MATH 380

Hemanshu Kaul

kaul@iit.edu

Vertex cover problem as an optimization problem

Let $G = (V(G), E(G))$ be the given graph.

Suppose $V(G) = \{v_1, v_2, \ldots, v_n\}$.

Vertex cover problem as an optimization problem

Let $G = (V(G), E(G))$ be the given graph.

Suppose $V(G) = \{v_1, v_2, \dots, v_n\}$.

For each $v_i \in V(G)$, we have to decide whether or not to include it in our set S, vertex cover.

Let $x_i = \begin{cases} 1 & \text{if } v_i \in S \\ 0 & \text{if } v_i \notin S \end{cases}$

Using the variables x_i, how will you model the requirement that each edge must be incident to at least one vertex in S?

Vertex cover problem as an optimization problem

Let $G = (V(G), E(G))$ be the given graph.

Suppose $V(G) = \{v_1, v_2, \dots, v_n\}$.

For each $v_i \in V(G)$, we have to decide whether or not to include it in our set S, vertex cover.

Let $x_i = \begin{cases} 1 & \text{if } v_i \in S \\ 0 & \text{if } v_i \notin S \end{cases}$

Using the variables x_i, how will you model the requirement that each edge must be incident to at least one vertex in S?

$$x_i + x_j \geq 1 \quad \text{for all } v_iv_j \in E(G)$$

Objective function?

Vertex cover problem as an optimization problem

Let $G = (V(G), E(G))$ be the given graph.

Suppose $V(G) = \{v_1, v_2, \ldots, v_n\}$.

For each $v_i \in V(G)$, we have to decide whether or not to include it in our set S, vertex cover.

Let $x_i = \begin{cases} 1 & \text{if } v_i \in S \\ 0 & \text{if } v_i \notin S \end{cases}$ for each $v_i \in V(G)$, $i = 1, 2, \ldots, n$.

$$\min \sum_{i=1}^{n} x_i$$

[minimize total # vertices picked]

s.t.

$$x_i + x_j \geq 1 \quad \forall v_i v_j \in E(G)$$

[each edge is "covered" by at least one vertex]

$$x_i \in \{0, 1\} \quad \forall i = 1, \ldots, n$$

Vertex cover problem as an optimization problem

Let $G=(V(G),E(G))$ be the given graph.

Suppose $V(G)=\{v_1, v_2, \dots, v_n\}$.

For each $v_i \in V(G)$, we have to decide whether or not to include it in our set S, vertex cover.

Let $x_i = \begin{cases} 1 & \text{if } v_i \in S \\ 0 & \text{if } v_i \notin S \end{cases}$ for each $v_i \in V(G)$, $i=1,2,\dots,n$.

$$\min \sum_{i=1}^{n} x_i$$

[minimize total # vertices picked]

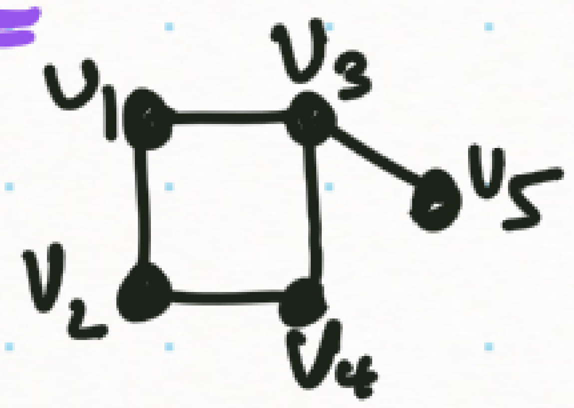
s.t.

$$x_i + x_j \geq 1 \quad \forall v_i v_j \in E(G)$$

[each edge is "covered" by at least one vertex]

$$x_i \in \{0,1\} \quad \forall i=1,\dots,n$$

e.g.



$\min\ x_1+x_2+x_3+x_4+x_5$ s.t.

$x_1+x_2 \geq 1$	$x_2+x_4 \geq 1$	$x_3+x_5 \geq 1$
$x_1+x_3 \geq 1$	$x_3+x_4 \geq 1$	

$x_1, x_2, x_3, x_4, x_5 \in \{0,1\}$

Most Binary Optimization Problems are very hard to solve.
We use different ideas to get approximate / non optimal solutions

1. Heuristics based on greedy algorithms / local search.

2. Randomized Rounding schemes.

Consider $\min \vec{c}^T \vec{x}$ s.t. $A\vec{x} \le \vec{b}$, $x_1, x_2, \ldots, x_n \in \{0,1\}$ $\xrightarrow{\text{LP relaxation}}$ $\min \vec{c}^T \vec{y}$ s.t. $A\vec{y} \le \vec{b}$, $y_1, \ldots, y_n \in [0,1]$

Most Binary Optimization Problems are very hard to solve.
We use different ideas to get approximate / non optimal solutions

① Heuristics based on greedy algorithms / local search.

② Randomized Rounding schemes.

Consider $\min \ \vec{c}^T\vec{x}$
s.t. $A\vec{x} \le \vec{b}$
$x_1, x_2, \ldots, x_n \in \{0,1\}$

$\xrightarrow{\text{LP relaxation}}$

$\min \ \vec{c}^T\vec{y}$
s.t. $A\vec{y} \le \vec{b}$
$y_1, \ldots, y_n \in [0,1]$

LP, easy to solve for $y_1, \ldots, y_n \in [0,1]$

Most Binary Optimization Problems are very hard to solve.
We use different ideas to get approximate / non optimal solutions

1. Heuristics based on greedy algorithms / local search.

2. Randomized Rounding schemes.

Consider

$$\begin{aligned} \min \quad & \vec{c}^T \vec{x} \\ \text{s.t.} \quad & A\vec{x} \leq \vec{b} \\ & x_1, x_2, \ldots, x_n \in \{0,1\} \end{aligned} \xrightarrow{\text{LP relaxation}} \begin{aligned} \min \quad & \vec{c}^T \vec{y} \\ \text{s.t.} \quad & A\vec{y} \leq \vec{b} \\ & y_1, \ldots, y_n \in [0,1] \end{aligned}$$

LP, easy to solve for $y_1, \ldots, y_n \in [0,1]$

$\xleftarrow{\text{"round"}}$ each $y_i \in [0,1]$ to $x_i \in \{0,1\}$

e.g. $x_i = \begin{cases} 1 & \text{if } y_i \geq 0.5 \\ 0 & \text{if } y_i < 0.5 \end{cases}$

Most Binary Optimization Problems are very hard to solve.
We use different ideas to get approximate / non optimal solutions

① Heuristics based on greedy algorithms / local search.

② Randomized Rounding schemes.

Consider
$$\min \ \vec{c}^T \vec{x}$$
$$\text{s.t.} \ A\vec{x} \le \vec{b}$$
$$x_1, x_2, \ldots, x_n \in \{0,1\}$$

$\xrightarrow{\text{LP relaxation}}$

$$\min \ \vec{c}^T \vec{y}$$
$$\text{s.t.} \ A\vec{y} \le \vec{b}$$
$$y_1, \ldots, y_n \in [0,1]$$

LP, easy to solve for $y_1, \ldots, y_n \in [0,1]$

"round" each $y_i \in [0,1]$ to $x_i \in \{0,1\}$

e.g. $x_i = \begin{cases} 1 & \text{if } y_i \geq 0.5 \\ 0 & \text{if } y_i < 0.5 \end{cases}$

Check whether these variables satisfy $A\vec{x} \le \vec{b}$

Issues
- may not be satisfied
- this solution might be far from optimal solution.

Most Binary Optimization Problems are very hard to solve.
We use different ideas to get approximate / non optimal solutions

① Heuristics based on greedy algorithms / local search.

② Randomized Rounding schemes.

