5. Network flow problems

- Example: Sailco
- Minimum-cost flow problems
- Transportation problems
- Shortest/longest path problems
- Max-flow problems
- Integer solutions

Example: Sailco

Sailco manufactures sailboats. During the next 4 months the company must meet the following demands for their sailboats:

Month	1	2	3	4
Number of boats	40	60	70	25

At the beginning of Month 1, Sailco has 10 boats in inventory. Each month it must determine how many boats to produce. During any month, Sailco can produce up to 40 boats with regular labor and an unlimited number of boats with overtime labor. Boats produced with regular labor cost \$400 each to produce, while boats produced with overtime labor cost \$450 each. It costs \$20 to hold a boat in inventory from one month to the next. Find the production and inventory schedule that minimizes the cost of meeting the next 4 months' demands.

Example: Sailco

Summary of problem data:

- Regular labor: \$400/boat (at most 40 boats/month).
- Overtime labor: \$450/boat (no monthly limit).
- Holding a boat in inventory costs \$20/month.
- Inventory initially has 10 boats.
- Demand for next 4 months is:

Month	1	2	3	4
Number of boats	40	60	70	25

What are the decision variables?

Example: Sailco

Remember: Decision variables aren't always things that you decide directly!

For this problem, the decision variables are:

- x_1, x_2, x_3, x_4: boats produced each month with regular labor.
- y_1, y_2, y_3, y_4: boats produced each month with overtime.
- h_1, h_2, h_3, h_4, h_5: boats in inventory at start of each month.

Parameters:

- d_1, d_2, d_3, d_4: demand at each month
- h_1: initial number of boats in inventory

Example: Sailco

The constraints are:

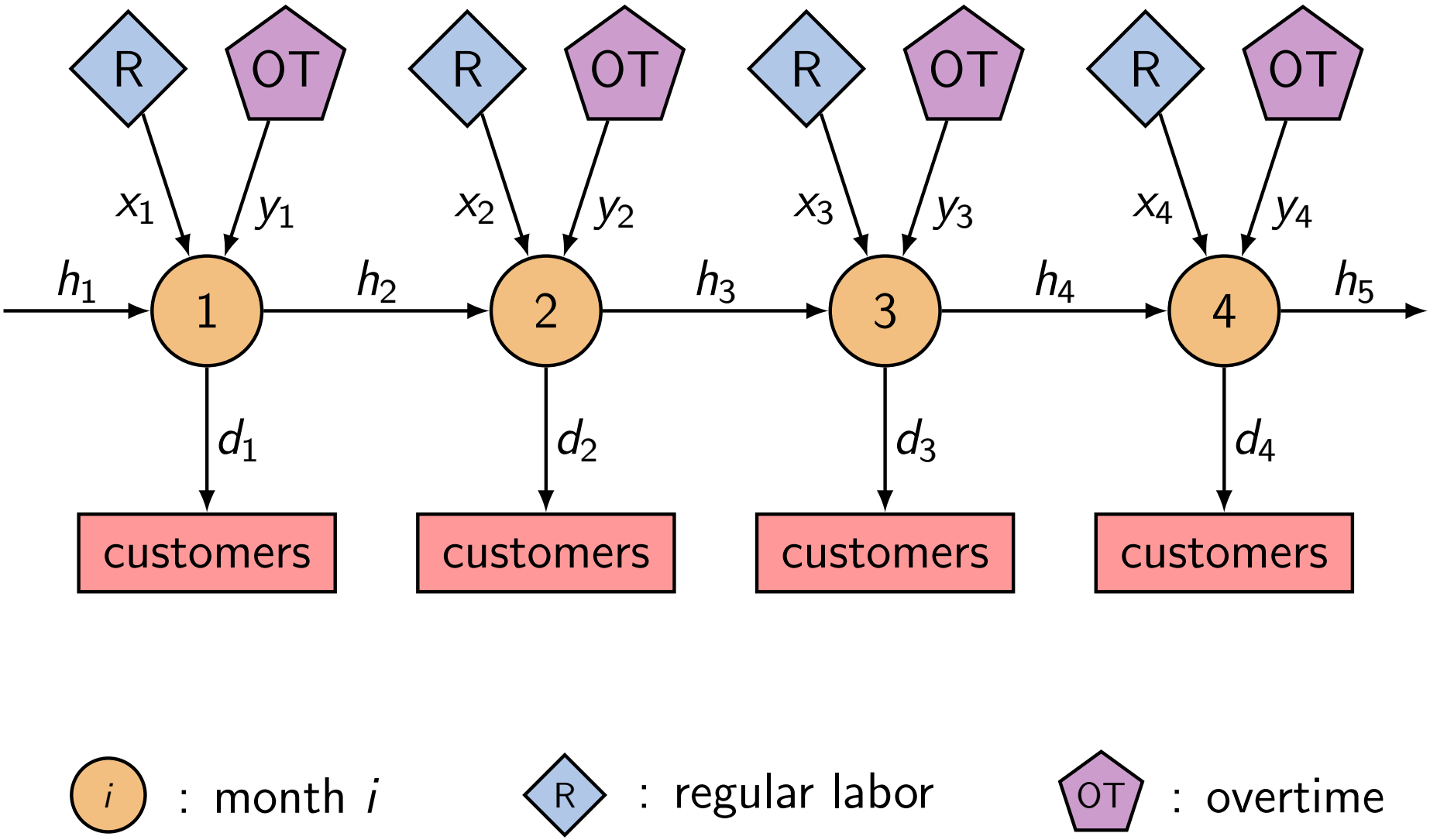
- $0 \leq x_i \leq 40$ (monthly limit of regular production)
- $y_i \geq 0$ (unlimited overtime production)
- Conservation of boats:
  - $h_i + x_i + y_i = d_i + h_{i+1}$ (for $i = 1, 2, 3, 4$)
  - $h_1 = 10$ (initial inventory)

