

Welcome to  
**Machine Learning for  
Discrete Optimization!**

# About me



## **Ellen Vitercik**

Assistant Professor at Stanford

*Management Science & Engineering*

*Computer Science*

Research revolves around

- Machine learning for discrete optimization
- Interface between economics and computation

# About me



Grew up in Lincoln, Vermont



BA: Columbia  
*Math*



PhD: Carnegie Mellon  
*Computer Science*



Postdoc: UC Berkeley

# Plan for today

1. Introduction
2. Course logistics
3. Overview of course topics

# How to integrate **machine learning** into **discrete optimization**?



## **Algorithm configuration**

*How to tune an algorithm's parameters?*



## **Algorithm selection**

*Given a variety of algorithms, which to use?*



## **Algorithm design**

*Can machine learning guide algorithm discovery?*

# How to integrate **machine learning** into **discrete optimization**?



## **Algorithm configuration**

*How to tune an algorithm's parameters?*



## **Algorithm selection**

*Given a variety of algorithms, which to use?*



## **Algorithm design**

*Can machine learning guide algorithm discovery?*

# Algorithm configuration

## Example: **Integer programming solvers**

Most popular tool for solving combinatorial (& nonconvex) problems



Routing



Manufacturing



Scheduling



Planning



Finance

# Algorithm configuration

IP solvers (CPLEX, Gurobi) have a **ton** parameters

- CPLEX has **170-page** manual describing **172** parameters
- Tuning by hand is notoriously **slow, tedious,** and **error-prone**

