Gurobi Seminar 3
Network Flows

Demetrios Papazaharias

UB INFORMS Student Chapter
Department of Industrial & Systems Engineering
University at Buffalo, SUNY

October 10, 2019

University at Buffalo The State University of New York

Welcome!

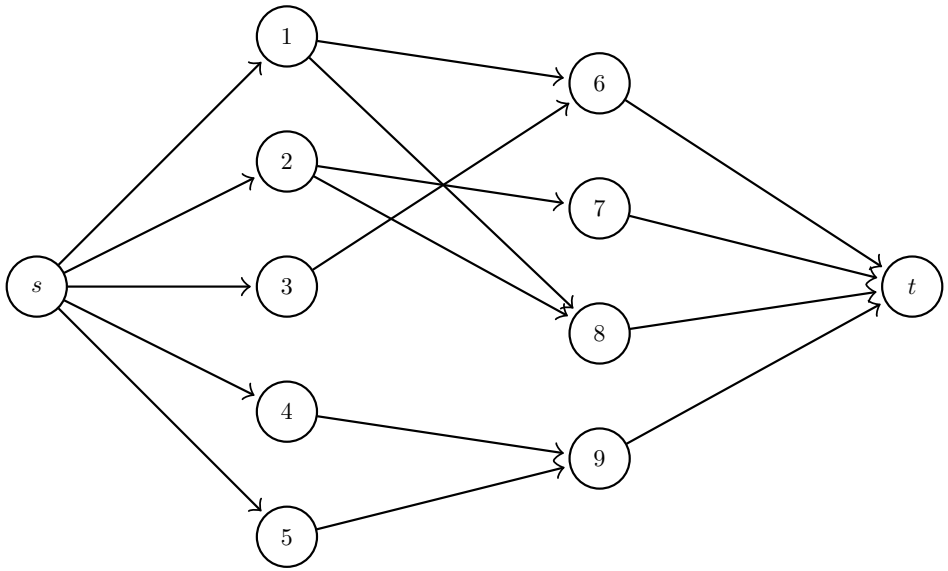
- Welcome to the third installment of our Gurobi series!
- Materials for this workshop can be found here:

  `github.com/Dpapazaharias1/UB-INFORMS-Gurobi-Seminar`

- Practice questions have been added for each session so far.
- In this workshop we will cover the minimum cost network flow problem

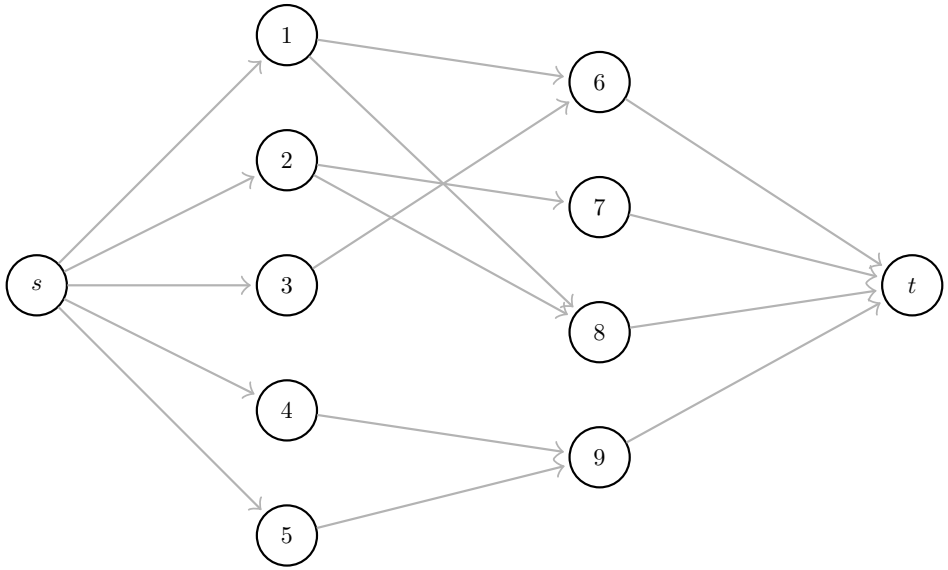
Problem Description

Consider a directed graph or *digraph* $G = (N, A)$ where N is a set of N nodes and A is a set of arcs.