The objective is to minimize:

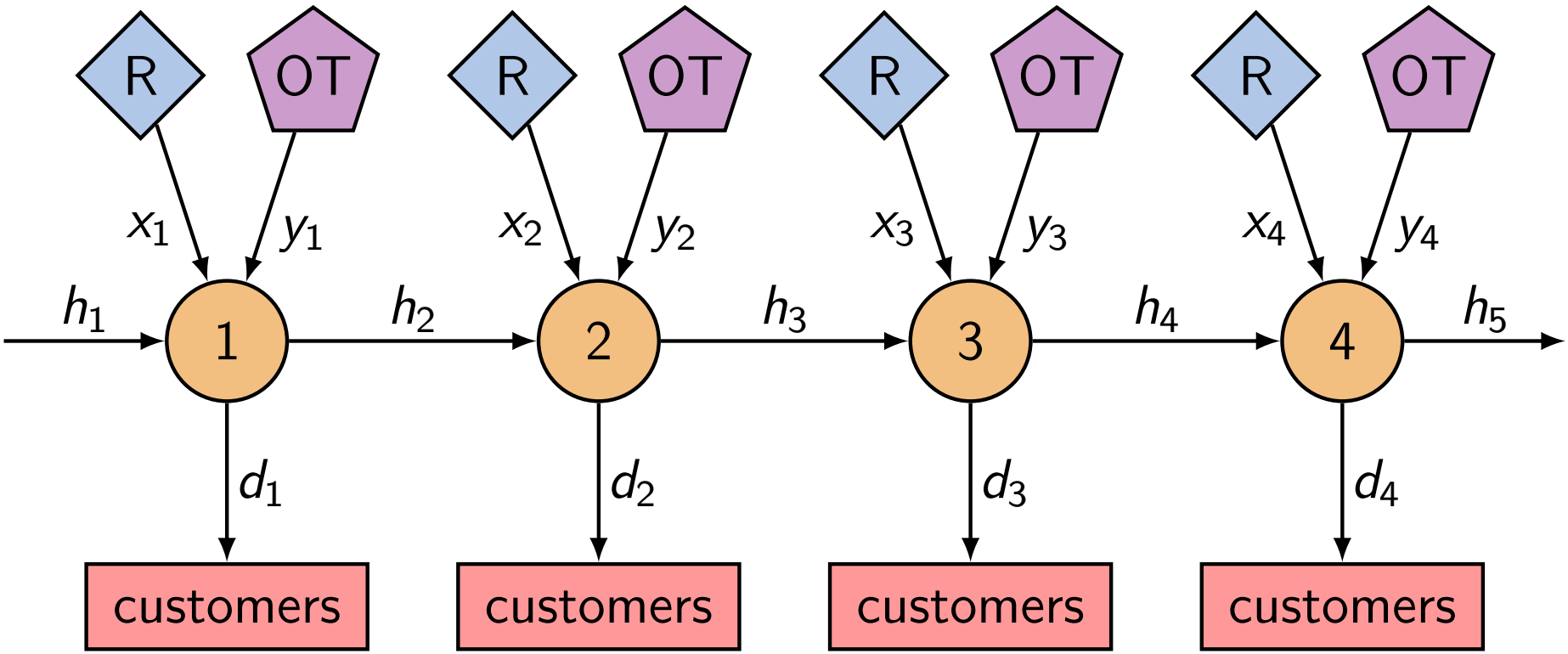
$$400 \sum_{i=1}^{4} x_i + 450 \sum_{i=1}^{4} y_i + 20 \sum_{i=1}^{5} h_i$$

Solution: Sailco.ipynb

Example: Sailco



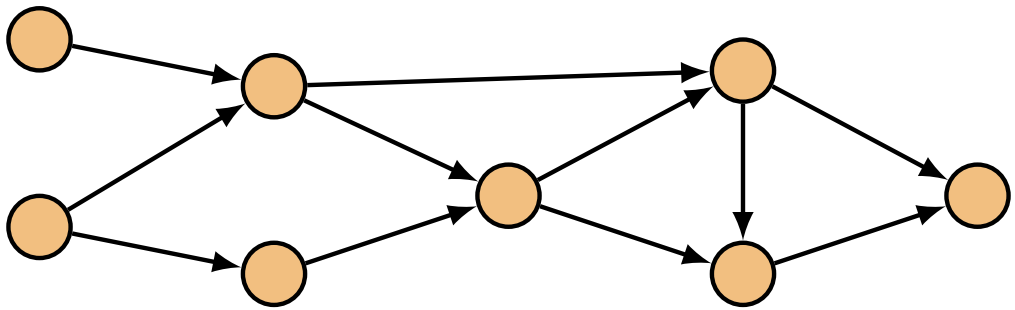
Example: Sailco



- Arrows indicate flow of boats
- conservation at nodes: $h_1 + x_1 + y_1 = d_1 + h_2$, etc.