CPX_PARAM_NODEFILEIND 100	CPX_PARAM_TRELIM 160	CPX_PARAM_RANDOMSEED 130	CPXPARAM_MIP_Pool_RelGap 148	CPX_PARAM_FLOWCOVERS 70	CPX_PARAM_BRDIR 39
CPX_PARAM_NODELIM 101	CPX_PARAM_TUNINGDETTILIM 160	CPX_PARAM_REDUCE 131	CPXPARAM_MIP_Pool_Replace 151	CPX_PARAM_FLOWPATHS 71	CPX_PARAM_BTTOL 40
CPX_PARAM_NODESEL 102	CPX_PARAM_TUNINGDISPLAY 162	CPX_PARAM_REINV 131	CPXPARAM_MIP_Strategy_Branch 39	CPX_PARAM_FPHEUR 72	CPX_PARAM_CALCQCPCDUALS 41
CPX_PARAM_NUMERICALEMPHASIS 102	CPX_PARAM_TUNINGMEASURE 163	CPX_PARAM_RELAXPREIND 132	CPXPARAM_MIP_Strategy_MIQCPStrat 93	CPX_PARAM_FRACCAND 73	CPX_PARAM_CLIQUES 42
CPX_PARAM_NZREADLIM 103	CPX_PARAM_TUNINGREPEAT 164	CPX_PARAM_RELOBJDIF 133	CPXPARAM_MIP_Strategy_StartAlgorithm 139	CPX_PARAM_FRACCUTS 73	CPX_PARAM_CLOCKTYPE 43
CPX_PARAM_OBJDIF 104	CPX_PARAM_TUNINGTILIM 165	CPX_PARAM_REPAIRTRIES 133	CPXPARAM_MIP_Strategy_VariableSelect 166	CPX_PARAM_FRACPASS 74	CPX_PARAM_CLONELOG 43
CPX_PARAM_OBJLLIM 105	CPX_PARAM_VARSEL 166	CPX_PARAM_REPEATPRESOLVE 134	CPXPARAM_MIP_SubMIP_NodeLimit 155	CPX_PARAM_GUBCOVERS 75	CPX_PARAM_COEREDIND 44
CPX_PARAM_OBJULIM 105	CPX_PARAM_WORKDIR 167	CPX_PARAM_RINSHEUR 135	CPXPARAM_OptimalityTarget 106	CPX_PARAM_HEURFREQ 76	CPX_PARAM_COLREADLIM 45
CPX_PARAM_PARALLELMODE 108	CPX_PARAM_WORKMEM 168	CPX_PARAM_RLT 136	CPXPARAM_Output_WriteLevel 169	CPX_PARAM_IMPLBD 76	CPX_PARAM_CONFLICTDISPLAY 46
CPX_PARAM_PERIND 110	CPX_PARAM_WRITELEVEL 169	CPX_PARAM_ROWREADLIM 141	CPXPARAM_Preprocessing_Aggregator 19	CPX_PARAM_INTSOLFILEPREFIX 78	CPX_PARAM_COVERS 47
CPX_PARAM_PERLIM 111	CPX_PARAM_ZEROHALFCUTS 170	CPX_PARAM_SCAIND 142	CPXPARAM_Preprocessing_Fill 19	CPX_PARAM_INTSOLLIM 79	CPX_PARAM_CPUMASK 48
CPX_PARAM_POLISHAFTERDETTIME 111	CPXPARAM_Benders_Strategy 30	CPX_PARAM_SCRIND 143	CPXPARAM_Preprocessing_Linear 120	CPX_PARAM_ITLIM 80	CPX_PARAM_CRAIN 50
CPX_PARAM_POLISHAFTEREPAGAP 112	CPXPARAM_Benders_Tolerances_feasibilitycut 35	CPX_PARAM_SIFTALG 143	CPXPARAM_Preprocessing_Reduce 131	CPX_PARAM_LANDPCUTS 82	CPX_PARAM_CUTLO 51
CPX_PARAM_POLISHAFTEREPGAP 113	CPXPARAM_Benders_Tolerances_optimalitycut 36	CPX_PARAM_SIFTDISPLAY 144	CPXPARAM_Preprocessing_Symmetry 156	CPX_PARAM_LBHEUR 81	CPX_PARAM_CUTPASS 52
CPX_PARAM_POLISHAFTERINTSOL 114	CPXPARAM_Conflict_Algorithm 46	CPX_PARAM_SIFTTILIM 145	CPXPARAM_Read_DataCheck 54	CPX_PARAM_LPMETHOD 136	CPX_PARAM_CUTSFACTOR 52
CPX_PARAM_POLISHAFTERNODE 115	CPXPARAM_CPUmask 48	CPX_PARAM_SIMDISPLAY 145	CPXPARAM_Read_Scale 142	CPX_PARAM_MFCUTS 82	CPX_PARAM_CUTUP 53
CPX_PARAM_POLISHAFTERTIME 116	CPXPARAM_DistMIP_Rampup_Duration 128	CPX_PARAM_SINGLIM 146	CPXPARAM_ScreenOutput 143	CPX_PARAM_MEMORYEMPHASIS 83	CPXPARAM_DATACHECK 54