Consider $\min \vec{c}^T \vec{x}$
s.t. $A\vec{x} \leq \vec{b}$
$x_1, x_2, \dots, x_n \in \{0,1\}$

$\xrightarrow{\text{LP relaxation}}$

$\min \vec{c}^T \vec{y}$
s.t. $A\vec{y} \leq \vec{b}$
$y_1, \dots, y_n \in [0,1]$

LP, easy to solve for $y_1, \dots, y_n \in [0,1]$

"randomly round" each $y_i \in [0,1]$ to $x_i \in \{0,1\}$

Most Binary Optimization Problems are very hard to solve.
We use different ideas to get approximate / non optimal solutions

① Heuristics based on greedy algorithms / local search.

② Randomized Rounding schemes.

Consider

$$\begin{aligned} \min \quad & \vec{c}^T \vec{x} \\ \text{s.t.} \quad & A\vec{x} \leq \vec{b} \\ & x_1, x_2, \ldots, x_n \in \{0,1\} \end{aligned} \xrightarrow{\text{LP relaxation}} \begin{aligned} \min \quad & \vec{c}^T \vec{y} \\ \text{s.t.} \quad & A\vec{y} \leq \vec{b} \\ & y_1, \ldots, y_n \in [0,1] \end{aligned}$$

LP, easy to solve for $y_1, \ldots, y_n \in [0,1]$

"randomly round" each $y_i \in [0,1]$ to $x_i \in \{0,1\}$

Flip a coin with $\mathbb{P}[\text{heads}] = y_i$
$\mathbb{P}[\text{tails}] = 1 - y_i$

$$x_i = \begin{cases} 1 & \text{with probab. } y_i \\ 0 & \text{with probab. } 1-y_i \end{cases}$$

Most Binary Optimization Problems are very hard to solve.
We use different ideas to get approximate / non optimal solutions

① Heuristics based on greedy algorithms / local search.

② Randomized Rounding schemes.

Consider $\min \vec{c}^T \vec{x}$
s.t. $A\vec{x} \leq \vec{b}$
$x_1, x_2, \dots, x_n \in \{0,1\}$

$\xrightarrow{\text{LP relaxation}}$

$\min \vec{c}^T \vec{y}$
s.t. $A\vec{y} \leq \vec{b}$
$y_1, \dots, y_n \in [0,1]$

LP, easy to solve for $y_1, \dots, y_n \in [0,1]$

"randomly round" each $y_i \in [0,1]$ to $x_i \in \{0,1\}$

Flip a coin with $\mathbb{P}[\text{heads}] = y_i$
$\mathbb{P}[\text{tails}] = 1 - y_i$

$$x_i = \begin{cases} 1 & \text{with probab. } y_i \\ 0 & \text{with probab. } 1-y_i \end{cases}$$

Compute the probability of "failure" & Expected value of objective function

Most Binary Optimization Problems are very hard to solve.
We use different ideas to get approximate / nonoptimal solutions

① Heuristics based on greedy algorithms / local search.

② Randomized Rounding schemes.

Consider
$$\min \vec{c}^T \vec{x} \quad \text{s.t. } A\vec{x} \le \vec{b}, \quad x_1, x_2, \dots, x_n \in \{0,1\}$$

$\xrightarrow{\text{LP relaxation}}$

$$\min \vec{c}^T \vec{y} \quad \text{s.t. } A\vec{y} \le \vec{b}, \quad y_1, \dots, y_n \in [0,1]$$

LP, easy to solve for $y_1, \dots, y_n \in [0,1]$

"randomly round" each $y_i \in [0,1]$ to $x_i \in \{0,1\}$

Flip a coin with $\mathbb{P}[\text{heads}] = y_i$, $\mathbb{P}[\text{tails}] = 1 - y_i$

$$x_i = \begin{cases} 1 & \text{with probab. } y_i \\ 0 & \text{with probab. } 1 - y_i \end{cases}$$

Compute the probability of "failure" & Expected value of objective function

Analyze this to guarantee a good / close to optimal solution with high probability

(Monte Carlo) Randomized Algorithm:

Assigning jobs to qualified applicants / processors / machines

Let $a_1, a_2, \ldots, a_n$ be jobs to be carried out.

Let $b_1, b_2, \ldots, b_m$ be a pool of available applicants / processors / machines...

Each b_j is capable of carrying out only certain jobs a_i.

We have to assign each job to one applicant qualified to do it.

How can we maximize the number of assigned jobs?

Assigning jobs to qualified applicants / processors / machines

Let $a_1, a_2, \dots, a_n$ be jobs to be carried out.
Let $b_1, b_2, \dots, b_m$ be a pool of available applicants / processors / machines...
Each b_j is capable of carrying out only certain jobs a_i.
We have to assign each job to one applicant qualified to do it.
How can we maximize the number of assigned jobs?

Let $G=(V(G),E(G))$ be a graph that captures the given information as
$V(G) = V_a \cup V_b$ where $V_a = \{a_1, a_2, \dots, a_n\}$
$V_b = \{b_1, \dots, b_m\}$
Edges between any pair a_i and b_j if applicant b_j is qualified for job a_i.

Assigning jobs to qualified applicants / processors / machines

Let $a_1, a_2, \dots, a_n$ be jobs to be carried out.

Let $b_1, b_2, \dots, b_m$ be a pool of available applicants / processors / machines...

Each b_j is capable of carrying out only certain jobs a_i.

We have to assign each job to one applicant qualified to do it.

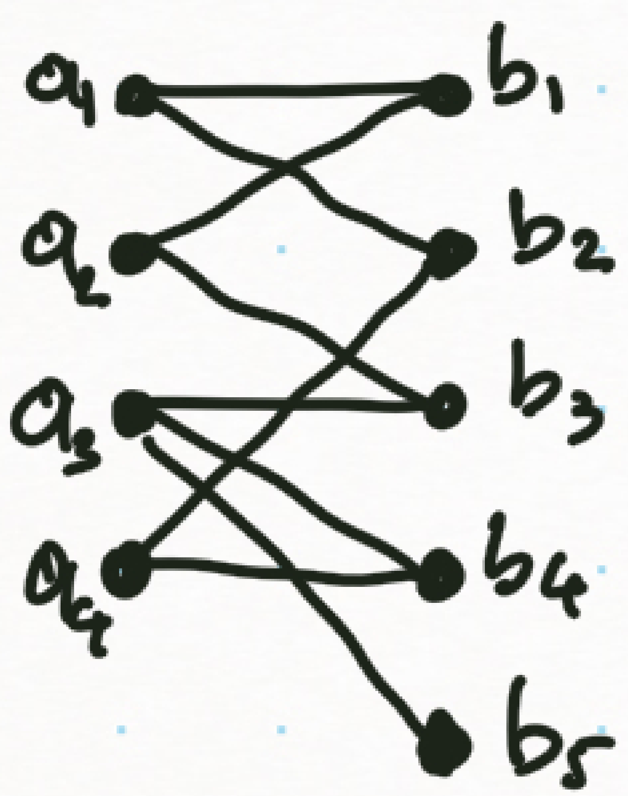
How can we maximize the number of assigned jobs?

Let $G = (V(G), E(G))$ be a graph that captures the given information as

$V(G) = V_a \cup V_b$ where $V_a = \{a_1, a_2, \dots, a_n\}$

$V_b = \{b_1, \dots, b_m\}$

Edges between any pair a_i and b_j if applicant b_j is qualified for job a_i.



A graph whose vertices can be partitioned into two disjoint subsets such that all edges only go inbetween the two parts, is called a bipartite graph.

Assigning jobs to qualified applicants / processors / machines

Let $a_1, a_2, \dots, a_n$ be jobs to be carried out.

Let $b_1, b_2, \dots, b_m$ be a pool of available applicants / processors / machines...

Each b_j is capable of carrying out only certain jobs a_i.