Minimum-cost flow problems

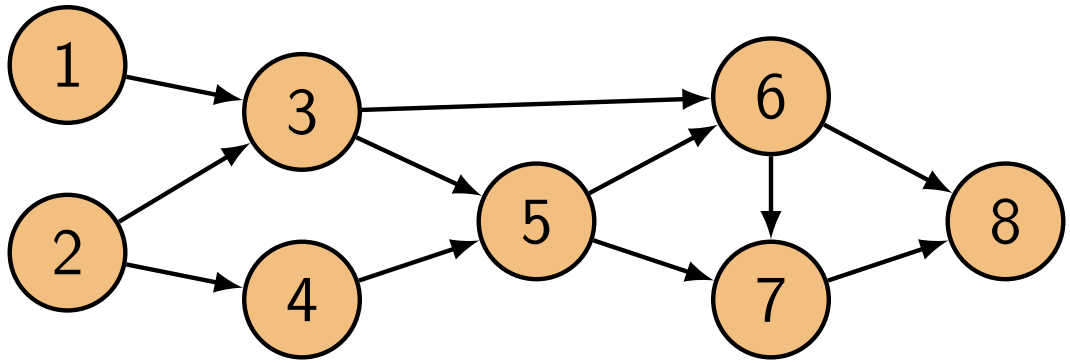
- Many optimization problems can be interpreted as **network flow problems** on a directed graph.



- Decision variables: **flow on each edge**.
- Edges have flow costs and capacity constraints
- Each node can either:
  - produce/supply flow (source)
  - consume/demand flow (sink)
  - conserve flow (relay)

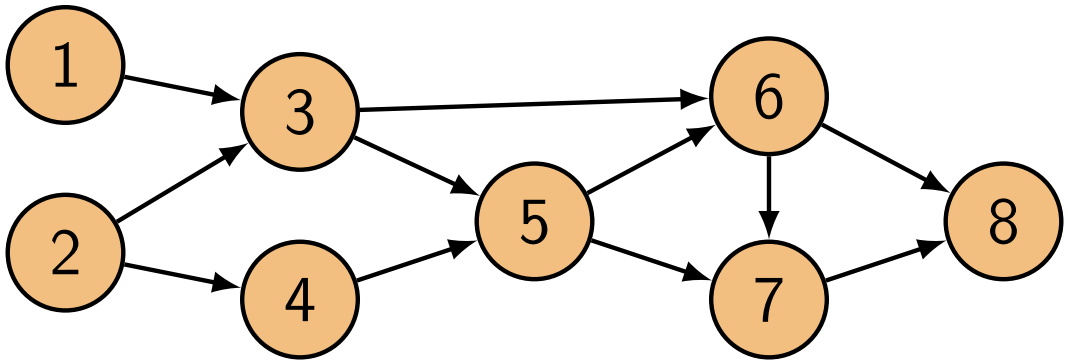
What is the minimum-cost feasible flow?

Minimum-cost flow problems



- The set of nodes: $\mathcal{N} = \{1, \ldots, 8\}$.
- The set of directed edges: $\mathcal{E} = \{(1,3), (2,3), (2,4), \ldots\}$.
- Each node $i \in \mathcal{N}$ supplies a flow b_i. Node i is called a *source* if $b_i > 0$, a *relay* if $b_i = 0$, and a *sink* if $b_i < 0$.
- **Decision variables**: x_{ij} is the flow on edge $(i,j) \in \mathcal{E}$.
- **Flow cost**: c_{ij} is cost per unit of flow on edge $(i,j) \in \mathcal{E}$.

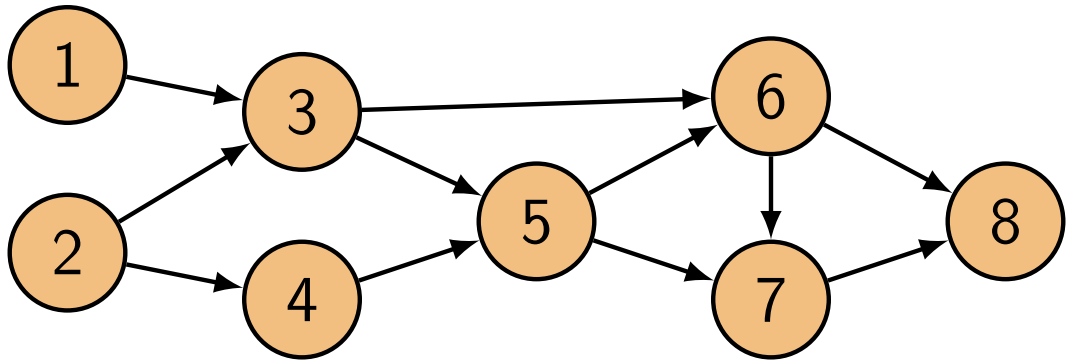
Minimum-cost flow problems



- **Decision variables**: x_{ij} is the flow on edge $(i, j) \in \mathcal{E}$.
- **Capacity constraints**: $p_{ij} \leq x_{ij} \leq q_{ij}$ $\quad \forall (i, j) \in \mathcal{E}$.
- **Conservation**: $\sum_{j \in \mathcal{N}} x_{kj} - \sum_{i \in \mathcal{N}} x_{ik} = b_k$ $\quad \forall k \in \mathcal{N}$.
- **Total cost**: $\sum_{(i,j) \in \mathcal{E}} c_{ij} x_{ij}$.

Note: $b_k, c_{ij}, p_{ij}, q_{ij}$ are *parameters*.