CPX_PARAM_POLISHTIME (deprecated) 116	CPXPARAM_LPMethod 136	CPX_PARAM_SOLNPOOLGAP 146	CPXPARAM_Sifting_Algorithm 143	CPX_PARAM_MIPCBREDLP 84	CPX_PARAM_DEPIND 55
CPX_PARAM_POPULATELIM 117	CPXPARAM_MIP_Cuts_BQP 38	CPX_PARAM_SOLNPOOLCAPACITY 147	CPXPARAM_Sifting_Display 144	CPX_PARAM_MIPDISPLAY 85	CPX_PARAM_DETTILIM 56
CPX_PARAM_PPRIND 118	CPXPARAM_MIP_Cuts_LocalImplied 77	CPX_PARAM_SOLNPOOLREPLACE 151	CPXPARAM_Sifting_Iterations 145	CPX_PARAM_MIPEMPHASIS 87	CPX_PARAM_DISJCUTS 57
CPX_PARAM_PREDUAL 119	CPXPARAM_MIP_Cuts_RLT 136	CPX_PARAM_SOLNPOOLINTENSITY 149	CPXPARAM_Simplex_Display 145	CPX_PARAM_MIPINTERVAL 88	CPX_PARAM_DIVETYPE 58
CPX_PARAM_PREIND 120	CPXPARAM_MIP_Cuts_ZeroHalfCut 170	CPX_PARAM_SOLNPOOLREPLACE 151	CPXPARAM_Simplex_Limits_Singularity 146	CPX_PARAM_MIPKAPPASTATS 89	CPX_PARAM_DPRIIND 59
CPX_PARAM_PRLINEAR 120	CPXPARAM_MIP_Limits_CutsFactor 52	CPX_PARAM_SOLUTIONTARGET (deprecated: see CPXPARAM_OptimalityTarget 106)	CPXPARAM_SolutionType 152	CPX_PARAM_MIPORDIND 90	CPX_PARAM_EACHCUTLIM 60
CPX_PARAM_PREPASS 121	CPXPARAM_MIP_Limits_RampupDetTimeLimit 127	CPXPARAM_SOLUTIONTYPE 152	CPXPARAM_Threads 157	CPX_PARAM_MIPORDTYPE 91	CPX_PARAM_EPAGAP 61
CPX_PARAM_PRESLVND 122	CPXPARAM_MIP_Limits_RampupTimeLimit 128	CPX_PARAM_STARTALG 139	CPXPARAM_TimeLimit 159	CPX_PARAM_MIPSEARCH 92	CPX_PARAM_EPGAP 61
CPX_PARAM_PRICELIM 123	CPXPARAM_MIP_Limits_Solutions 79	CPX_PARAM_STRONGCANDLIM 154	CPXPARAM_Tune_DefTimeLimit 160	CPX_PARAM_MIQCPSTRAT 93	CPX_PARAM_EPINT 62
CPX_PARAM_PROBE 123	CPXPARAM_MIP_Limits_StrongCand 154	CPX_PARAM_STRONGCANDLIM 154	CPXPARAM_Tune_Display 162	CPX_PARAM_MIRCUTS 94	CPX_PARAM_EPMRK 64
CPX_PARAM_PROBEDETTIME 124	CPXPARAM_MIP_Limits_StrongIt 154	CPX_PARAM_STRONGITLIM 154	CPXPARAM_Tune_Measure 163	CPX_PARAM_MPSLONGNUM 94	CPX_PARAM_EPOPT 65
CPX_PARAM_PROBETIME 124	CPXPARAM_MIP_Limits_TreeMemory 160	CPX_PARAM_SUBALG 99	CPXPARAM_Tune_Repeat 164	CPX_PARAM_NETDISPLAY 95	CPX_PARAM_EPPER 65
CPX_PARAM_QPMAKEPSDIND 125	CPXPARAM_MIP_OrderType 91	CPX_PARAM_SUBMIPNODELIMIT 155	CPXPARAM_Tune_TimeLimit 165	CPX_PARAM_NETEPOPT 96	CPX_PARAM_EPRELAX 66
CPX_PARAM_QPMETHOD 138	CPXPARAM_MIP_Pool_AbsGap 146	CPX_PARAM_SYMMETRY 156	CPXPARAM_WorkDir 167	CPX_PARAM_NETEPRHS 96	CPX_PARAM_EPRHS 67
CPX_PARAM_QPNZREADLIM 126	CPXPARAM_MIP_Pool_Capacity 147	CPX_PARAM_THREADS 157	CPXPARAM_WorkMem 168	CPX_PARAM_NETFIND 97	CPX_PARAM_FEASOPTMODE 68
	CPXPARAM_MIP_Pool_Intensity 149	CPX_PARAM_TILIM 159	CraInd 50	CPX_PARAM_NETITLIM 98	CPX_PARAM_FILEENCODING 69
				CPX_PARAM_NETPRIIND 98	



