ORIE 3310/5310 : Optimization II

Course Introduction

Prof. Oktay Günlük
Cornell University, School of ORIE

Spring 2024

Class Overview

- Instructor: Oktay Günlük (ong5@cornell.edu)
- Office Hours (Rhodes 221):
  - Wednesday 3-4 PM, or by appointment
- Class meetings:
  - Mon and Wed 1:25PM - 2:40PM, Gates Hall G01
  - Questions and corrections during class are very welcome.
- Labs start **this week**
  - Tu 10:10AM - 12:05PM, Rhodes Hall 571
  - Tu 2:30PM - 4:25PM, Rhodes Hall 453
  - We 10:10AM - 12:05PM, Rhodes Hall 571
- Weekly homework: Due on Tuesday 9pm
  - You may turn two homeworks late (9pm on Thursday)
  - Lowest homework grade will be dropped
  - Regrade requests via Gradescope (before next HW due)
- Textbook (not required): *Introduction to mathematical programming : operations research*, Wayne L. Winston.

Goal of the course

Primary course objectives:

- Introduce problems and algorithms beyond LP and simplex method.
- Study problems and algorithms, understand the importance of problem assumptions, and understand why these algorithms work.
- Learn how to apply these models and algorithms to actual problems.

Outline of topics:

- Integer programming: Formulation and modeling; Branch and bound, cutting planes, column generation.
- Network optimization: Minimum-cost network flow, maximum flow, shortest-path problem, assignment problem
- Dynamic programming: Bellman equation, principle of optimality, applications to various problems, stochastic dynamic programming

Prerequisites:

- Knowledge of linear algebra and ORIE 3300/5300 material, which include: formulating LPs, simplex method, duality.

Grading

Homework (weekly)	20%	(Due Tue. 9pm)
Prelim 1	20%	(February 22)
Prelim 2	20%	(April 16)
Final Exam	30%	(TBD, cumulative)
Lab Exercises	10%	
Participation	5%	
Total	105%	

- Homeworks:
  - No collaboration
  - Submit to Gradescope
- Labs:
  - Labs start this week (today): Python, Jupiter, intro to OR Tools
  - You may work with a partner and submit a single assignment
  - You must attend the lab session you have enrolled in unless approved by the head TA (Ishan)

TAs and course websites

- PhD TAs:
  - Ishan Bansal (ib332) **(head TA)**
  - Hannane Yaghoubizade (hy465)
  - Trang Tran (htt27)

    Office hours: (2 x 1 hour/week – TBA, check Canvas)
- MEng TA
  - Taojie Wang (tw533)

    (Office hours: 1 hour/week – TBA, check Canvas)
- Undergrad TAs
  - Andrew The (alt73), Jian Kai Ang (ja686) ,
  - Saniya Vaidya (ssv33), Francis Bahk (feb47)

    (Office hours: 1 hour/week – TBA, check Canvas)
- All office hours will take place at **Rhodes**, room TBA

Canvas: `https://canvas.cornell.edu/courses/59783`

Gradescope: `https://www.gradescope.com/courses/715386`

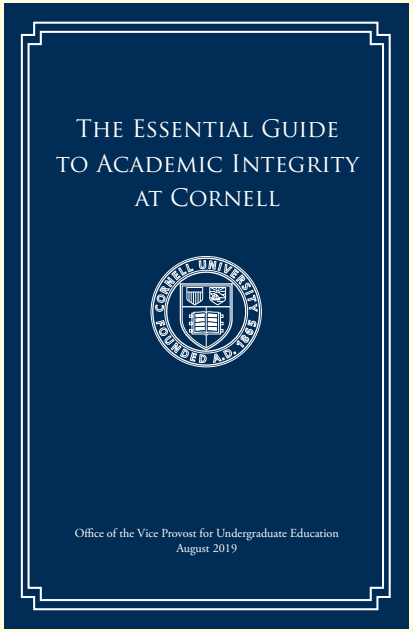
Ed Dis.: `https://edstem.org/us/courses/54464/discussion/`

Ground rules

- Your work on the written homework and exams should be your own.
- For the homework, you may discuss approaches to problems with other students,
  - This may **not** involve taking notes.
  - You must write up solutions on your own independently, and acknowledge anyone with whom you discussed the problem.
- For the lab exercises, you may work with a partner, and you may turn in a single assignment with your partner.
- You may bring a 1-sided **hand written** cheat sheet to the exams.