Minimum-cost flow problems



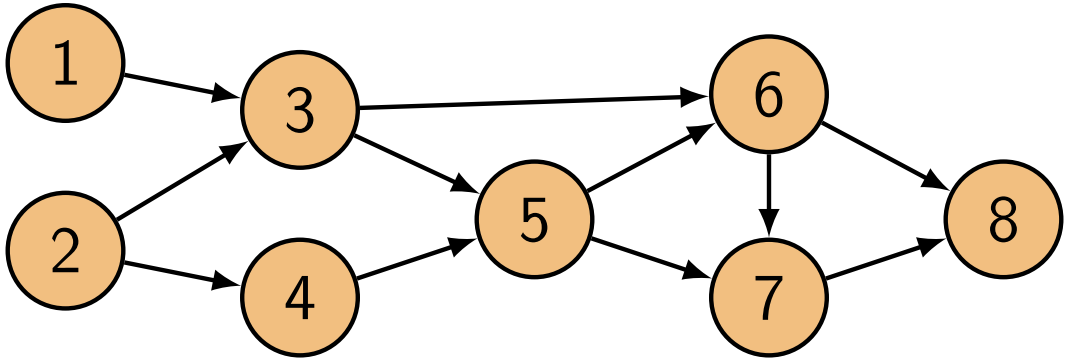
The entire model:

$$\underset{x_{ij} \in \mathbb{R}}{\text{minimize}} \quad \sum_{(i,j) \in \mathcal{E}} c_{ij} x_{ij}$$

$$\text{subject to:} \quad \sum_{j \in \mathcal{N}} x_{kj} - \sum_{i \in \mathcal{N}} x_{ik} = b_k \quad \forall k \in \mathcal{N}$$

$$p_{ij} \leq x_{ij} \leq q_{ij}$$

Minimum-cost flow problems

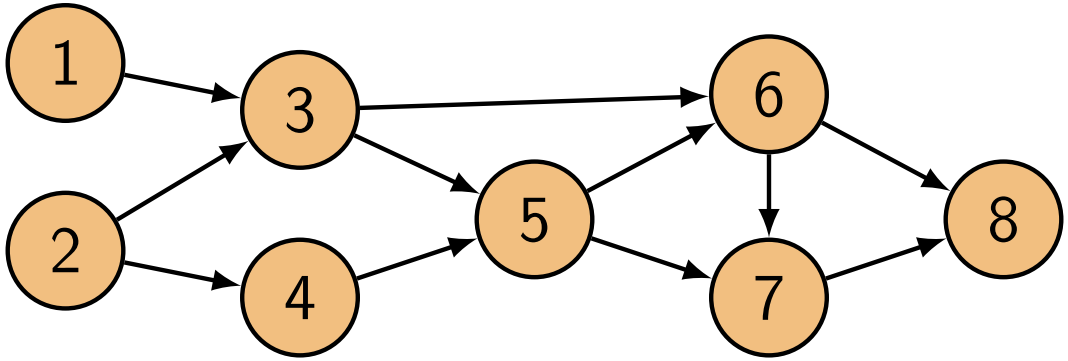


Expanded conservation constraint:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \end{bmatrix} \begin{bmatrix} x_{13} \\ x_{23} \\ x_{24} \\ x_{35} \\ x_{36} \\ x_{45} \\ x_{56} \\ x_{57} \\ x_{67} \\ x_{68} \\ x_{78} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \end{bmatrix}$$

$A =$ incidence matrix

Minimum-cost flow problems



The entire model (compact form):

$$\begin{array}{ll} \underset{x \in \mathbb{R}^{|\mathcal{E}|}}{\text{minimize}} & c^{\mathsf{T}} x \\ \text{subject to:} & Ax = b \\ & p \leq x \leq q \end{array}$$

Note: The incidence matrix A is a property of the graph.
It does not depend on which nodes are sources/sinks/relays.

Balanced problems

Each column of the incidence matrix has zero sum. In other words: $1^T A = 0$. Since $Ax = b$ is a constraint, we must therefore have: $1^T Ax = 1^T b = 0$. Therefore:

$$\sum_{i \in \mathcal{N}} b_i = 0 \quad \text{(total supply = total demand)}$$

- If $\sum_{i \in \mathcal{N}} b_i = 0$, the model is called **balanced**.
- Unbalanced models are *always* infeasible.
- Note: balanced models may still be infeasible.

Unbalanced models still make sense in practice, e.g. we may have excess supply or allow excess demand. These cases can be handled by making small modifications to the problem, such as changing "$=$" to "$\leq$".

Minimum-cost flow problems

Many problem types are actually min-cost flow models:

- transportation problems
- assignment problems
- transshipment problems
- shortest path problems
- max-flow problems

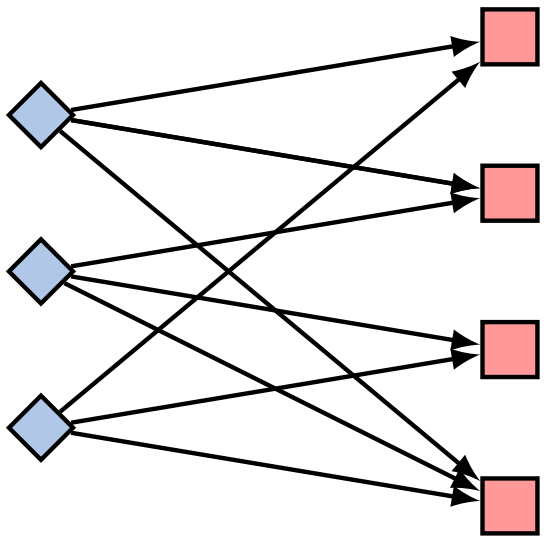
Let's look at these in more detail...

Legend:  : source  : relay  : sink

Transportation problems

The objective is to transport a particular commodity from several possible sources to several possible destinations while minimizing the total cost.

- Sources have known supply limits
- Destinations each have demands
- Edges may have capacity limits
- Each link has an associated cost



Transportation example

Millco has three wood mills and is planning three new logging sites. Each mill has a maximum capacity and each logging site can harvest a certain number of truckloads of lumber per day. The cost of a haul is \$2/mile of distance. If distances from logging sites to mills are given below, how should the hauls be routed to minimize hauling costs while meeting all demands?