# Algorithm configuration

IP solvers (CPLEX, Gurobi) have a **ton** parameters

- CPLEX has **170-page** manual describing **172** parameters
- Tuning by hand is notoriously **slow, tedious**, and **error-prone**

What's the best **configuration** for the application at hand?



Best configuration for **routing** problems  
likely not suited for **scheduling**



# How to integrate **machine learning** into **discrete optimization**?

- **Algorithm configuration**  
*How to tune an algorithm's parameters?*
- **Algorithm selection**  
*Given a variety of algorithms, which to use?*
- **Algorithm design**  
*Can machine learning guide algorithm discovery?*

# Example: Clustering

Many different algorithms

K-means



Mean shift



Ward



Agglomerative



Birch



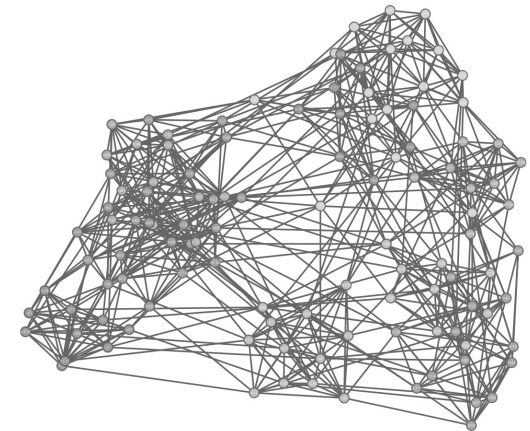
How to **select** the best algorithm for the application at hand?

# Algorithm selection in theory

**Worst-case analysis** has been the main framework for decades  
*Has led to beautiful, practical algorithms*

Worst-case analysis's approach to **algorithm selection**:  
Select the algorithm that's best in worst-case scenarios

Worst-case instances **rarely occur in practice**



# How to integrate **machine learning** into **discrete optimization**?

- **Algorithm configuration**  
*How to tune an algorithm's parameters?*
- **Algorithm selection**  
*Given a variety of algorithms, which to use?*
- **Algorithm design**  
*Can machine learning guide algorithm discovery?*

# How to integrate **machine learning** into **discrete optimization**?

## **My objective:**

Future engineers will have a nuanced understanding of ML's  
**power** and **limitations**  
when used to solve discrete optimization problems

# How to integrate **machine learning** into **discrete optimization**?

## **My long-term goal:**

Researchers will be empowered with **data-driven tools** to

 Conceive

 Prototype

 Validate

algorithmic ideas...

**and** provide theoretical guarantees for their discoveries

# How to integrate **machine learning** into **discrete optimization**?

Area is built on a key observation:

**In practice, we have data about  
the application domain**




A stack of several cardboard boxes, each secured with red and white striped string. The boxes are brown and feature a 'FRAGILE' label with three icons: a vertical arrow, a wine glass, and an umbrella. The background is blurred, showing what appears to be a warehouse or shipping area with other boxes and equipment.

**In practice, we have data about  
the application domain**

Routing problems a shipping company solves

**In practice, we have data about  
the application domain**



Clustering problems a biology lab solves

**In practice, we have data about  
the application domain**



Scheduling problems an airline solves

# In practice, we have data about the application domain

How can we use this data to guide:

- **Algorithm configuration**  
*How to tune an algorithm's parameters?*
- **Algorithm selection**  
*Given a variety of algorithms, which to use?*
- **Algorithm design**  
*Can machine learning guide algorithm discovery?*