  (Must write your name on it)
- Buying, selling, or reposting course materials on external websites will be considered an academic integrity violation.
- SDS students needing exam accommodations must notify me and the head TA (Ishan) within three weeks of the start of the semester.
- Prelim dates are set by Cornell, you should plan accordingly.

Academic Integrity



Absolute integrity is expected of every Cornell student in all academic undertakings. Integrity entails a firm adherence to a set of values, and the values most essential to an academic community are grounded on the concept of honesty with respect to the intellectual efforts of oneself and others.

A Cornell student's submission of work for academic credit indicates that the work is the student's own. All outside assistance should be acknowledged, and the student's academic position truthfully reported at all times. In addition, Cornell students have a right to expect academic integrity from each of their peers.

Questions?

Course overview

What is this course about?

- Mathematical optimization in a nutshell:

$$\min_{x \in X} \; f(x)$$

  - $X \subseteq \mathbb{R}^n$ is the set of feasible solutions
  - $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function

- Linear programming is about picking a point from a polyhedral set X

$$X = \left\{ x \in \mathbb{R}^n \;\text{ s.t. }\; Ax \leq b,\; x \geq 0 \right\}$$

  to minimize a linear function $f(x) = \sum_{i=1}^{n} c_i x_i$ for a given $c \in \mathbb{R}^n$.

- Integer Programming deals with sets: $X \subseteq \mathbb{Z}^n$
- Mixed-Integer Programming deals with sets: $X \subseteq \mathbb{Z}^k \times \mathbb{R}^{n-k}$
- Many real-world optimization problems can be framed as mathematical optimization problems.

What is this course about?

- Mathematical optimization in a nutshell:

$$\min_{x \in X} \; f(x)$$

  - $X \subseteq \mathbb{R}^n$ is the set of feasible solutions
  - $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function

- Linear programming is about picking a point from a polyhedral set X

$$X = \left\{ x \in \mathbb{R}^n \;\text{ s.t. }\; Ax \leq b,\; x \geq 0 \right\}$$

  to minimize a linear function $f(x) = \sum_{i=1}^{n} c_i x_i$ for a given $c \in \mathbb{R}^n$.

- Integer Programming deals with sets: $X \subseteq \mathbb{Z}^n$
- Mixed-Integer Programming deals with sets: $X \subseteq \mathbb{Z}^k \times \mathbb{R}^{n-k}$
- Many real-world optimization problems can be framed as mathematical optimization problems.

What is this course about?

- Mathematical optimization in a nutshell:

$$\min_{x \in X} \; f(x)$$

  - $X \subseteq \mathbb{R}^n$ is the set of feasible solutions
  - $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function

- Linear programming is about picking a point from a polyhedral set X

$$X = \left\{ x \in \mathbb{R}^n \;\text{ s.t. }\; Ax \leq b,\; x \geq 0 \right\}$$

  to minimize a linear function $f(x) = \sum_{i=1}^{n} c_i x_i$ for a given $c \in \mathbb{R}^n$.

- Integer Programming deals with sets: $X \subseteq \mathbb{Z}^n$
- Mixed-Integer Programming deals with sets: $X \subseteq \mathbb{Z}^k \times \mathbb{R}^{n-k}$
- Many real-world optimization problems can be framed as mathematical optimization problems.

What is this course about?