Logging site	Distance to mill (miles) Mill A	Mill B	Mill C	Maximum truckloads/day per logging site
1	8	15	50	20
2	10	17	20	30
3	30	26	15	45
Mill demand (truckloads/day)	30	35	30	

Note: problem is balanced!

Source: J. Reeb and S. Leavengood, 2002

Transportation example

- Arrange nodes as: $\begin{bmatrix} 1 & 2 & 3 & A & B & C \end{bmatrix}$ (sources, sinks).
- Graph is fully connected. Incidence matrix:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

Julia code: Millco.ipynb

Transportation example

Logging site	Distance to mill (miles) Mill A	Mill B	Mill C	Maximum truckloads/day per logging site
1	8	15	50	20
2	10	17	20	30
3	30	26	15	45
Mill demand (truckloads/day)	30	35	30	

Solution is:

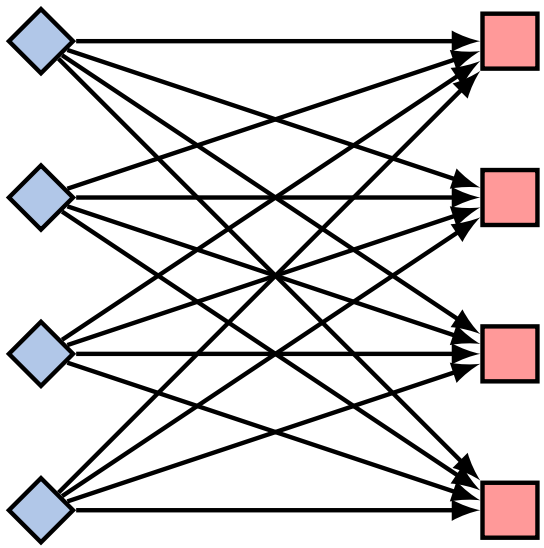
	A	**B**	**C**	
1	20	0	0	20
2	10	20	0	30
3	0	15	30	45
	30	35	30	

Similar to "magic squares" but without cell constraints

Assignment problems

We have n people and n tasks. The goal is to assign each person to a task. Each person has different preferences (costs) associated with performing each of the tasks. The goal is to find an assignment that minimizes the total cost.

- It's just a transportation problem!
- Each source has supply $= 1$
- Each sink has demand $= 1$
- Edges are unconstrained

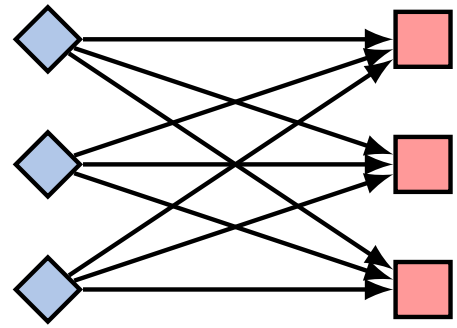


What about the integer constraint? More about this later...

Balanced assignment problems

List constraints one by one:

- $x_{ij} \geq 0$ for all i, j (nonnegativity)
- $\sum_j x_{ij} = 1$ for all i (sources provide 1)
- $\sum_i x_{ij} = 1$ for all j (sinks receive 1)



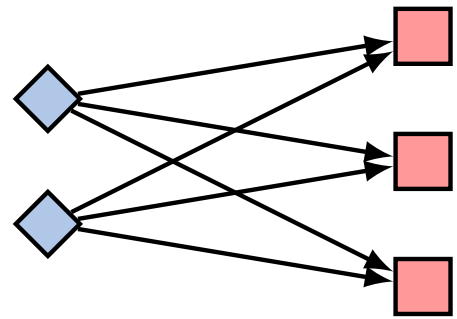
Alternative formulation using incidence matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

Unbalanced assignment problems

More sinks than sources:

- $x_{ij} \geq 0$ for all i, j (nonnegativity)
- $\sum_j x_{ij} = 1$ for all i (sources provide 1)
- $\sum_i x_{ij} \leq 1$ for all j (sinks receive ≤ 1)



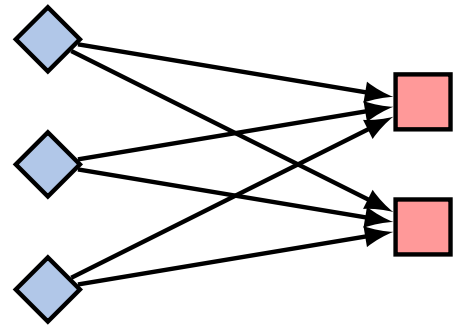
Alternative formulation using incidence matrix:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \end{bmatrix} \sim \begin{bmatrix} = & 1 \\ = & 1 \\ \geq & -1 \\ \geq & -1 \\ \geq & -1 \end{bmatrix}$$

Unbalanced assignment problems

More sources thank sinks:

- $x_{ij} \geq 0$ for all i, j (nonnegativity)
- $\sum_j x_{ij} \leq 1$ for all i (sources provide ≤ 1)
- $\sum_i x_{ij} = 1$ for all j (sinks receive 1)



Alternative formulation using incidence matrix:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ -1 & 0 & -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ x_{21} \\ x_{22} \\ x_{31} \\ x_{32} \end{bmatrix} \sim \begin{bmatrix} \leq & 1 \\ \leq & 1 \\ \leq & 1 \\ = & -1 \\ = & -1 \end{bmatrix}$$

Assignment example

The coach of a swim team needs to assign swimmers to a 200-yard medley relay team to compete in a tournament. The problem is that his best swimmers are good in more than one stroke, so it's not clear which swimmer to assign to which stroke. Here are the best times for each swimmer:

Stroke	Carl	Chris	David	Tony	Ken
Backstroke	37.7	32.9	33.8	37.0	35.4
Breaststroke	43.4	33.1	42.2	34.7	41.8
Butterfly	33.3	28.5	38.9	30.4	33.6
Freestyle	29.2	26.4	29.6	28.5	31.1

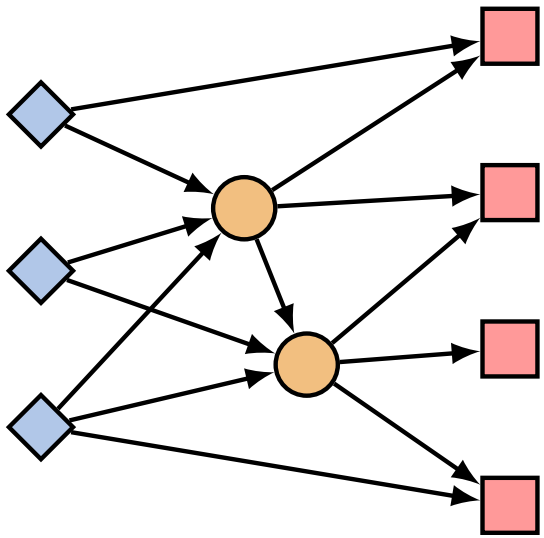
Julia code: Swim Relay.ipynb

Source: B. Van Roy and K. Mason

Transshipment problems

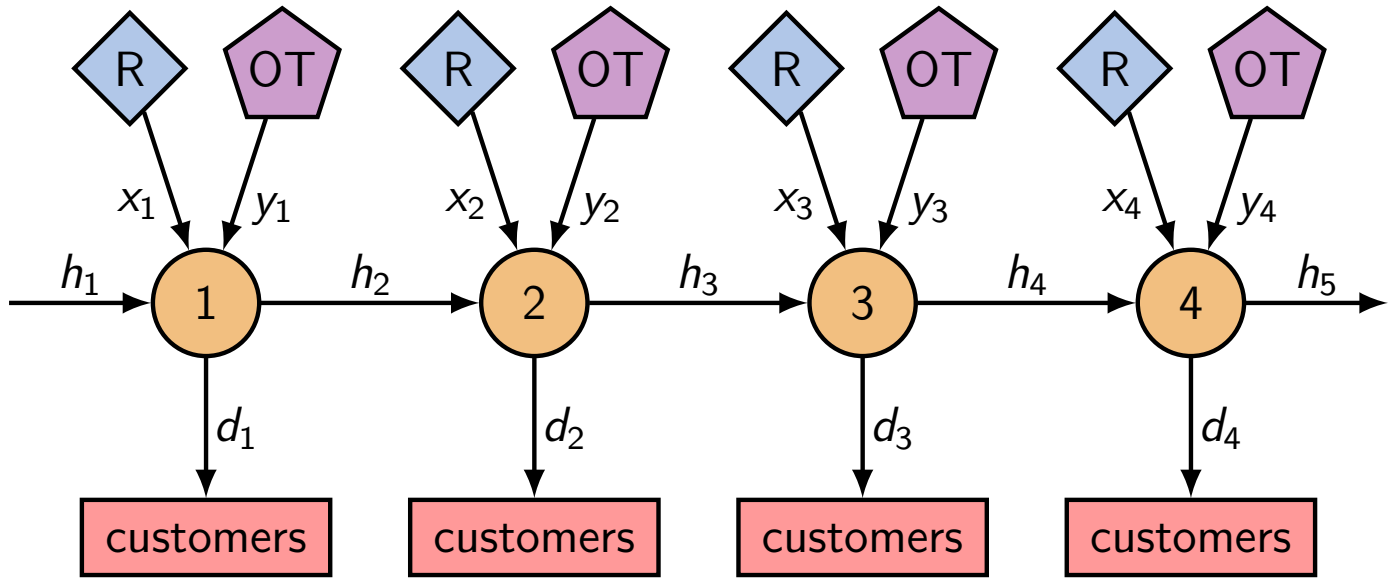
The same as a transportation problem, but in addition to sources and destinations, we also have warehouses that can store goods. The warehouses are **relay nodes**.

- Sources have known supply limits
- Destinations each have demands
- Links may have capacity limits
- Each link has an associated cost
- For warehouses, inflow $=$ outflow.



Sailco problem is a transshipment problem!

Transshipment example: Sailco

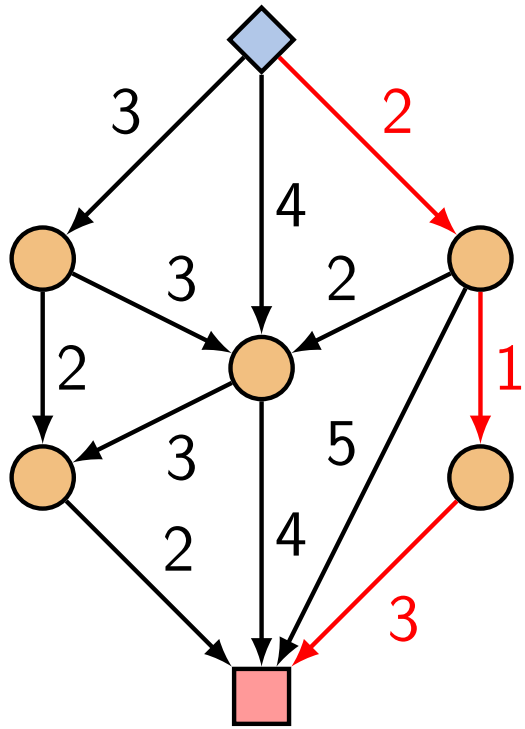


- The "warehouses" are the different months.
- Storing in inventory = shipping to the future.