We have to assign each job to one applicant qualified to do it.

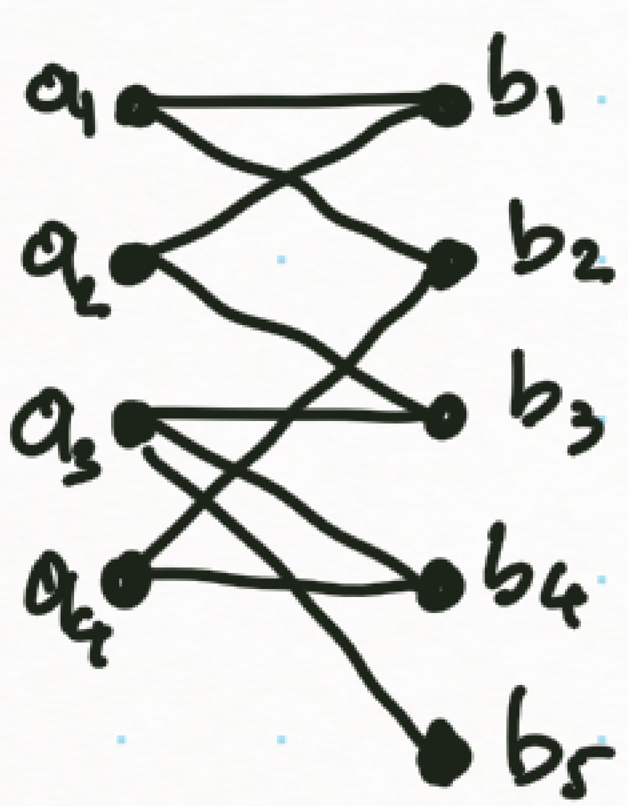
How can we maximize the number of assigned jobs?

Let $G=(V(G),E(G))$ be a graph that captures the given information as

$V(G) = V_a \cup V_b$ where $V_a=\{a_1,a_2,\dots,a_n\}$

$V_b=\{b_1,\dots,b_m\}$

Edges between any pair a_i and b_j if applicant b_j is qualified for job a_i.



Assigning jobs to qualified applicants / processors / machines

Let $a_1, a_2, \dots, a_n$ be jobs to be carried out.

Let $b_1, b_2, \dots, b_m$ be a pool of available applicants / processors / machines...

Each b_j is capable of carrying out only certain jobs a_i.

We have to assign each job to one applicant qualified to do it.

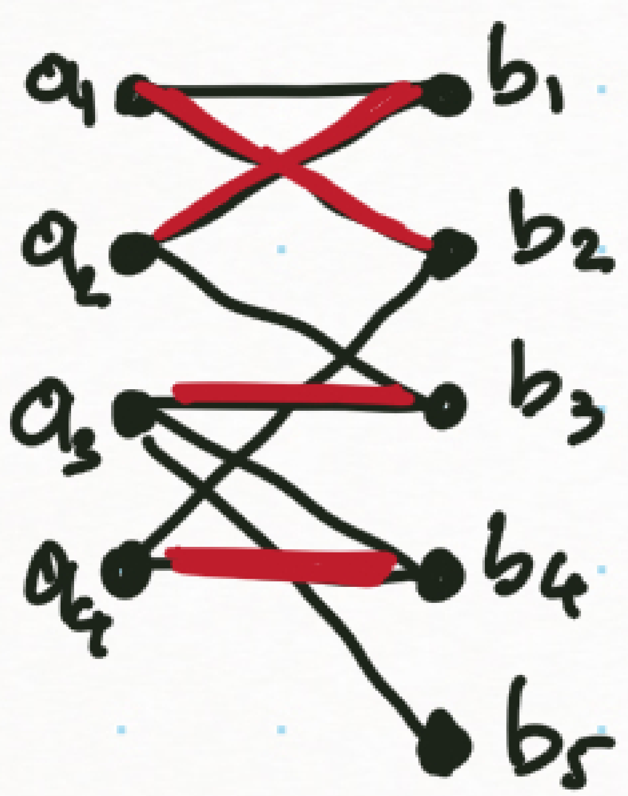
How can we maximize the number of assigned jobs?

Let $G = (V(G), E(G))$ be a graph that captures the given information as

$V(G) = V_a \cup V_b$ where $V_a = \{a_1, a_2, \dots, a_n\}$

$V_b = \{b_1, \dots, b_m\}$

Edges between any pair a_i and b_j if applicant b_j is qualified for job a_i.



a_1 assigned to b_2

a_2 —— " —— b_1

a_3 —— " —— b_3

a_4 —— " —— b_4

A matching is subset of $E(G)$ such that all edges in it are vertex disjoint.

Maximum Matching: $\max_{M \text{ matching}} |M|$

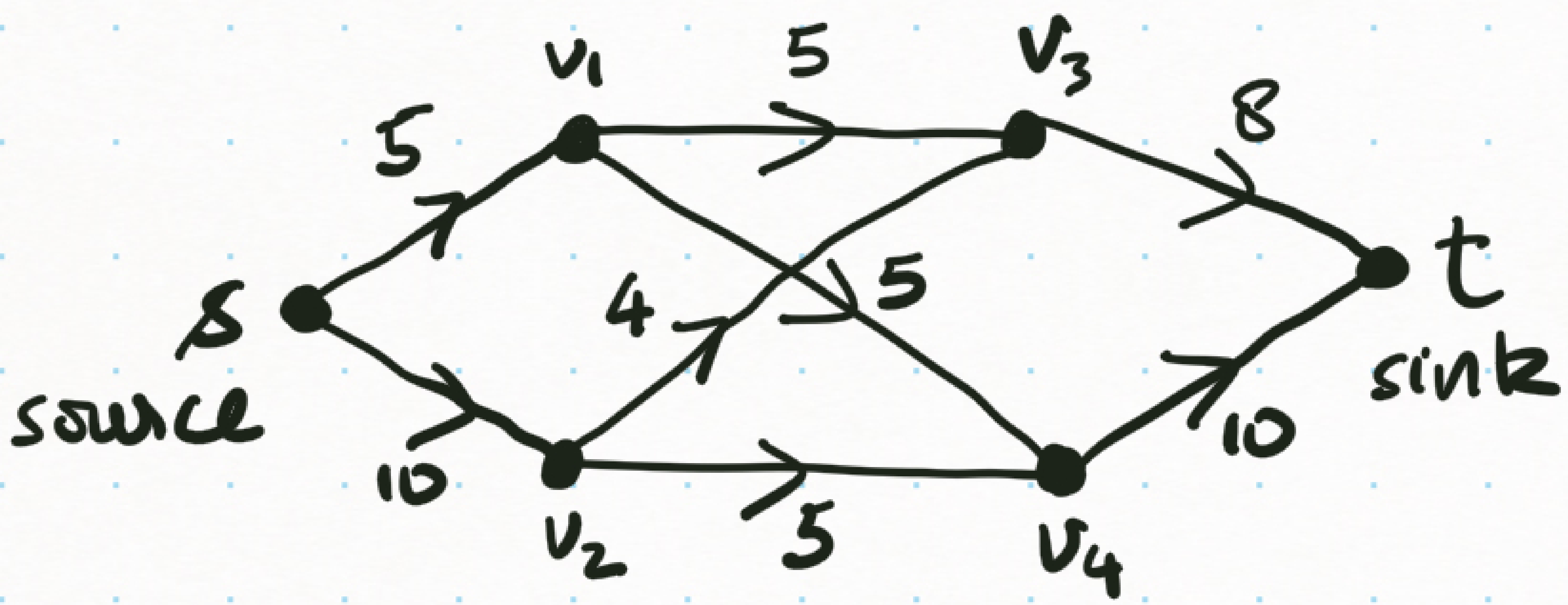
Maximum Matching problem in a bipartite graph can be modeled as a more general network optimization problem called the

Maximum Flow Problem

Given a network $G = (N, A)$ (nodes: N, arcs: A)

two special nodes $s \equiv$ source & $t =$ sink

capacity on each arc, $u_{ij} \geq 0$ for each arc $(v_i, v_j) \in A$



source $\equiv$ factory
sink $\equiv$ retail store
network $\equiv$ transportation netwk
capacity $\equiv$ how much can be transported using that link
$v_1, v_2, v_3, v_4 \equiv$ warehouses/ transhipment nodes.