- Mathematical optimization in a nutshell:

$$\min_{x \in X} \; f(x)$$

  - $X \subseteq \mathbb{R}^n$ is the set of feasible solutions
  - $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function

- Linear programming is about picking a point from a polyhedral set X

$$X = \left\{ x \in \mathbb{R}^n \;\text{ s.t. }\; Ax \leq b,\; x \geq 0 \right\}$$

  to minimize a linear function $f(x) = \sum_{i=1}^{n} c_i x_i$ for a given $c \in \mathbb{R}^n$.

- Integer Programming deals with sets: $X \subseteq \mathbb{Z}^n$
- Mixed-Integer Programming deals with sets: $X \subseteq \mathbb{Z}^k \times \mathbb{R}^{n-k}$
- Many real-world optimization problems can be framed as mathematical optimization problems.

Beyond linear programming: Machine assignment

- Consider the following problem. You have :
  - A set of jobs: $J = \{1, \ldots, n_J\}$, and,
  - A set of machines: $M = \{1, \ldots, n_M\}$
  - Cost of processing job j on machine m is c_{jm}
  - Find a least cost assignment of the jobs to the machines.
- You cannot simply assign each job to the cheapest machine it can be processed on because machines have limited processing capacity:
  - Machine $m \in M$ has processing capacity b_m
  - Processing job $j \in J$ on machine $m \in M$ requires p_{jm} units of machine capacity.
- We will next formulate this problem as a mathematical optimization problem.

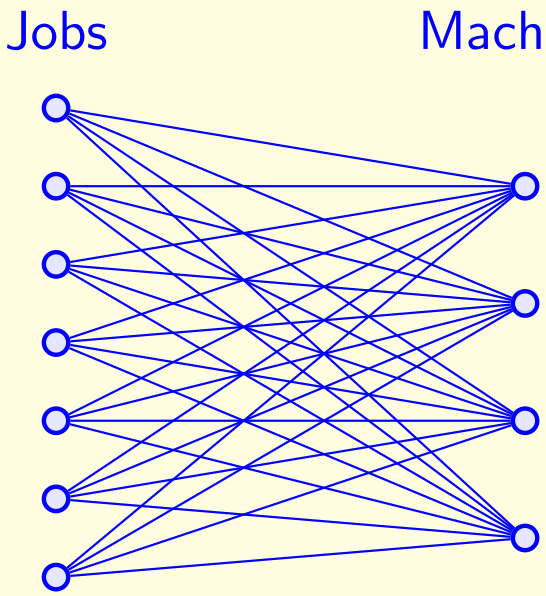
Beyond linear programming: Machine assignment

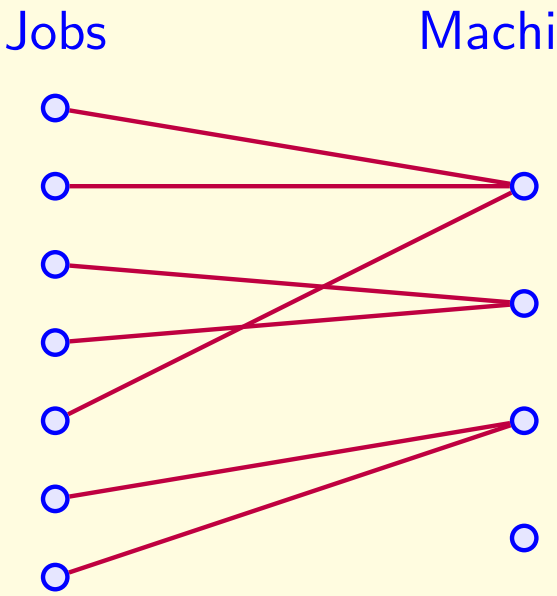
- Consider the following problem. You have :
  - A set of jobs: $J = \{1, \ldots, n_J\}$, and,
  - A set of machines: $M = \{1, \ldots, n_M\}$
  - Cost of processing job j on machine m is c_{jm}
  - Find a least cost assignment of the jobs to the machines.
- You cannot simply assign each job to the cheapest machine it can be processed on because machines have limited processing capacity:
  - Machine $m \in M$ has processing capacity b_m
  - Processing job $j \in J$ on machine $m \in M$ requires p_{jm} units of machine capacity.
- We will next formulate this problem as a mathematical optimization problem.

