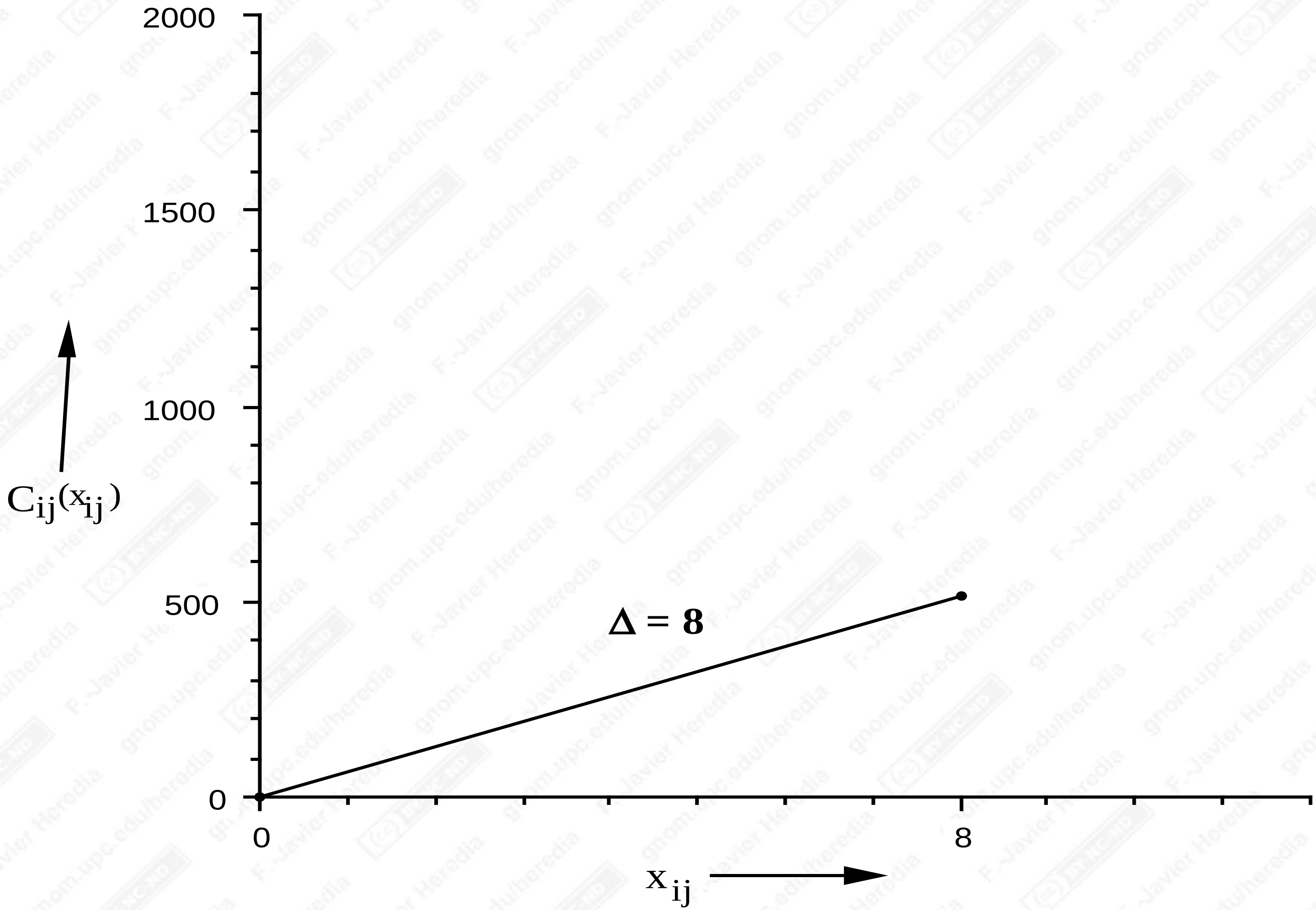
- The capacity scaling algorithm approximates the convex cost function with greater refinements (in terms of a parameter Δ).
- This algorithm is a modification of the capacity scaling algorithm for the minimum cost flow problem and has the same running time.