Aim maximize the amount of flow from s to t while respecting the capacities

Flow $\equiv$ quantity of goods transported across each arc.

Max Flow Problem

Decision variables $\quad x_{ij}$ = amount of flow from v_i to v_j for each arc (v_i, v_j)

$$\max \sum_{i \in N^+(s)} x_{si}$$

such that

$$\sum_{v_k \in N^-(v_i)} x_{ki} - \sum_{v_j \in N^+(v_i)} x_{ij} = 0 \quad \forall v_i \in N$$

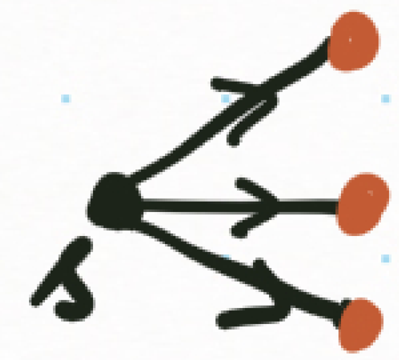
$$x_{ij} \leq u_{ij} \quad \forall (v_i, v_j) \in A$$

$$x_{ij} \geq 0 \quad \forall (v_i, v_j) \in A$$

Max Flow Problem

Decision variables x_{ij} = amount of flow from v_i to v_j for each arc (v_i, v_j)

$$\max \sum_{i \in N^+(s)} x_{si}$$



[total flow out of node s]

$N^+(s)$
out-neighborhood of s

such that

$$\sum_{v_k \in N^-(v_i)} x_{ki} - \sum_{v_j \in N^+(v_i)} x_{ij} = 0 \quad \forall v_i \in N$$

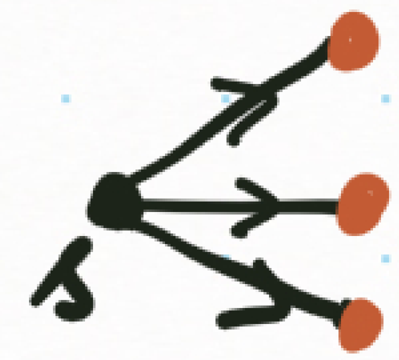
$$x_{ij} \leq u_{ij} \quad \forall (v_i, v_j) \in A$$

$$x_{ij} \geq 0 \quad \forall (v_i, v_j) \in A$$

Max Flow Problem

Decision variables x_{ij} = amount of flow from v_i to v_j for each arc (v_i, v_j)

$$\max \sum_{i \in N^+(s)} x_{si}$$



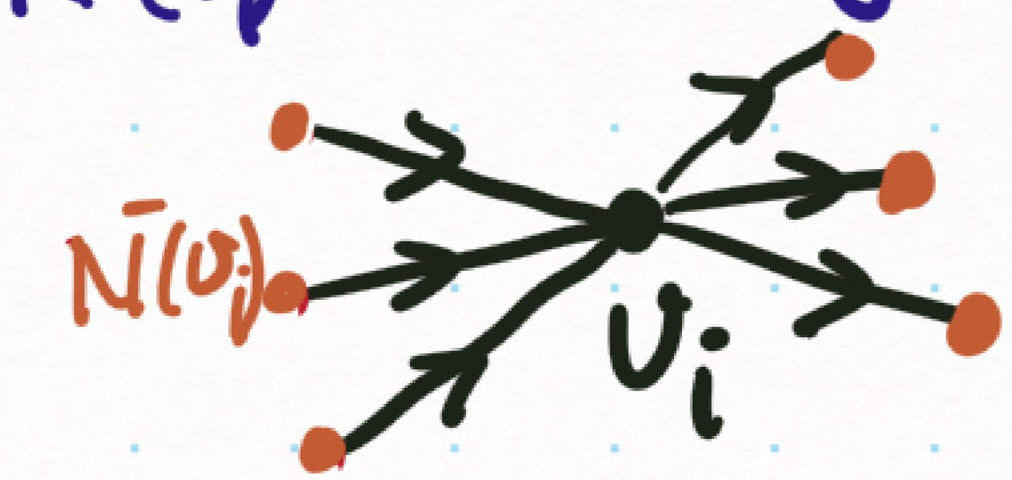
[total flow out of node s]

$N^+(s)$
out-neighborhood of s

such that

$$\sum_{v_k \in N^-(v_i)} x_{ki} - \sum_{v_j \in N^+(v_i)} x_{ij} = 0 \quad \forall v_i \in N$$

[In-flow = Out-flow at each node v_i]



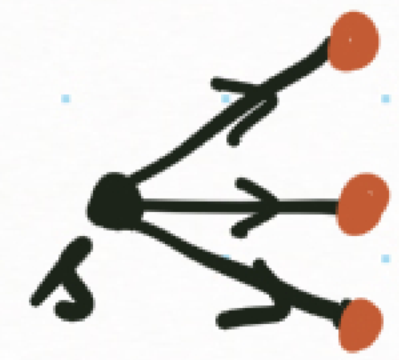
$$x_{ij} \leq u_{ij} \quad \forall (v_i, v_j) \in A$$

$$x_{ij} \geq 0 \quad \forall (v_i, v_j) \in A$$

Max Flow Problem

Decision variables x_{ij} = amount of flow from v_i to v_j for each arc (v_i, v_j)

$$\max \sum_{i \in N^+(s)} x_{si}$$



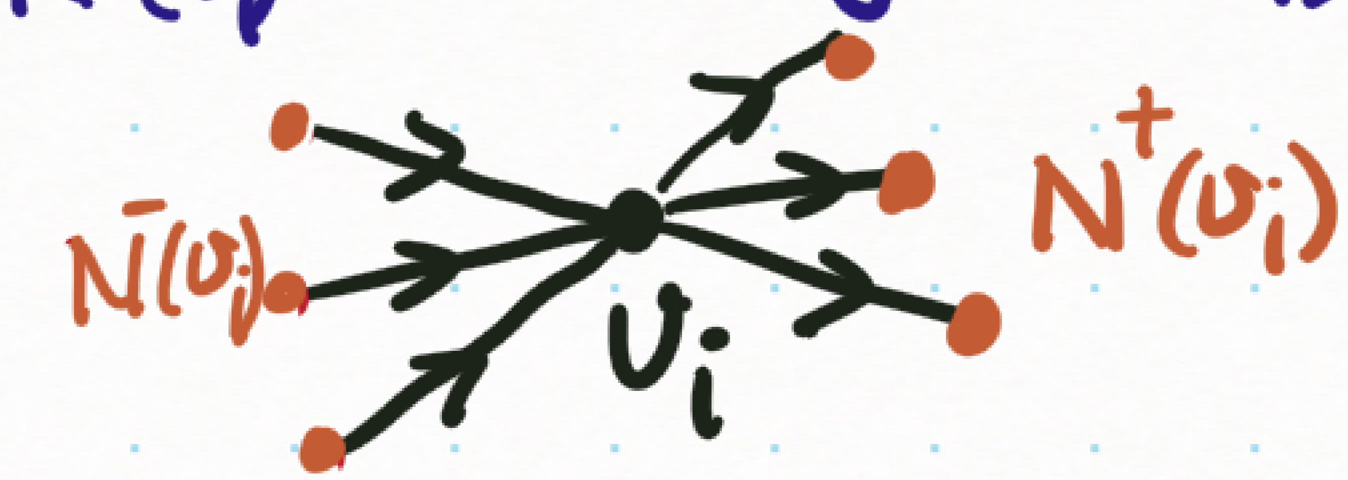
[total flow out of node s]