# ML + discrete opt: Potential impact

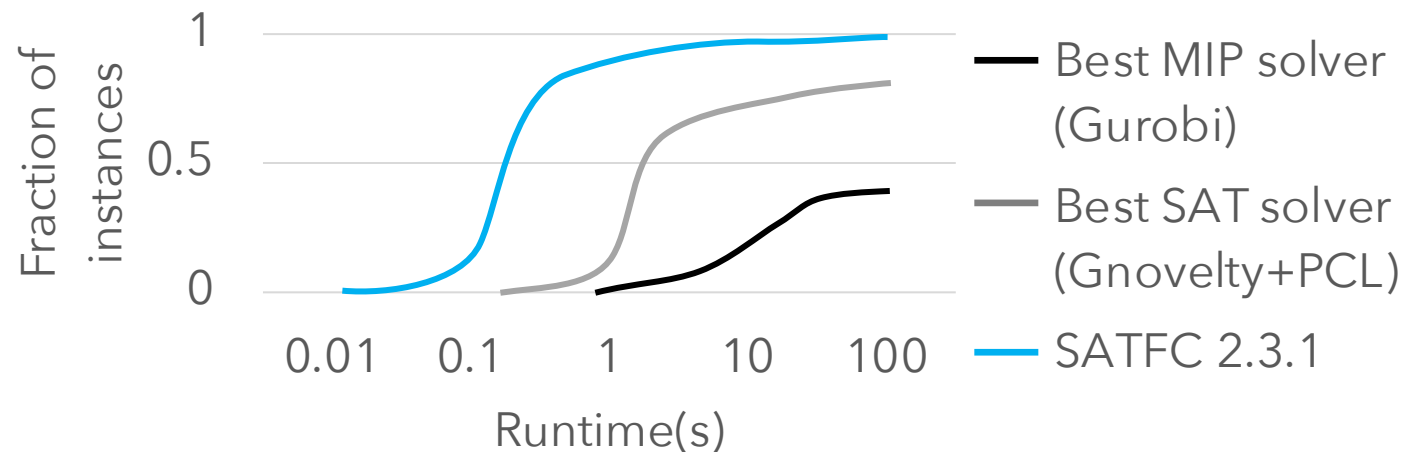
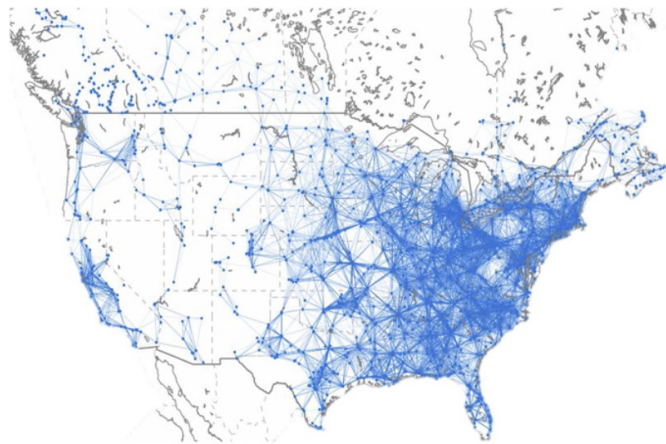
## Example: integer programming

- Used heavily throughout industry and science
- **Many** different ways to incorporate **learning** into solving
- Solving is very difficult, so ML can make a huge difference



# Example: Spectrum auctions

- In '16-'17, FCC held a \$19.8 billion radio spectrum auction
  - Involves solving huge graph-coloring problems



- SATFC uses algorithm configuration + selection
- Simulations indicate SATFC saved the government billions