Example Problem: Machine assignment

- Find a least cost assignment of set of jobs J to machines M.
- Cost of assigning job $j \in J$ to machine $m \in M$ is c_{jm}
- Processing job j on machine m requires p_{jm} units of machine capacity.
- Machine $m \in M$ has capacity b_m. (assignment must respect capacities)
- How to assign jobs to machines? $\rightarrow$ decision variables: $x_{jm} \in \underbrace{\{0,1\}}_{0 \text{ or } 1}$

($x_{jm} = 1$ means that job j is assigned to machine m; 0, otherwise.)





Example Problem: Machine assignment

- Find a least cost assignment of set of jobs J to machines M.
- Cost of assigning job $j \in J$ to machine $m \in M$ is c_{jm}
- Processing job j on machine m requires p_{jm} units of machine capacity.
- Machine $m \in M$ has capacity b_m. (assignment must respect capacities)
- How to assign jobs to machines? $\rightarrow$ decision variables: $x_{jm} \in \underbrace{\{0,1\}}_{0 \text{ or } 1}$

($x_{jm} = 1$ means that job j is assigned to machine m; 0, otherwise.)

$$\min \sum_{j \in J} \sum_{m \in M} c_{jm} x_{jm} \quad \longleftarrow \quad \text{objective function}$$

$$\text{s.t.} \sum_{m \in M} x_{jm} = 1 \quad \forall j \in J \quad \text{assign each jobs once}$$

$$\sum_{j \in J} p_{jm} x_{jm} \leq b_m \quad \forall m \in M \quad \text{do not exceed capacity}$$

$$\underbrace{x_{jm} \in \{0,1\}}_{\text{not a linear constraint}} \quad \forall j \in J, m \in M \quad \text{variables can either be 0 or 1}$$

Example Problem: Machine assignment

- Find a least cost assignment of set of jobs J to machines M.
- Cost of assigning job $j \in J$ to machine $m \in M$ is c_{jm}
- Processing job j on machine m requires p_{jm} units of machine capacity.
- Machine $m \in M$ has capacity b_m. (assignment must respect capacities)
- How to assign jobs to machines? $\rightarrow$ decision variables: $x_{jm} \in \underbrace{\{0,1\}}_{0 \text{ or } 1}$

($x_{jm} = 1$ means that job j is assigned to machine m; 0, otherwise.)

$$\min \quad \sum_{j \in J} \sum_{m \in M} c_{jm} x_{jm} \qquad \longleftarrow \qquad \text{objective function}$$

$$\text{s.t.} \quad \sum_{m \in M} x_{jm} = 1 \qquad \forall j \in J \qquad \text{assign each jobs once}$$

$$\sum_{j \in J} p_{jm} x_{jm} \leq b_m \qquad \forall m \in M \qquad \text{do not exceed capacity}$$

$$\underbrace{x_{jm} \in \{0,1\}}_{\text{not a linear constraint}} \qquad \forall j \in J, m \in M \qquad \text{variables can either be 0 or 1}$$

Example Problem: Machine assignment

- Find a least cost assignment of set of jobs J to machines M.
- Cost of assigning job $j \in J$ to machine $m \in M$ is c_{jm}
- Processing job j on machine m requires p_{jm} units of machine capacity.
- Machine $m \in M$ has capacity b_m. (assignment must respect capacities)
- How to assign jobs to machines? $\rightarrow$ decision variables: $x_{jm} \in \underbrace{\{0,1\}}_{0 \text{ or } 1}$

($x_{jm} = 1$ means that job j is assigned to machine m; 0, otherwise.)