$N^+(s)$ out-neighborhood of s

such that

$$\sum_{v_k \in N^-(v_i)} x_{ki} - \sum_{v_j \in N^+(v_i)} x_{ij} = 0 \quad \forall v_i \in N$$

[In-flow = Out-flow at each node v_i]



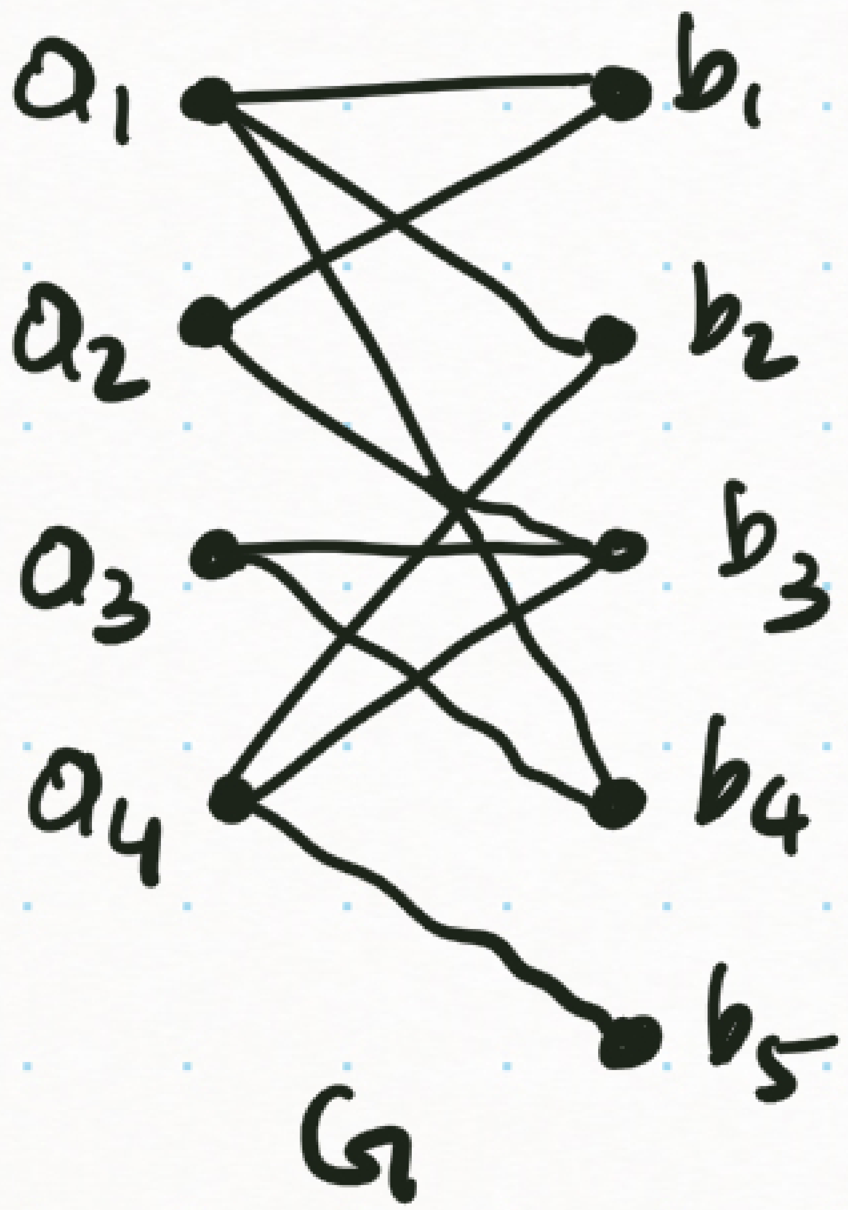
$$x_{ij} \leq u_{ij} \quad \forall (v_i, v_j) \in A$$

[capacity on each arc $v_i v_j$]

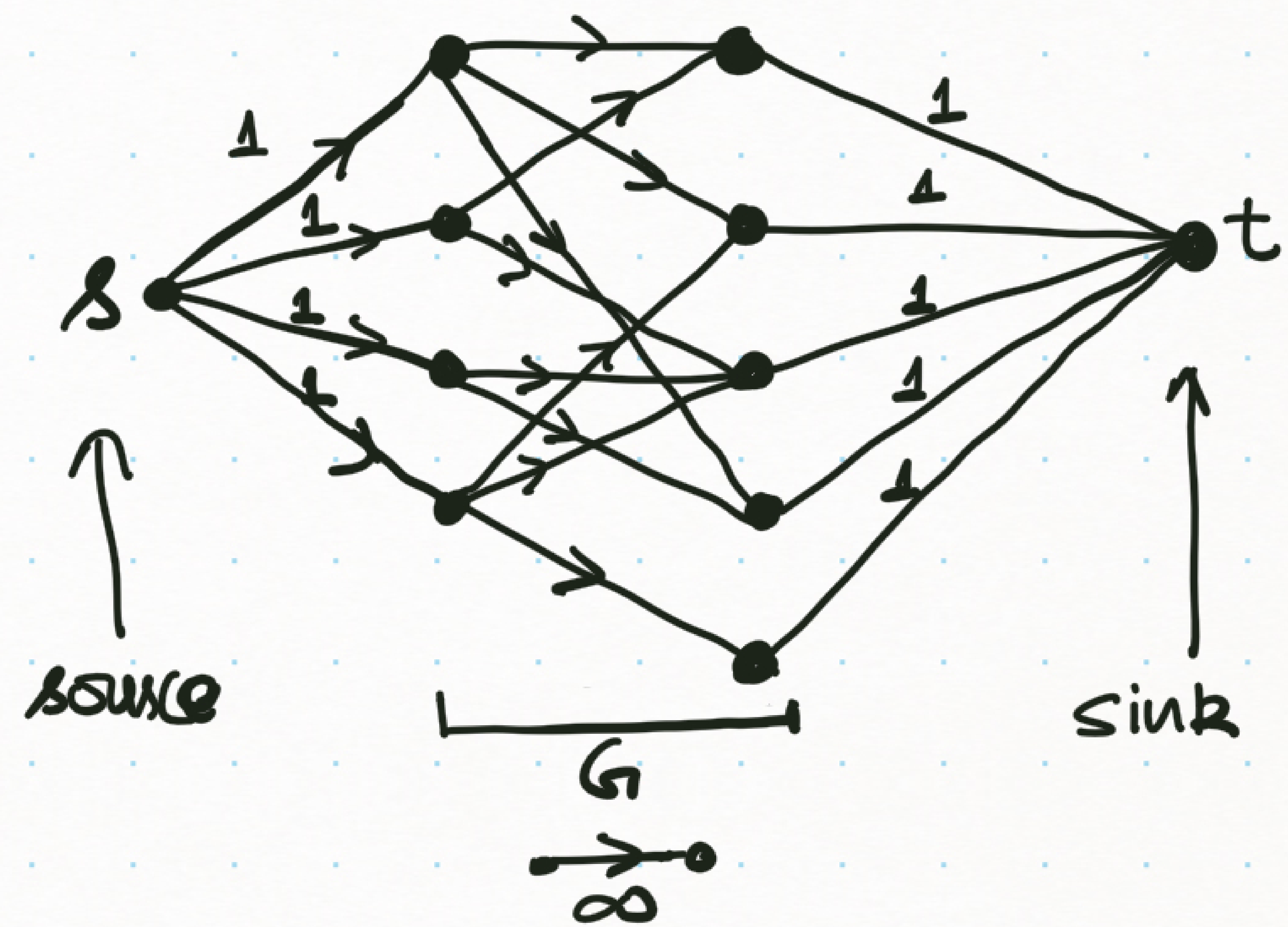
$$x_{ij} \geq 0 \quad \forall (v_i, v_j) \in A$$

[non-negativity]

Finding max matching on



is the same as finding the max flow on



Modeling Logistics problem on a transportation network

We have a transportation network connecting factories, warehouses and retail stores. Each factory can supply certain amount of goods and Each store has a demand for a certain amount of goods. How can we satisfy demand while minimizing transportation costs?