Shortest/longest path problems

We have a directed graph and edge lengths. The goal is to find the shortest or longest path between two given nodes.

- Again, a transportation problem!
- Edge cost = length of path.
- The source has supply = 1
- The sink has demand = 1
- To find longest path, just change the min to a max!

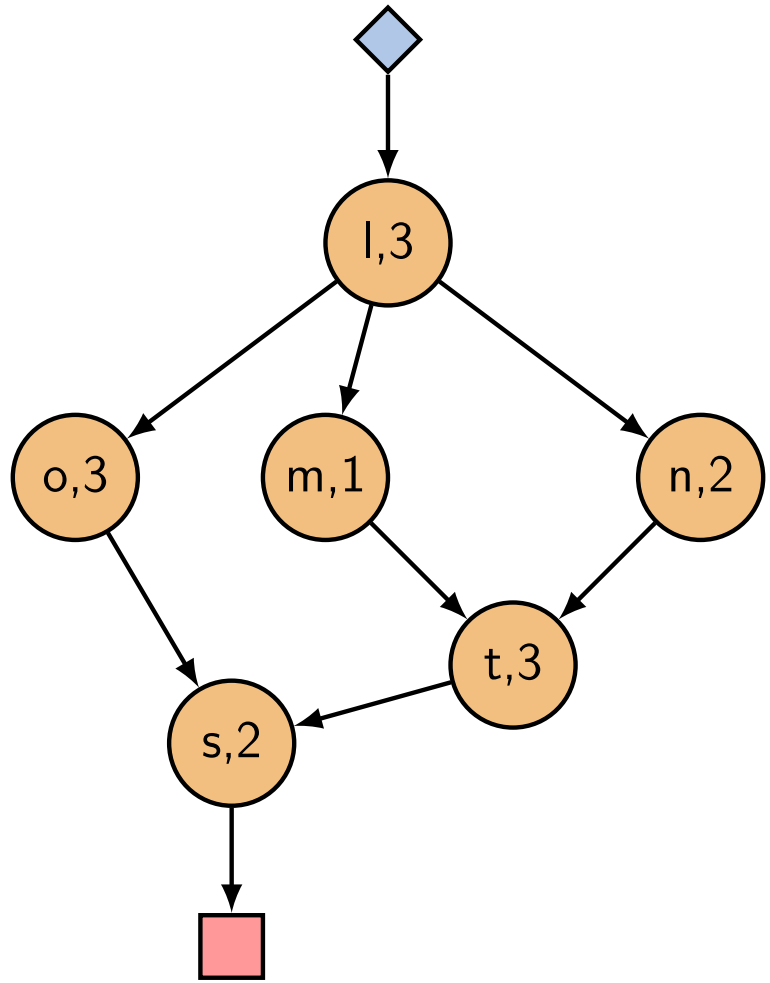


Again we need integer constraints on the edges...

Longest path example

The house building example is a longest path problem!

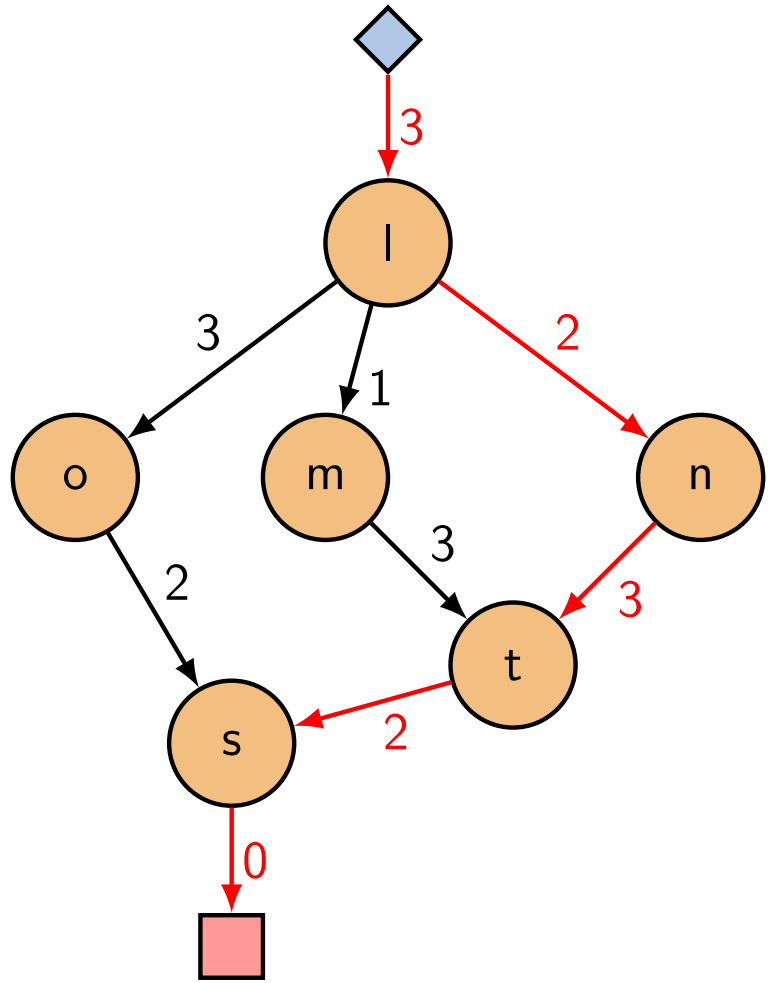
- Add source and sink nodes
- Move times out of nodes and onto preceding edges



Longest path example

The house building example is a longest path problem!