The Original Cost Function

Consider, for instance, $C_{ij}(x_{ij})=x_{ij}^4$, linearized with step length



The Approximate Cost Function (Δ = 8)

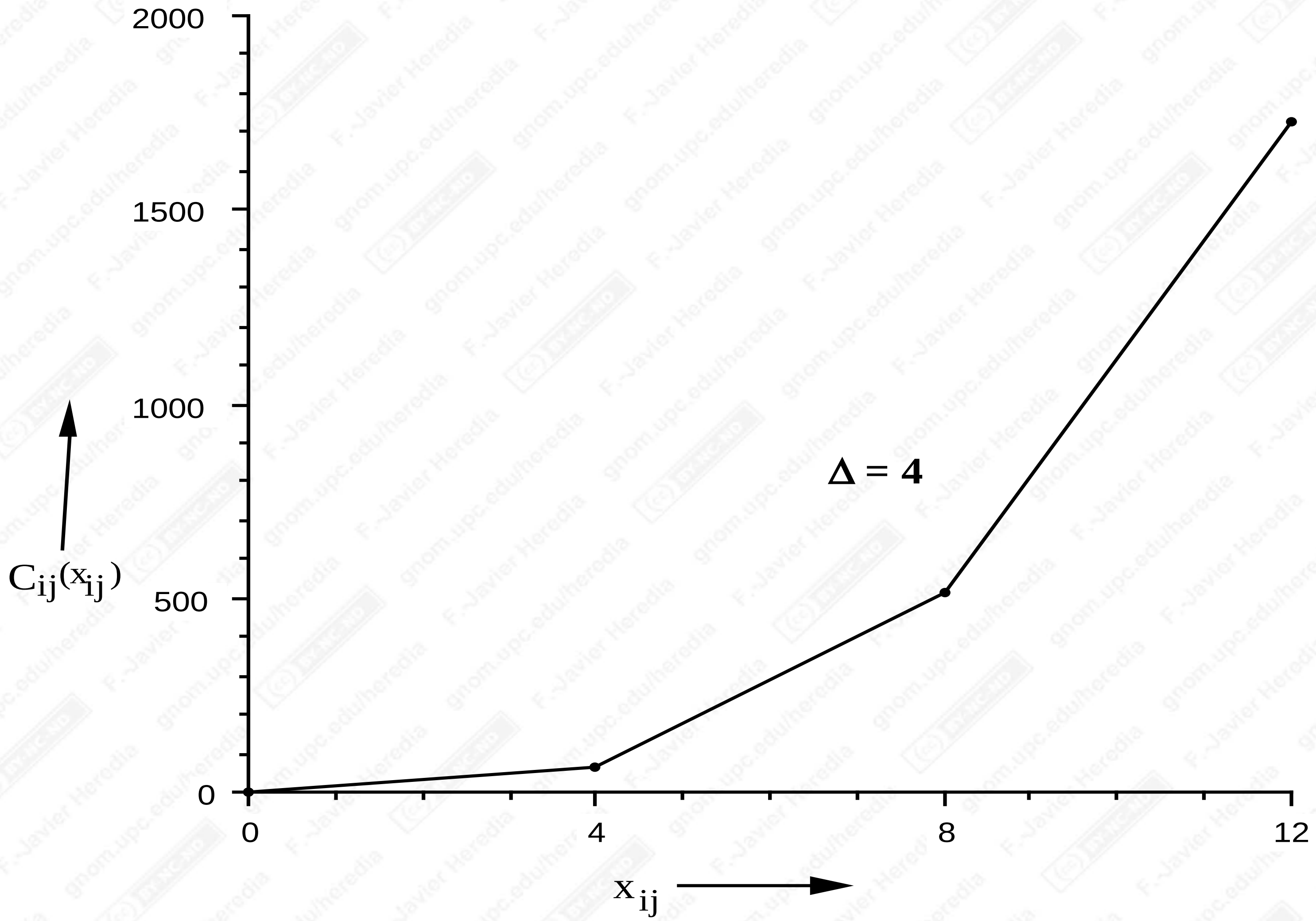


(a)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Estadística
i Investigació Operativa