$$\min \quad \sum_{j \in J} \sum_{m \in M} c_{jm} x_{jm} \qquad \longleftarrow \qquad \text{objective function}$$

$$\text{s.t.} \quad \sum_{m \in M} x_{jm} = 1 \qquad \forall j \in J \qquad \text{assign each jobs once}$$

$$\sum_{j \in J} p_{jm} x_{jm} \leq b_m \qquad \forall m \in M \qquad \text{do not exceed capacity}$$

$$\underbrace{x_{jm} \in \{0,1\}}_{\text{not a linear constraint}} \qquad \forall j \in J, m \in M \qquad \text{variables can either be 0 or 1}$$

A Simpler Example: The Knapsack Problem

- You are choosing what to bring in your backpack.
- You can carry at most b pounds.
- You have n possible items.
- Item $i \in I = \{1, \ldots, n\}$ would give benefit of value c_i and has weight a_i.
- Which items to pack in your backpack for the trip?

Using variables $x_i \in \{0, 1\}$ to denote if item $i \in I$ is packed, we can formulate a simple integer program:

$$\max \sum_{i=1}^{n} c_i x_i$$

$$\text{s.t.} \sum_{i=1}^{n} a_i x_i \leq b, \quad x_i \in \{0, 1\} \text{ for all } i \in I$$

A Simpler Example: The Knapsack Problem

- You are choosing what to bring in your backpack.
- You can carry at most b pounds.
- You have n possible items.
- Item $i \in I = \{1, \ldots, n\}$ would give benefit of value c_i and has weight a_i.
- Which items to pack in your backpack for the trip?

Using variables $x_i \in \{0, 1\}$ to denote if item $i \in I$ is packed, we can formulate a simple integer program:

$$\max \quad \sum_{i=1}^{n} c_i x_i$$

$$\text{s. t.} \quad \sum_{i=1}^{n} a_i x_i \leq b, \quad x_i \in \{0, 1\} \text{ for all } i \in I$$

One might think that integer programming is not hard

- In the Knapsack Problem, what we want is to :
  - Choose a subset $S \subseteq I$ of possible items $I = \{1, \ldots, n\}$
  - Make sure they fit: $\sum_{i \in S} a_i \leq b$
  - Maximize: reward $= \sum_{i \in S} c_i$
- One simple way to solve this: Enumerate all possible subsets of I, consider the ones that weigh at most b, pick the best among them.
- But how long will it take using the fastest supercomputer:

n	Solutions to check	Time
3	8	0
10	1024	0
50	2^{50}	2 sec
110		

One might think that integer programming is not hard

- In the Knapsack Problem, what we want is to :
  - Choose a subset $S \subseteq I$ of possible items $I = \{1, \ldots, n\}$
  - Make sure they fit: $\sum_{i \in S} a_i \leq b$
  - Maximize: reward $= \sum_{i \in S} c_i$
- One simple way to solve this: Enumerate all possible subsets of I, consider the ones that weigh at most b, pick the best among them.
- But how long will it take using the fastest supercomputer:

n	Solutions to check	Time
3	8	0
10	1024	0
50	2^{50}	2 sec
110		

One might think that integer programming is not hard

- In the Knapsack Problem, what we want is to :
  - Choose a subset $S \subseteq I$ of possible items $I = \{1, \ldots, n\}$
  - Make sure they fit: $\sum_{i \in S} a_i \leq b$
  - Maximize: reward $= \sum_{i \in S} c_i$
- One simple way to solve this: Enumerate all possible subsets of I, consider the ones that weigh at most b, pick the best among them.
- But how long will it take using the fastest supercomputer:

n	Solutions to check	Time
3	8	0
10	1024	0
50	2^{50}	2 sec
110	2^{110}	Take a guess

One might think that integer programming is not hard

- In the Knapsack Problem, what we want is to :
  - Choose a subset $S \subseteq I$ of possible items $I = \{1, \ldots, n\}$
  - Make sure they fit: $\sum_{i \in S} a_i \leq b$
  - Maximize: reward $= \sum_{i \in S} c_i$
- One simple way to solve this: Enumerate all possible subsets of I, consider the ones that weigh at most b, pick the best among them.
- But how long will it take using the fastest supercomputer:

n	Solutions to check	Time
3	8	0
10	1024	0
50	2^{50}	2 sec
110	2^{110}	69 billion years*

*That's four times the age of the universe as we know it!