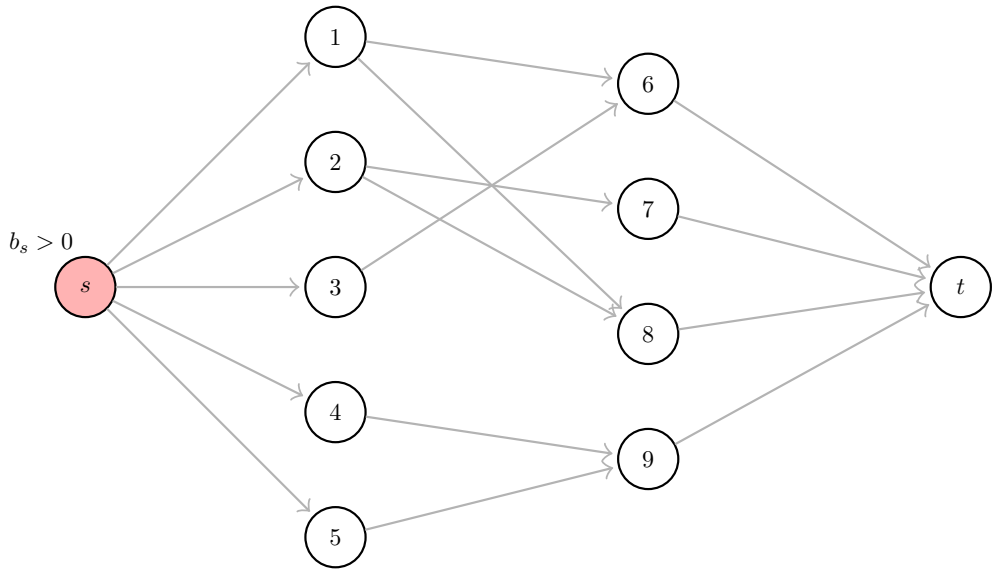
Problem Description - Nodes

Associated with each node $i \in N$ is a parameter b_i which represents its available supply or demand



Problem Description - Nodes

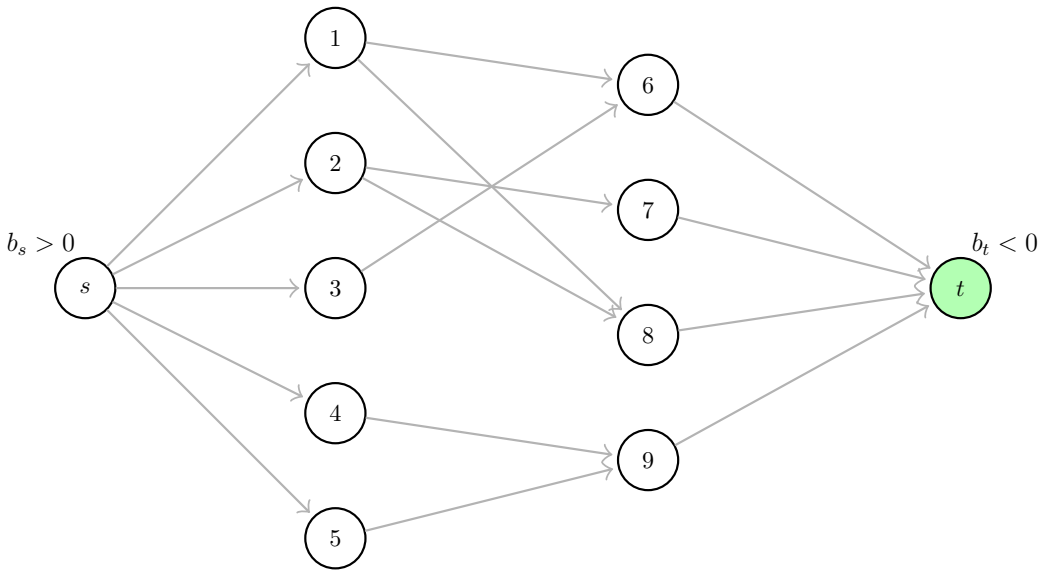
Associated with each node $i \in N$ is a parameter b_i which represents its available supply or demand



If $b_i > 0$, then i is a *source*. Network flows originate from sources.