Modeling Logistics problem on a Transportation network

We have a transportation network connecting factories, warehouses and retail stores. Each factory can supply certain amount of goods and Each store has a demand for a certain amount of goods. How can we satisfy demand while minimizing transportation costs?

Given a network $G=(N,A)$

for each arc (v_i, v_j), we are given capacity bounds $l_{ij} \le x_{ij} \le u_{ij}$

and there is cost per unit amount of transportation on that arc c_{ij}

for each node v_i, There is supply/demand $b(i)$ associated with it [$b(i) > 0$ means supply from factory

$b(i) < 0$ means demand from store

$b(i) = 0$ means transhipment point]

Decision variables x_{ij} = flow over the arc (v_i, v_j) for all arcs $(v_i, v_j) \in A$. $\quad v_i \longrightarrow v_j$

$$\min \sum_{(v_i,v_j)\in A} c_{ij} x_{ij}$$

such that

$$x_{ij} \geq l_{ij} \quad \text{for all } (v_i,v_j)\in A$$

$$x_{ij} \leq u_{ij} \quad \text{for all } (v_i,v_j)\in A$$

$$\sum_{v_j \in N^+(v_i)} x_{ij} - \sum_{v_k \in N^-(v_i)} x_{ki} = b(i) \quad \text{for all } v_i \in N$$

Transportation Problem

Decision variables x_{ij} = flow over the arc (v_i, v_j) for all arcs $(v_i, v_j) \in A$. $v_i \longrightarrow v_j$

$$\min \sum_{(v_i,v_j)\in A} c_{ij} x_{ij}$$ [Total cost of flow]

such that

$$x_{ij} \geq l_{ij} \quad \text{for all } (v_i,v_j)\in A$$

$$x_{ij} \leq u_{ij} \quad \text{for all } (v_i,v_j)\in A$$

[capacity bound on flow over each arc]

$$\sum_{v_j \in N^+(v_i)} x_{ij} - \sum_{v_k \in N^-(v_i)} x_{ki} = b(i) \quad \text{for all } v_i \in N$$

Transportation Problem

Decision variables x_{ij} = flow over the arc (v_i, v_j) for all arcs $(v_i, v_j) \in A$. $\quad v_i \longrightarrow v_j$

$$\min \sum_{(v_i,v_j)\in A} c_{ij} x_{ij}$$

[Total cost of flow]

such that

$$x_{ij} \geq l_{ij} \quad \text{for all } (v_i,v_j)\in A$$

$$x_{ij} \leq u_{ij} \quad \text{for all } (v_i,v_j)\in A$$

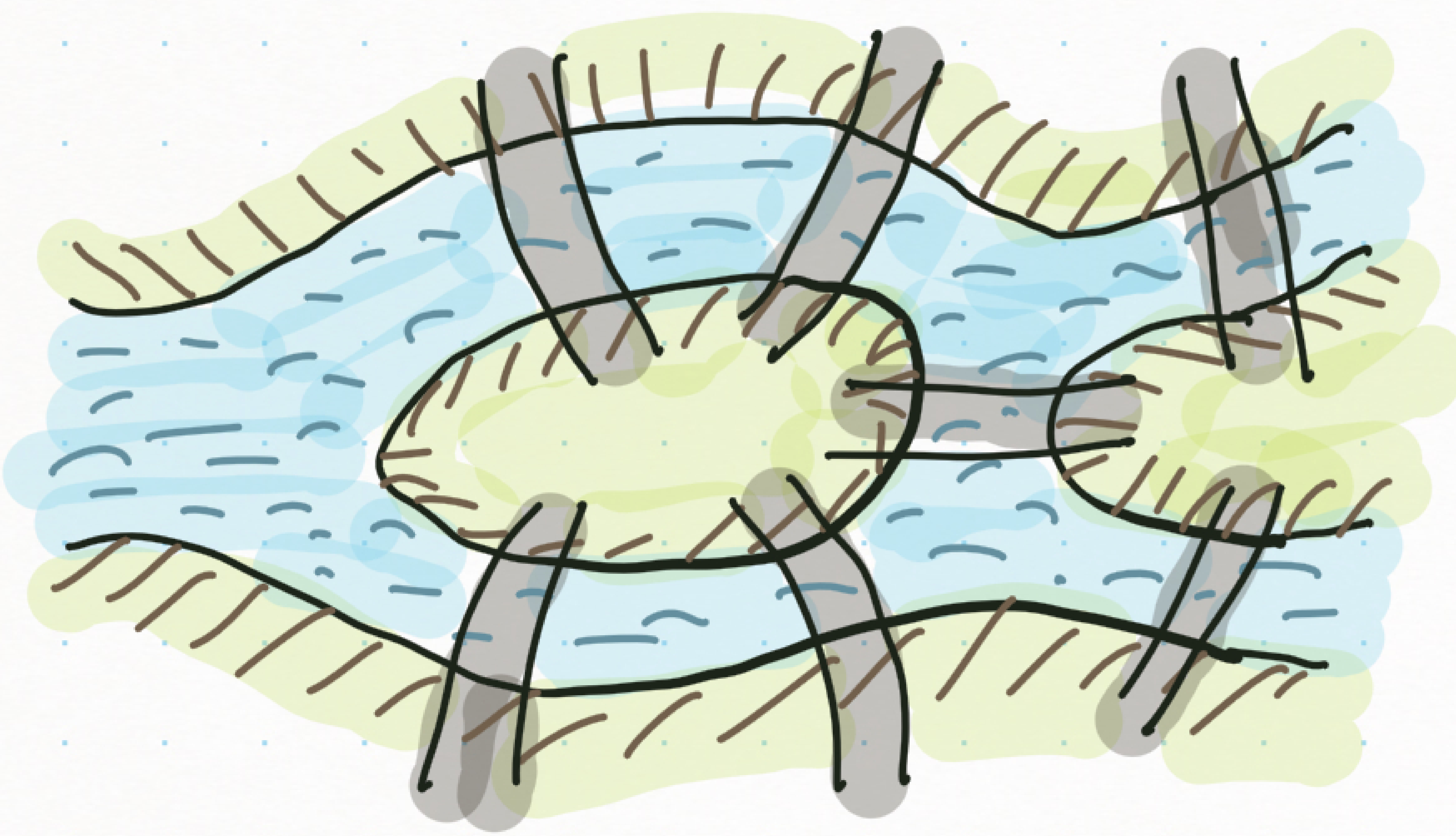
[capacity bound on flow over each arc]

$$\sum_{v_j \in N^+(v_i)} x_{ij} - \sum_{v_k \in N^-(v_i)} x_{ki} = b(i) \quad \text{for all } v_i \in N$$

[Outflow − Inflow = "supply" at node v_i]