(remember, I has 2^n distinct subsets.)

A different kind of problem: Maximum Flows

- We want to send as many trucks as possible in a street network from one location s to another location t.
- For each street segment e, there is an upper bound u_e on the number of trucks that are allowed to use e.

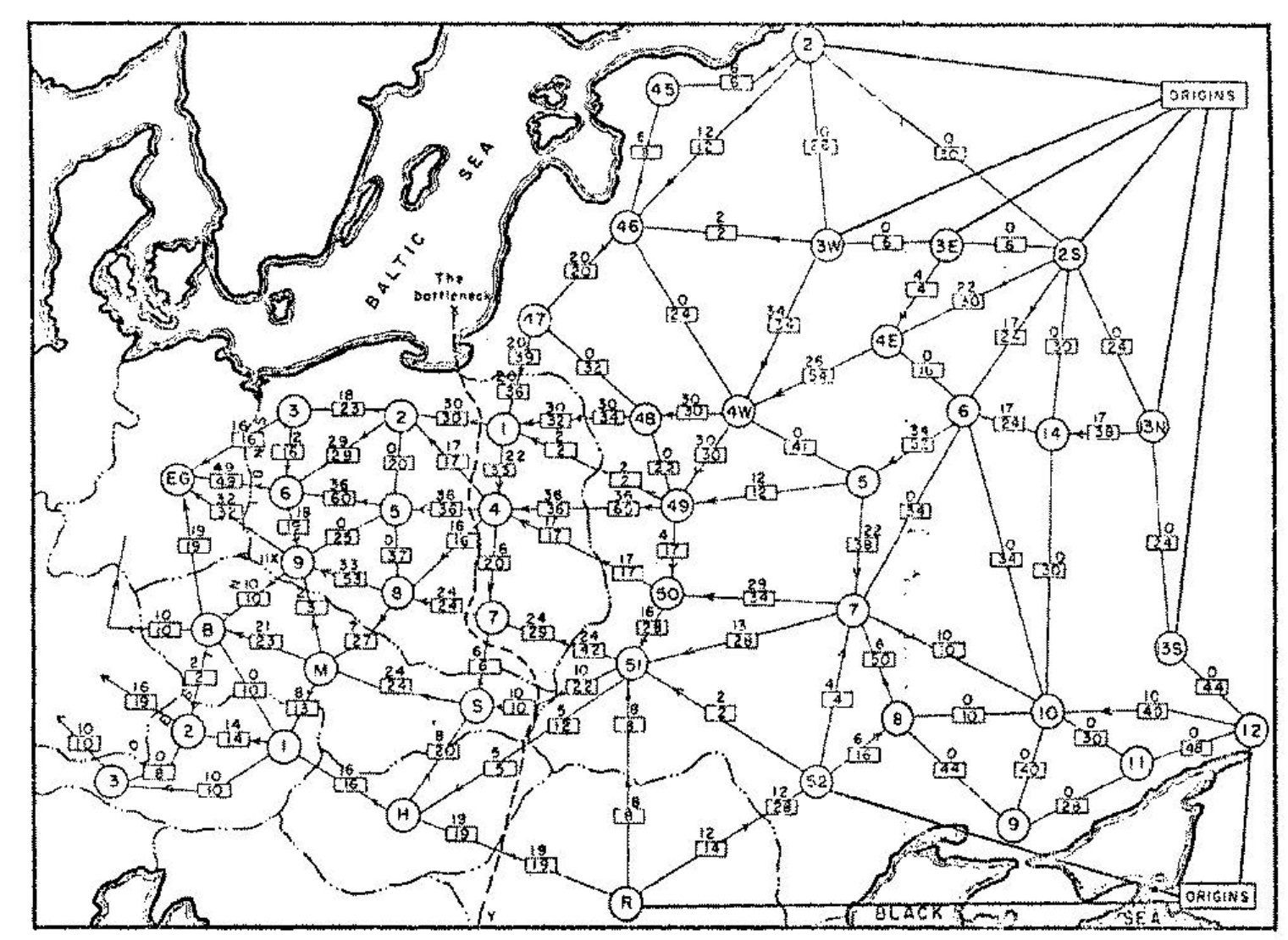
- The original motivation for studying this problem comes from the railway system of the Soviet Union (USSR) during the cold war.
- In a classified report Harris and Ross (1955) study the railway network in the Western Soviet Union and Eastern Europe
- The motivation was to understand (and disrupt) shipment of arms from the Soviet Union to Western Europe during a military conflict.