Problem Description - Nodes

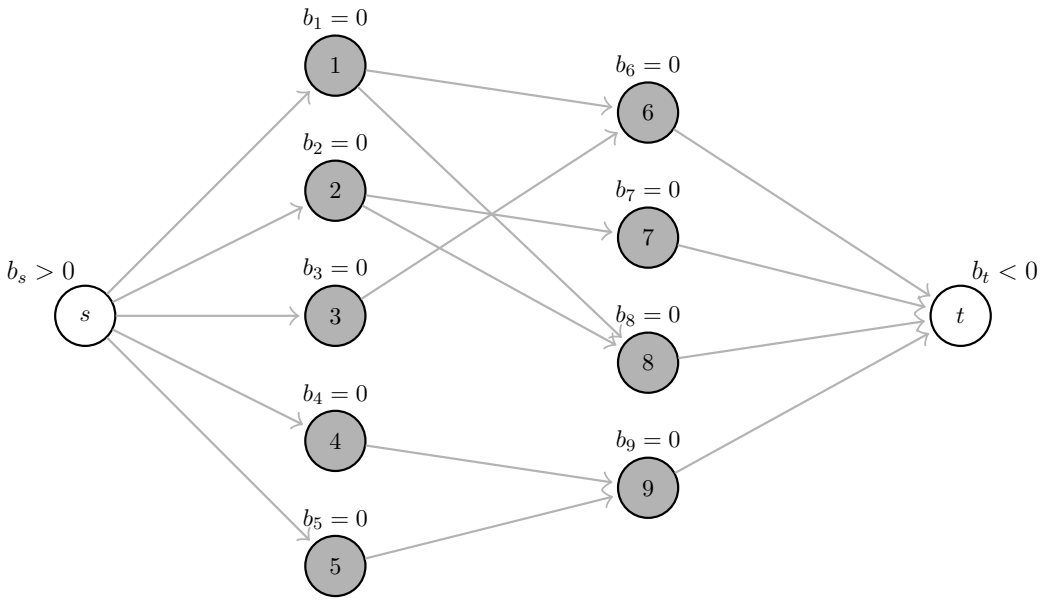
Associated with each node $i \in N$ is a parameter b_i which represents its available supply or demand



If $b_i < 0$, then i is a *sink*. Network flows terminate at sinks.

Problem Description - Nodes

Associated with each node $i \in N$ is a parameter b_i which represents its available supply or demand



If $b_i = 0$, then i is an *intermediate node*. All flow that enters must leave.

Problem Description - Arcs

Goal: Satisfy node demands by pushing flow through the arcs

- Let x_{ij} represent the amount of flow on arc (i, j)

Problem Description - Arcs

Goal: Satisfy node demands by pushing flow through the arcs

- Let x_{ij} represent the amount of flow on arc (i, j)
- Let c_{ij} represent the per unit cost along arc (i, j)

Problem Description - Arcs

Goal: Satisfy node demands by pushing flow through the arcs

- Let x_{ij} represent the amount of flow on arc (i, j)
- Let c_{ij} represent the per unit cost along arc (i, j)
- Let u_{ij} be the capacity of arc (i, j)