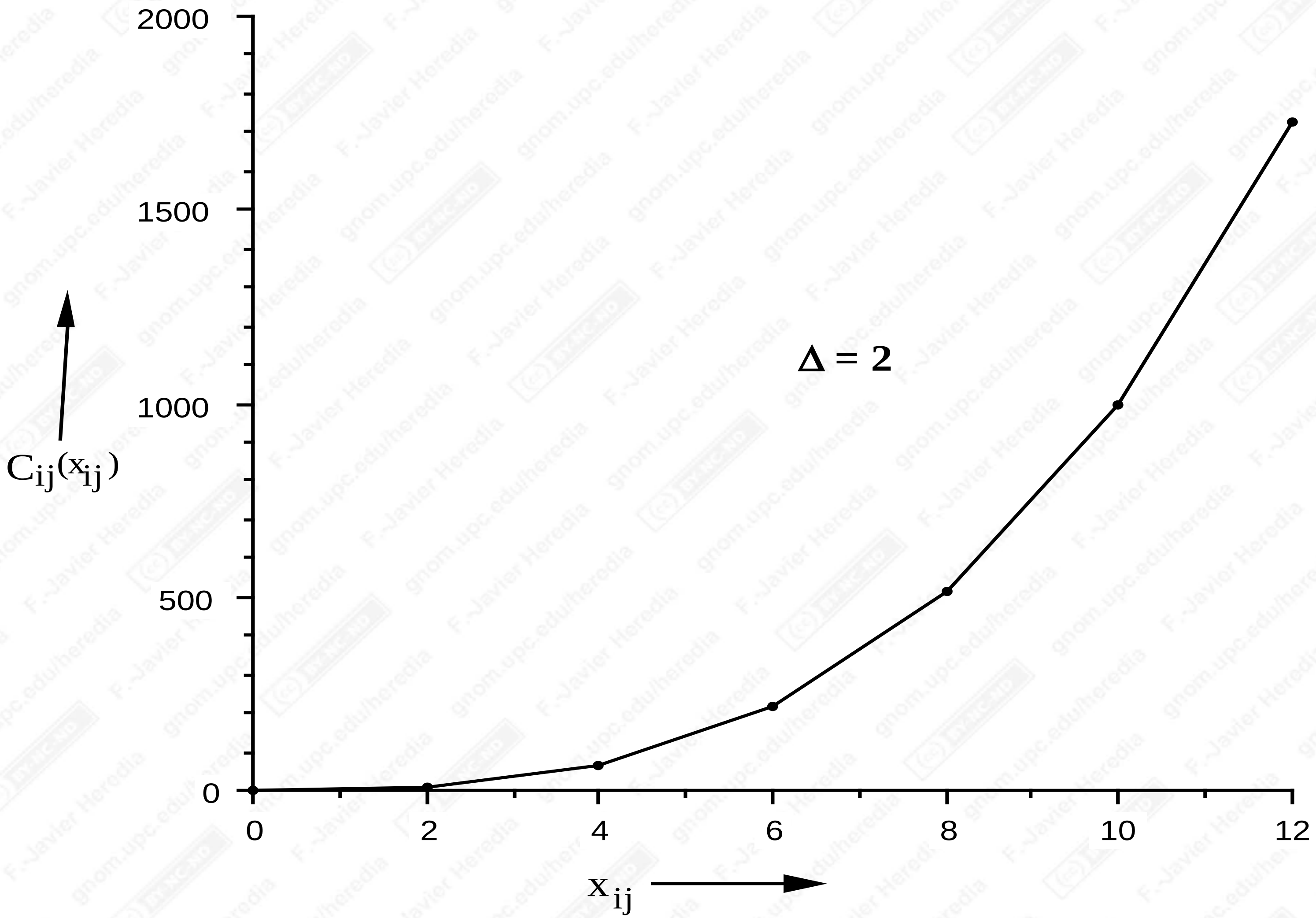
F.-Javier Heredia http://gnom.upc.edu/heredia

The Approximate Cost Function (Δ = 4)