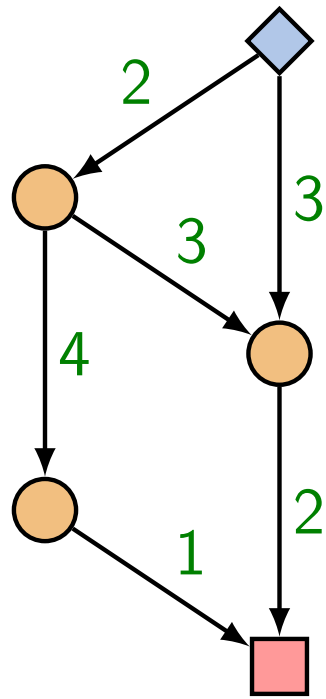
- Add source and sink nodes
- Move times out of nodes and onto preceding edges
- Each path says "it takes at least this long." Longest path gives the shortest time we have to wait.



Max-flow problems

We are given a directed graph and **edge capacities**. Find the maximum flow that we can push from source to sink.

- Edges have max capacities
- Flow can split!
- notions of supply and demand don't make sense...
- add a feedback path and make every node a relay!



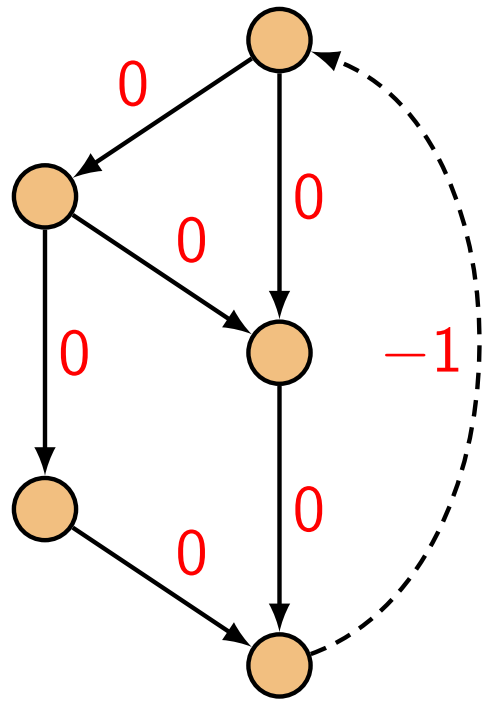
(edge capacity)

Max-flow problems

We are given a directed graph and **edge capacities**. Find the maximum flow that we can push from source to sink.

- Edges have max capacities
- Flow can split!
- notions of supply and demand don't make sense...
- add a feedback path and make every node a relay!

Solve minimum-cost flow where feedback path has cost (-1) and all other paths have zero cost.



Integer solutions

Some minimum-cost flow problems require integer solutions (assignment problems and shortest path problems). Is there a way of guaranteeing integer solutions? **yes!**

Definition: A matrix A is **totally unimodular** (TU) if every square submatrix of A has determinant 0, 1, or -1.

- The definition includes 1×1 submatrices, so every entry of A must be 0, 1, or -1.
- ex. $\begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ is TU but $\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ is not.

TU matrices

Theorem: If A is TU and b is an integer vector, then the vertices of $\{x \mid Ax \sim b\}$ have integer coordinates, where "$\sim$" is any combination of $\{\geq, =, \leq\}$.

Explanation: Vertices occur at intersections of hyperplanes. For each vertex, we can select the corresponding rows of A and b and obtain a (square) system of linear equations $\hat{A}x = \hat{b}$ whose solution is the vertex.

Since $\det(\hat{A}) = \pm 1$, the inverse $\hat{A}^{-1} = \det(\hat{A})^{-1}\text{adj}(\hat{A})$ has integer coordinates, and so does the vertex $x = \hat{A}^{-1}\hat{b}$.

TU matrix properties

- If A is TU, then so are $-A$, A^{T}, $\begin{bmatrix} A & I \end{bmatrix}$, and $\begin{bmatrix} A & -I \end{bmatrix}$.
- Removing any row or column preserves TU property.
- Multiplying any row or column by -1 preserves TU.
- Every incidence matrix is TU.

Important consequence: If a minimum-cost flow problem has integer supplies, integer demands, and integer capacities, then there is a minimum-cost **integer** flow.

- cost vector need not be integer.
- every assignment problem is an LP.
- every shortest path problem is an LP.

Example

Min-cost flow with integer supply/demand/capacity (b, p, q):

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^{\mathsf{T}}x \\ \text{subject to:} & Ax = b \\ & p \leq x \leq q \end{array}$$

Can rewrite as:

$$\begin{array}{r} Ax = b \\ x \geq p \\ x \leq q \end{array} \quad \Longrightarrow \quad \begin{bmatrix} A \\ I \\ I \end{bmatrix} x \sim \begin{bmatrix} b \\ p \\ q \end{bmatrix}$$

Because left-hand side is TU and right-hand side is integer, vertices are integer, and there is a min-cost flow that is integer.

Warning!

Can also rewrite as:

$$\begin{aligned} Ax &\geq b \\ Ax &\leq b \\ x &\geq p \\ x &\leq q \end{aligned} \quad \Longrightarrow \quad \begin{bmatrix} A \\ A \\ I \\ I \end{bmatrix} x \sim \begin{bmatrix} b \\ b \\ p \\ q \end{bmatrix}$$

The solutions are still integer of course, but now the left-hand side is **not necessarily TU** (concatenation of TU matrices does not preserve TU property in general!).

TU is a **sufficient condition**. TU guarantees integer solutions, but you can have integer solutions without TU.

Solving network flow problems

- Minimum-cost flow problems are LPs; this should provide useful intuition about the nature of optimal flows.
- In practice, generic LP solvers are typically **not used** for solving large network flow problems.
  - some solvers can leverage sparsity of large matrices.
  - specialized solvers for network flows, assignment, etc.