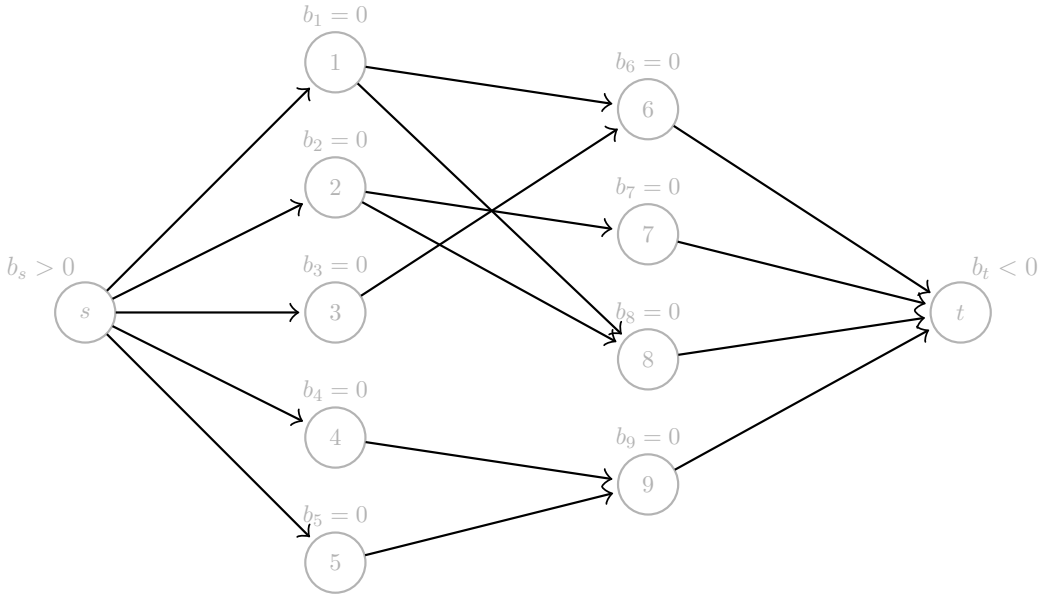
Problem Description - Arcs

Goal: Satisfy node demands by pushing flow through the arcs

- Let x_{ij} represent the amount of flow on arc (i, j)
- Let c_{ij} represent the per unit cost along arc (i, j)
- Let u_{ij} be the capacity of arc (i, j)
- Let ℓ_{ij} be the demand of arc (i, j)

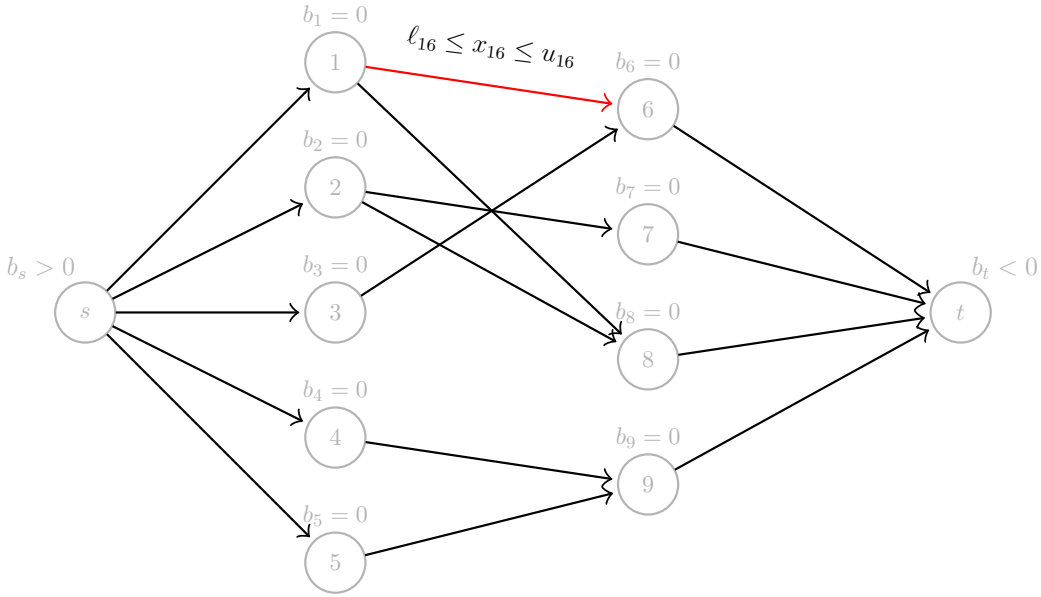
Problem Description - Arcs

Goal: Satisfy node demands by pushing flow through the arcs



Problem Description - Arcs

Goal: Satisfy node demands by pushing flow through the arcs



Minimum Cost Network Flow

- Decisions
  - x_{ij} - amount of flow pushed across arc (i, j)
- Objective
  - Minimize the shipping cost over the network.
- Constraints
  - Satisfy the supply/demand of every node. (Net flow of $i = b_i$)
  - Do not violate arc capacity
  - Satisfy the arc demand