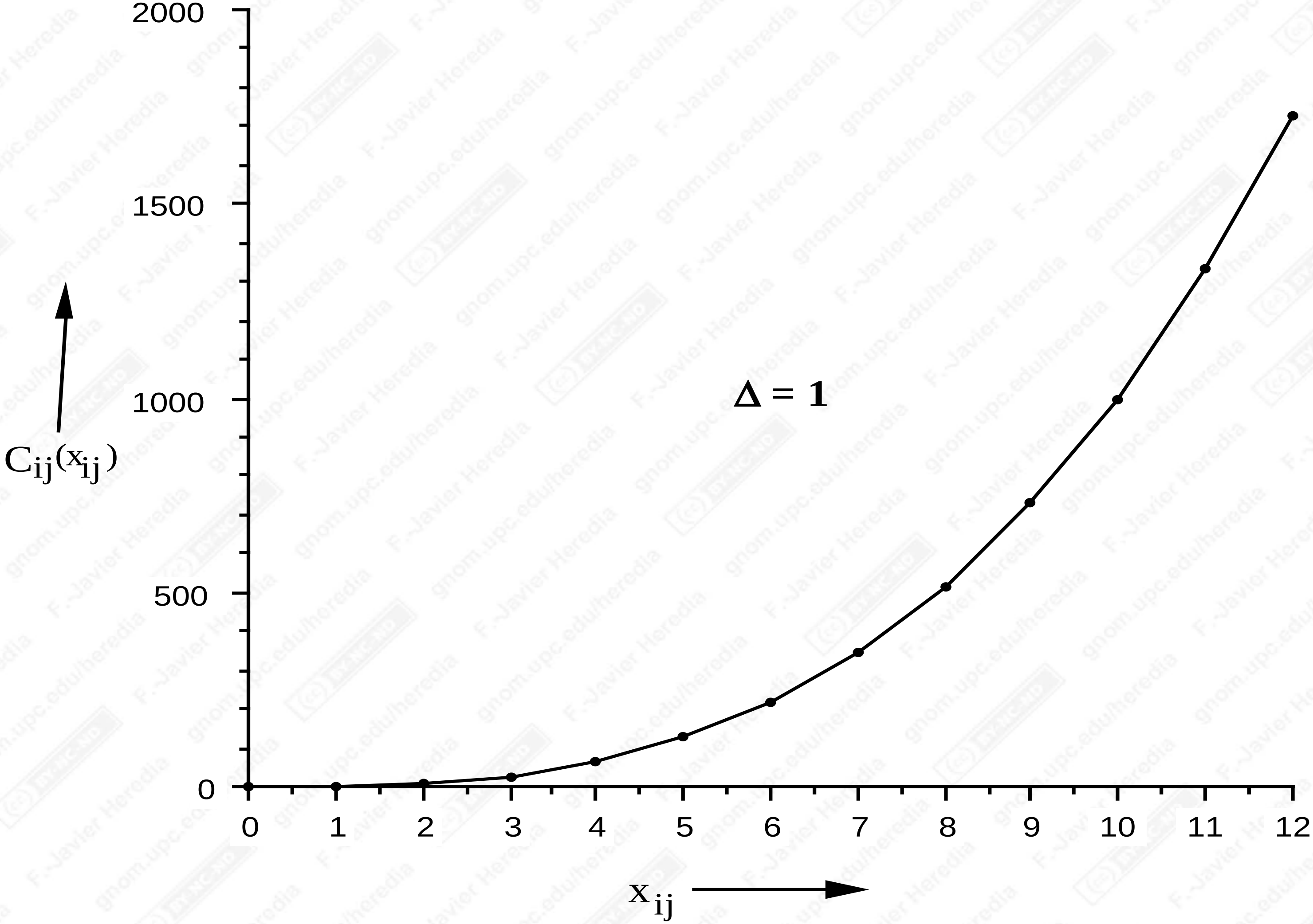
(b)

The Approximate Cost Function (Δ = 2)



(c)

The Approximate Cost Function (Δ = 1)



(d)

The Δ-Residual Network

- For any arc $(i, j) \in A$ with $x_{ij} + \Delta \leq u_{ij}$, the Δ-residual network contains the arc (i,j) with residual capacity Δ and cost equal to