A different kind of problem: Maximum Flows

- We want to send as many trucks as possible in a street network from one location s to another location t.
- For each street segment e, there is an upper bound u_e on the number of trucks that are allowed to use e.

- The original motivation for studying this problem comes from the railway system of the Soviet Union (USSR) during the cold war.
- In a classified report Harris and Ross (1955) study the railway network in the Western Soviet Union and Eastern Europe
- The motivation was to understand (and disrupt) shipment of arms from the Soviet Union to Western Europe during a military conflict.

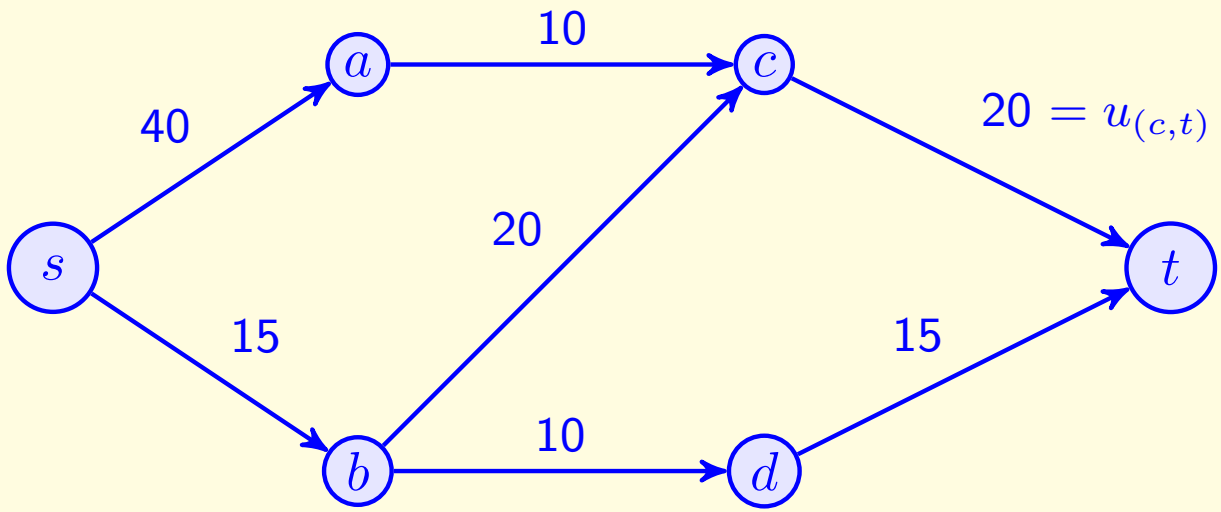
Soviet railroad network in 1955



Maximum s-t flow problem:

- **Given:** Directed graph $G = (V, A)$,
  a source node $s \in V$ and a sink node $t \in V$,
  arc capacities u_a for $a \in A$
- **Find:** Maximum flow that can be sent from s to t.

Example:



- This problem can be formulated as an LP but we will study an algorithm to solve this without using LP machinery.

Questions?