Objective

Objective: Minimize the shipping cost over the network

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

```
from gurobipy import *

def minCostFlow(Nodes, Arcs, Supply, cost, lb, ub):

    model = Model()

    x = {}
    for i, j in Arcs:
        x[i, j] = model.addVar(vtype=GRB.CONTINUOUS, lb = lb[i,j], ub = ub[i,j], obj = cost[i,j])

    model.modelSense = GRB.MINIMIZE
    model.update()
```

Objective

Objective: Minimize the shipping cost over the network

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

```
from gurobipy import *

def minCostFlow(Nodes, Arcs, Supply, cost, lb, ub):

    model = Model()

    x = {}
    for i, j in Arcs:
        x[i, j] = model.addVar(vtype=GRB.CONTINUOUS, lb = lb[i,j], ub = ub[i,j], obj = cost[i,j])

    model.modelSense = GRB.MINIMIZE
    model.update()
```

Objective

Objective: Minimize the shipping cost over the network

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

```
from gurobipy import *

def minCostFlow(Nodes, Arcs, Supply, cost, lb, ub):

    model = Model()

    x = {}
    for i, j in Arcs:
        x[i, j] = model.addVar(vtype=GRB.CONTINUOUS, lb = lb[i,j], ub = ub[i,j], obj = cost[i,j])

    model.modelSense = GRB.MINIMIZE
    model.update()
```

$$l_{ij} \leq x_{ij} \leq u_{ij}$$

Objective

Objective: Minimize the shipping cost over the network

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

```
from gurobipy import *

def minCostFlow(Nodes, Arcs, Supply, cost, lb, ub):

    model = Model()

    x = {}
    for i, j in Arcs:
        x[i, j] = model.addVar(vtype=GRB.CONTINUOUS, lb = lb[i,j], ub = ub[i,j], obj = cost[i,j])

    model.modelSense = GRB.MINIMIZE
    model.update()
```

Balance Flow Constraint

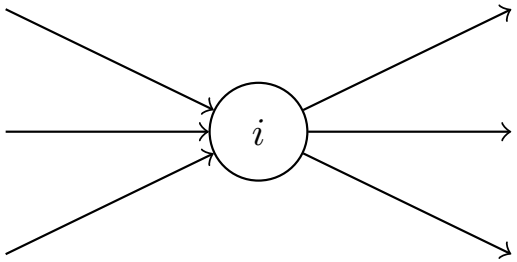
Satisfy the supply/demand of every node. (Net flow of $i = b_i$)

$$\sum_{j \in \delta^+(i)} x_{ij} - \sum_{j \in \delta^-(i)} x_{ji} = b_i \quad i \in N$$

$$\text{flow leaving} - \text{flow entering} = \text{supply/demand}$$

Where

- $\delta^+(i)$ (forward star) - the set of arcs leaving node i
- $\delta^-(i)$ (reverse star) - the set of arcs entering node i



Balance Flow Constraint

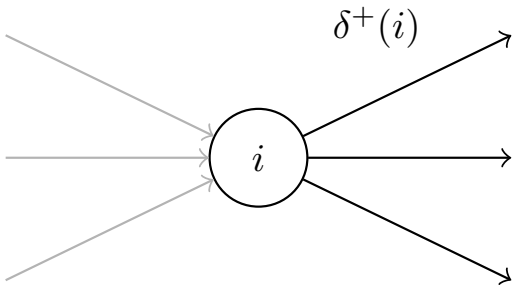
Satisfy the supply/demand of every node. (Net flow of $i = b_i$)

$$\sum_{j \in \delta^+(i)} x_{ij} - \sum_{j \in \delta^-(i)} x_{ji} = b_i \quad i \in N$$

$$\text{flow leaving} - \text{flow entering} = \text{supply/demand}$$

Where

- $\delta^+(i)$ (forward star) - the set of arcs leaving node i
- $\delta^-(i)$ (reverse star) - the set of arcs entering node i