$$(C_{ij}(x_{ij} + \Delta) - C_{ij}(x_{ij}))/\Delta$$

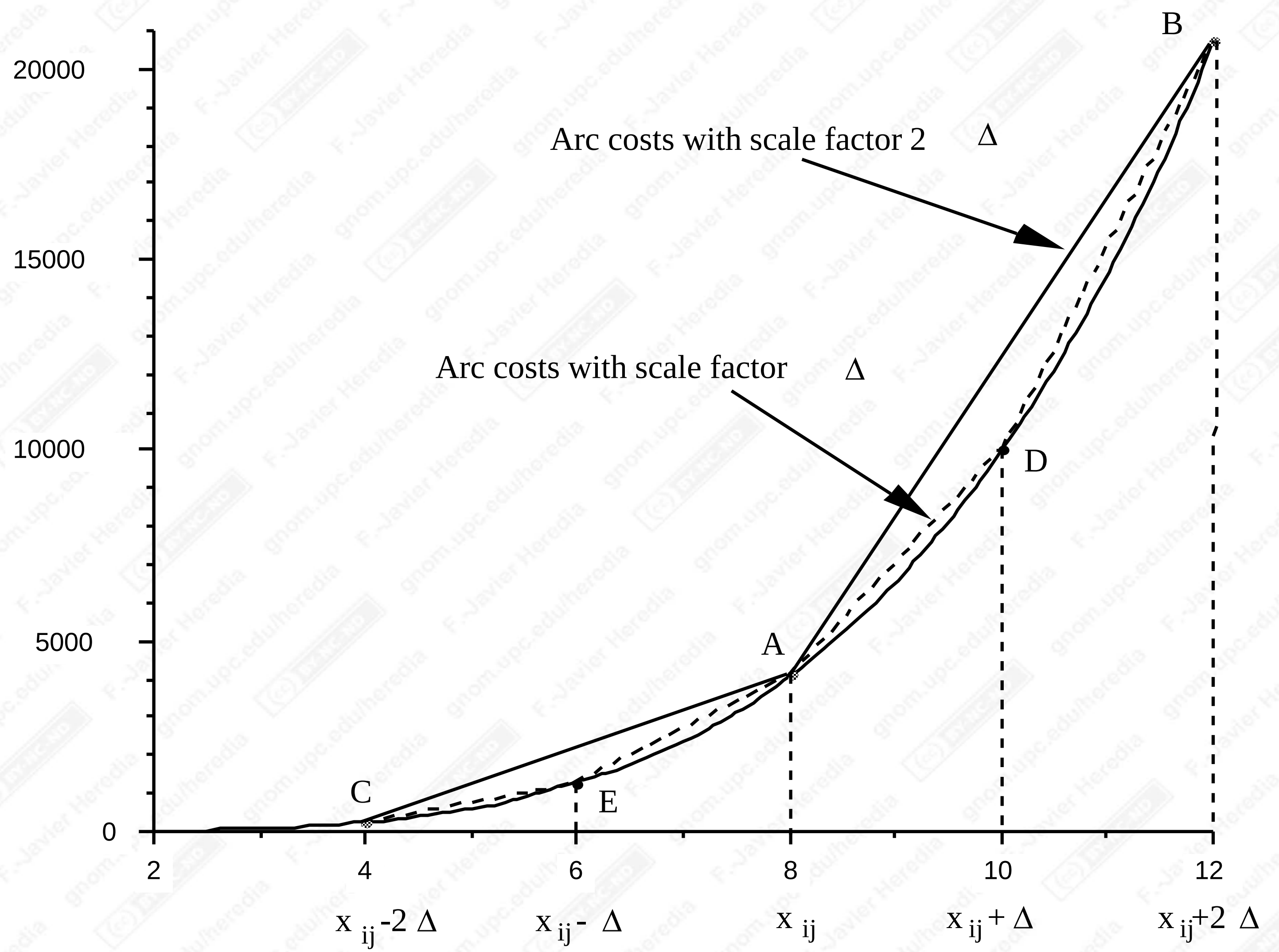
- For any arc $(i, j) \in A$ with $x_{ij} \geq \Delta$, the Δ-residual network contains the arc (j,i) with residual capacity Δ and cost equal to

$$(C_{ij}(x_{ij} - \Delta) - C_{ij}(x_{ij}))/\Delta$$

The Capacity Scaling Algorithm

algorithm capacity scaling;
begin
    $x := 0$, $\pi := 0$;
    $\Delta := 2^{\lfloor \log U \rfloor}$;
    while $\Delta \geq 1$
    begin {Δ-scaling phase}
        preprocessing;
        $S(\Delta) := \{i \in N : e(i) \geq \Delta\}$;
        $T(\Delta) := \{i \in N : e(i) \leq -\Delta\}$;
        while $S(\Delta) \neq \Phi$ and $T(\Delta) \neq \Phi$ *do*
        begin
            select a node $k \in S(\Delta)$ and a node $l \in T(\Delta)$;
            determine shortest path distances d(.) from node k to all other nodes in the Δ-residual network $G(x, \Delta)$ with respect to the reduced costs;
            let P denote a shortest path from node k to node *l* in $G(x, \Delta)$;
            update $\pi := \pi - d$;
            augment Δ units of flow along the path P;
            update x, $S(\Delta)$, $T(\Delta)$ and $G(x, \Delta)$;
        end;
        $\Delta := \Delta/2$;
    end;
end;

Going from One Scaling Phase to Another



Preprocessing

- As we go from 2Δ-scaling phase to Δ-scaling phase, the costs of the arc (i, j) and (j, i) change and so their reduced costs.
- There are four possibilities to consider after the reduced costs are changed:

(1)	$c^{\pi}_{ij} \geq 0$ **and** $c^{\pi}_{ji} \geq 0$	Nothing needs to be done.
(2)	$c^{\pi}_{ij} \geq 0$ **and** $c^{\pi}_{ji} < 0$	Decrease x_{ij} by Δ units
(3)	$c^{\pi}_{ij} < 0$ **and** $c^{\pi}_{ji} \geq 0$	Increase x_{ij} by Δ units.
(4)	$c^{\pi}_{ij} < 0$ **and** $c^{\pi}_{ji} < 0$	This case cannot occur.

It can be show that the actions in (2) and (3) conserve the RCOC.

Running Time Analysis

Theorem: *In a scaling phase, the algorithm performs at most m augmentations. Overall, the algorithm performs O(m log U) augmentations, and its running time equals solving O(m log U) shortest path problems.*