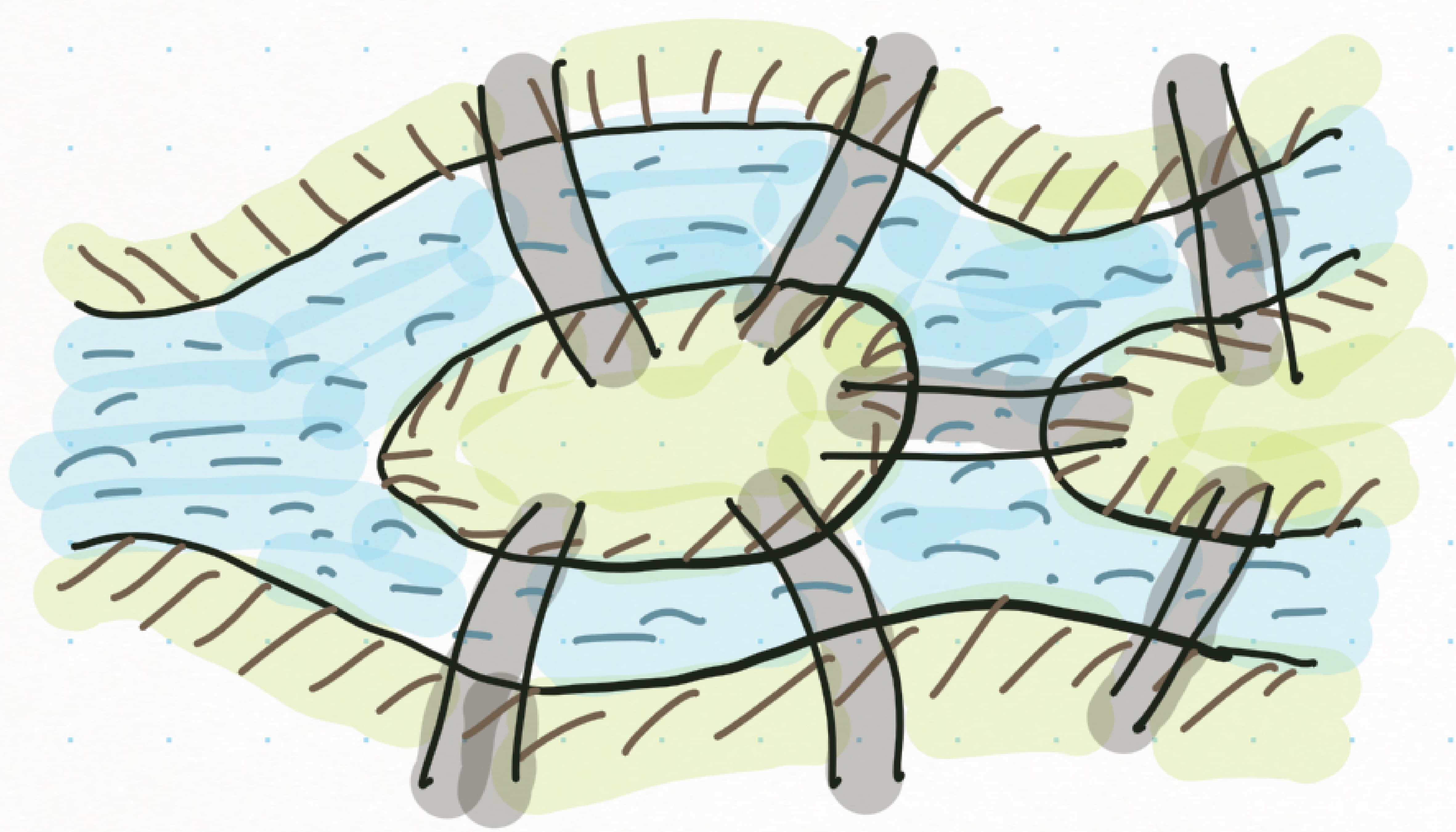
Transportation Problem

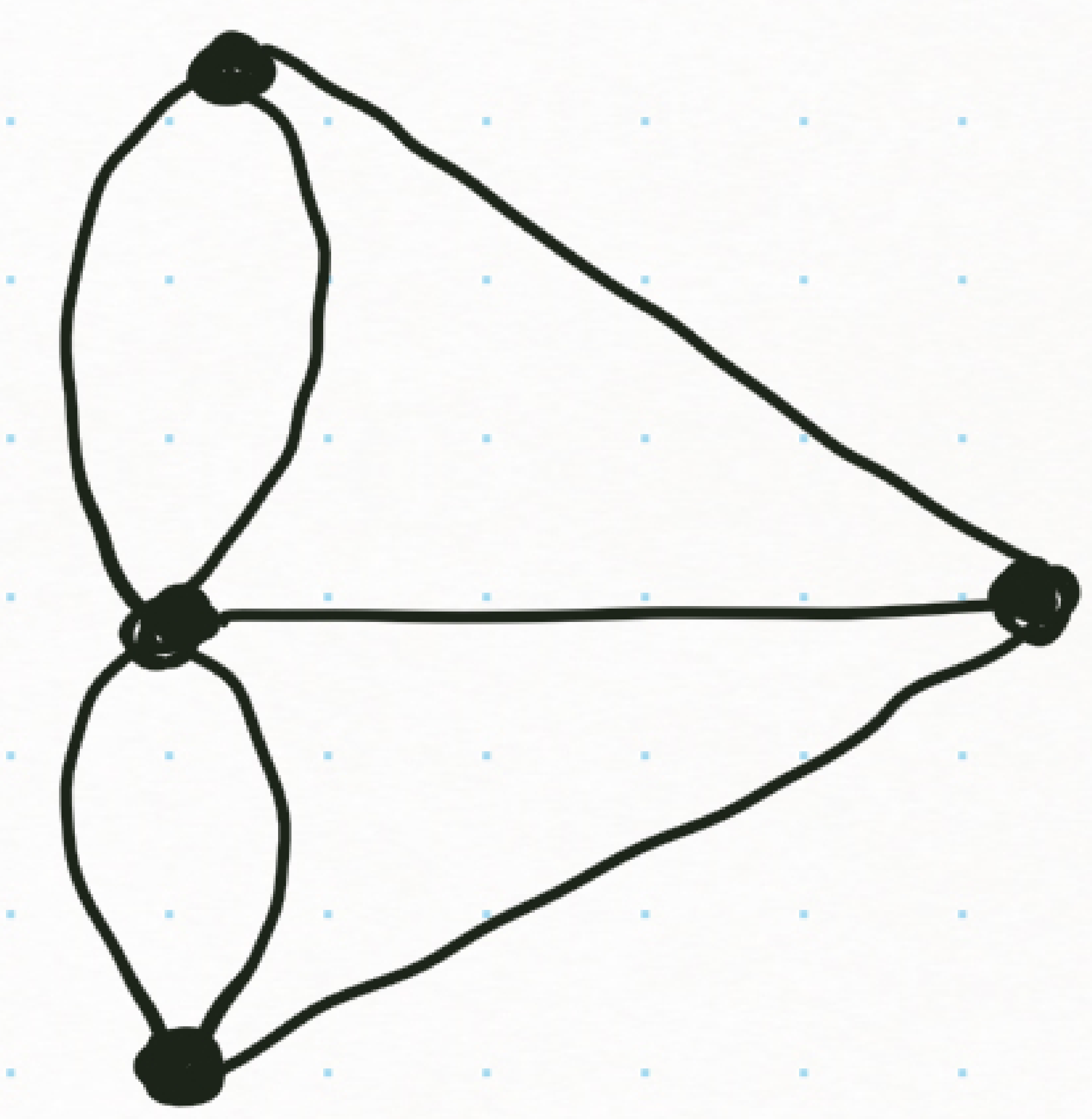
<u>Read</u> The Seven Bridges of Königsberg,
Euler's problem, and
how to solve this topological problem
using graph theory in section 8.1.



Start & end a walk at the same spot and cross each of the 7 bridges exactly once during the walk. ← Can this be done?