Balance Flow Constraint

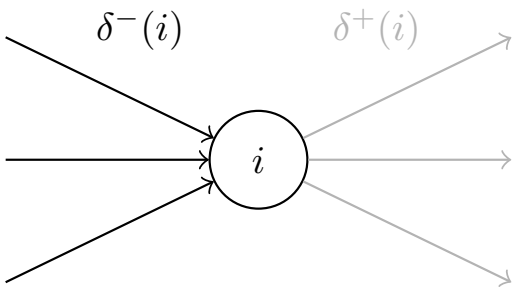
Satisfy the supply/demand of every node. (Net flow of $i = b_i$)

$$\sum_{j \in \delta^+(i)} x_{ij} - \sum_{j \in \delta^-(i)} x_{ji} = b_i \quad i \in N$$

$$\text{flow leaving} - \text{flow entering} = \text{supply/demand}$$

Where

- $\delta^+(i)$ (forward star) - the set of arcs leaving node i
- $\delta^-(i)$ (reverse star) - the set of arcs entering node i



Gurobi Object - tuplelist

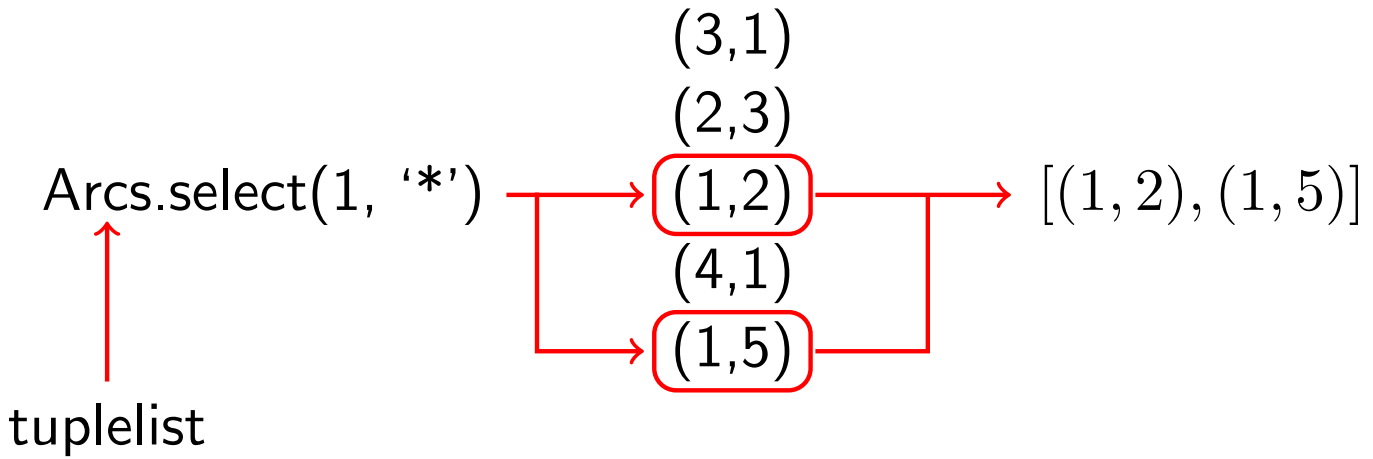
The `tuplelist` class is designed to efficiently build sub-lists from a list of tuples.

- A `tuple` in python is a collection that is ordered and unchangeable
- Written with round brackets e.g. `("apples", "bananas")`
- Same methods as `lists` except for those that change the elements of the `tuple`
- To create a `tuplelist` object pass a `list` of `tuples` e.g.
  `l = tuplelist([(3,1), (2,3), (1,2), (4,1), (1,5)])`

Balance Flow Constraint

$$\sum_{j \in \delta^+(i)} x_{ij} - \sum_{j \in \delta^-(i)} x_{ji} = b_i \quad i \in N$$

```
for v in Nodes:
    model.addConstr(quicksum(x[i,j] for i, j in Arcs.select(v, '*'))-
        quicksum(x[j, i] for j, i in Arcs.select('*', v)) == Supply[v], name="node %s" %v)
```



Minimum Cost Network Flow

$$\min \quad \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{j\in\delta^+(i)} x_{ij} - \sum_{j\in\delta^-(i)} x_{ji} = b_i \quad i \in N$$

$$\ell_{ij} \leq x_{ij} \leq u_{ij} \quad (i,j) \in A$$