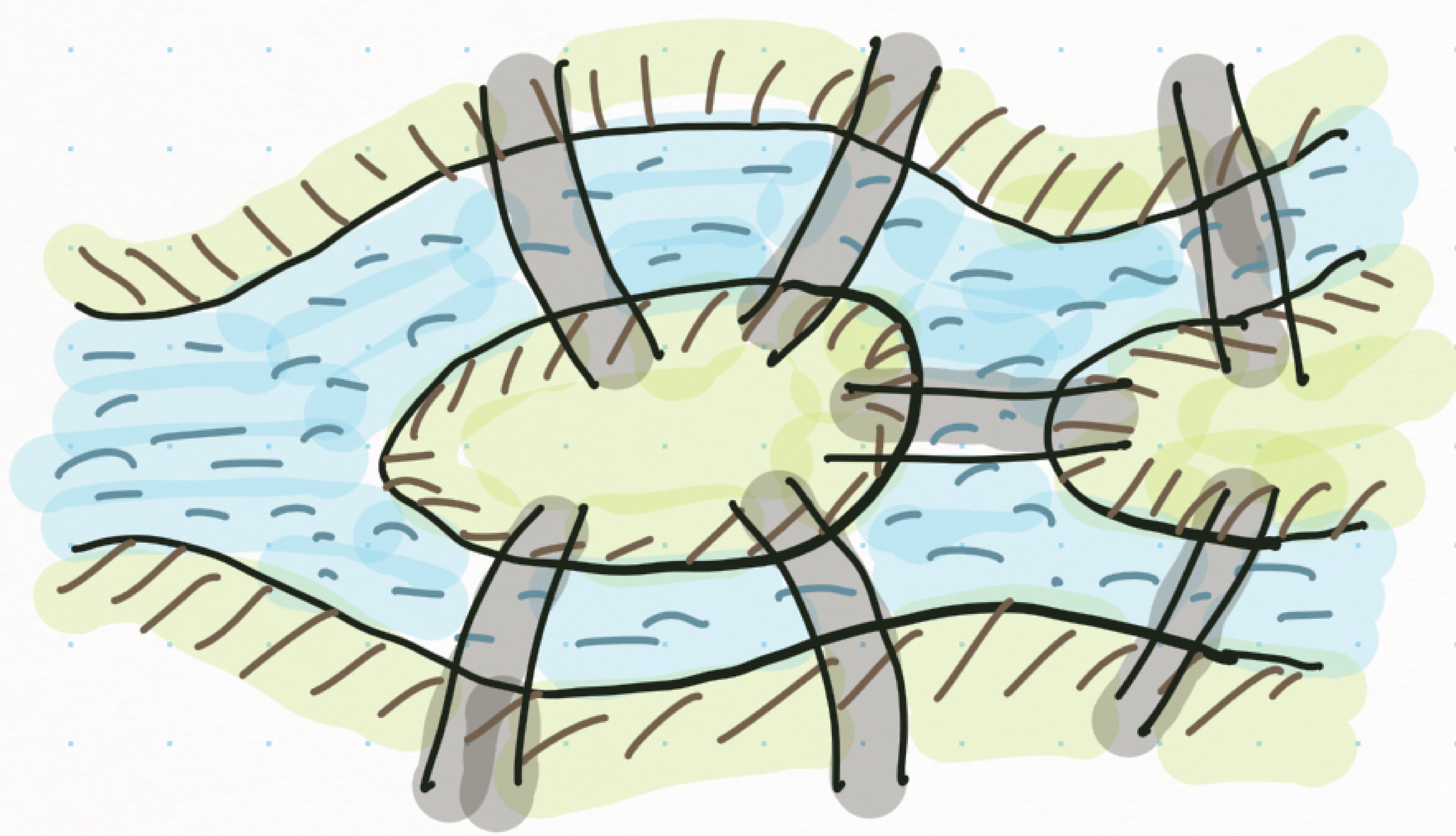
Read The Seven Bridges of Königsberg, Euler's problem, and how to solve this topological problem using graph theory in section 8.1.

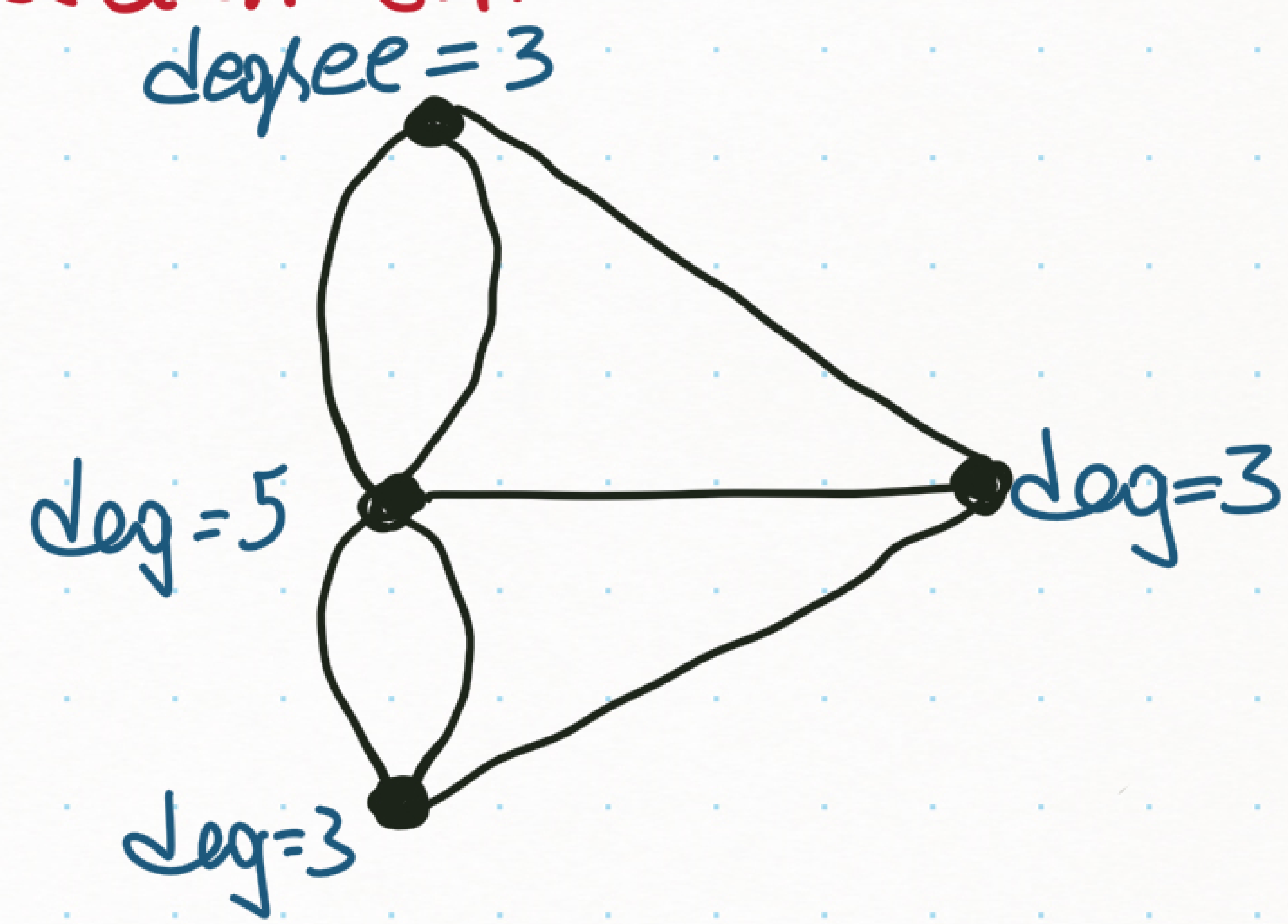




Start & end a walk at the same spot and cross each of the 7 bridges exactly once during the walk. ← Can this be done?

Read The Seven Bridges of Königsberg, Euler's problem, and how to solve this topological problem using graph theory in section 8.1.





Start & end a walk at the same spot and cross each of the 7 bridges exactly once during the walk. ← Can this be done?