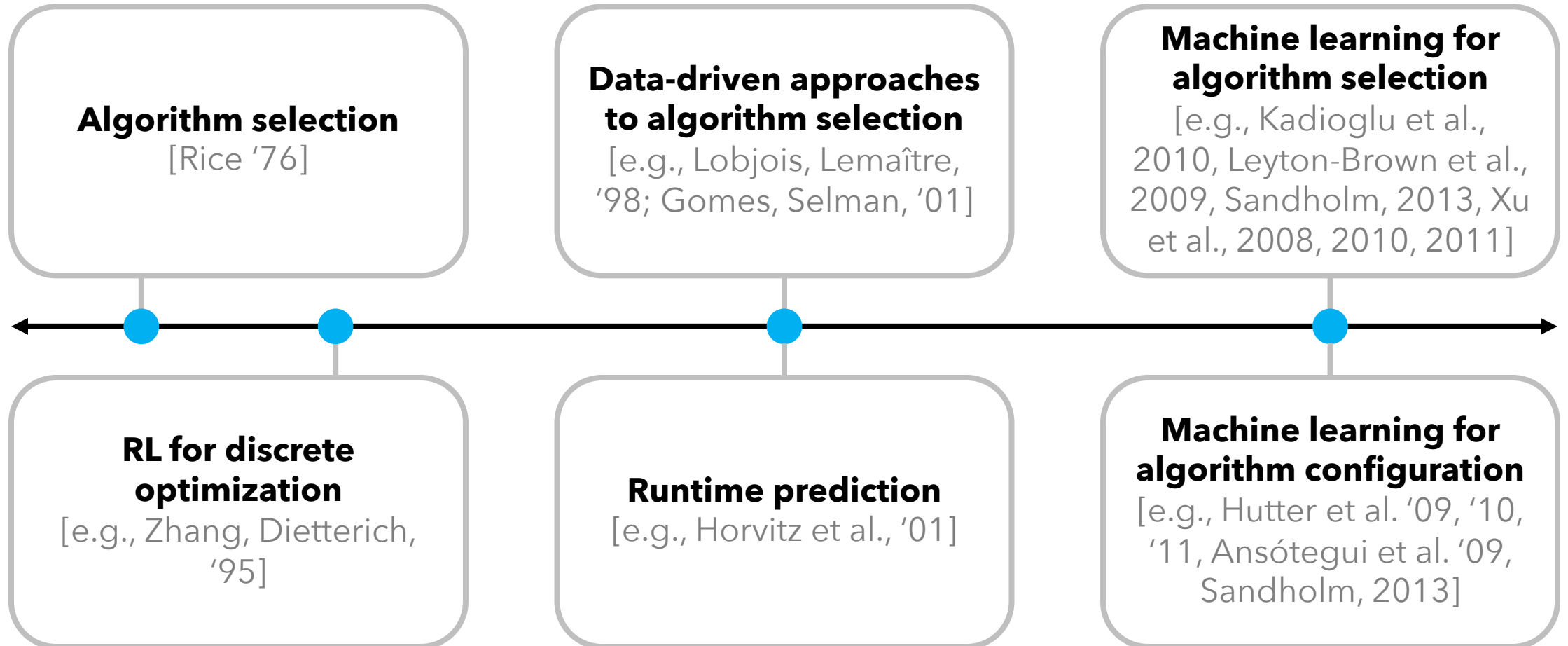
# A bit of history

Important research direction in artificial intelligence for decades

Has led to **breakthroughs** in

- Combinatorial auction winner determination
- SAT
- Constraint satisfaction
- Integer programming
- Many other areas

# A bit of history





# A bit of history



# Course topics

Range of techniques for integrating ML into algorithm design

## 1. Applied topics

- i. Sequence-to-sequence models
- ii. Graph neural networks
- iii. Transformers and LLMs

## 2. Theoretical topics

- i. Statistical guarantees and online algorithm configuration
- ii. Algorithms with predictions

# Outline

1. Introduction
- 2. Course logistics**
3. Applied topics
4. Theoretical topics

# Course overview

Website: [vitercik.github.io/ml4do](https://vitercik.github.io/ml4do)

On the website, you can find syllabus information like:

- Office hours
- Project policy
- Homework late policy
- Schedule of topics with supplementary readings

# Prerequisites

Introductory course in **algorithms/optimization**

- E.g., CS 161 or MS&E 111/211

Introductory course in **machine learning**

- E.g., CS 229
- You should be familiar with basic feed-forward neural networks

# Class breakdown

10% Participation

45% Homework assignments (3 total)

45% Project

# Policies

## **10% Participation**

- In-class participation recorded using in-class polls (starting next class)
- Can't come class?
  - Watch lecture online (posted after class on Canvas)
  - Send 100-word summary to instructors [details will be announced]
    - Must be set before the start of the next class
- 2 absent passes, no summary necessary, no questions asked

# Policies

## **45% Weekly assignments (3 total)**

- Total of 4 late days for assignments, e.g.:
  - No penalty if you submit 1 assignment 4 days late
  - Or 2 assignments 2 days late, ...
- Beyond that, grade goes down by 7 points for every 12 hours it's late
  - E.g., 90% to 83%
  - Lasts until week after deadline, at which point assignment will receive grade 0%
- **Ask questions on Ed Discussion** (linked to on Canvas)
  - Fastest way to reach course staff



# Policies

Policies intended to cover all

- Sickneses
- Family events
- Sports events
- ...

Use your late days carefully!

**Please** *come talk to me if you're struggling!*

# Policies

## **45% Project**

- Write a “mini-paper” as a final project
- Can take one of two forms:
  - Research
  - Survey

# Option 1: Research project

Present progress your group made on a relevant problem

Report should adopt the structure of a research paper  
(Not required to reach the standard for academic publishing)

# Option 2: Survey project

Choose **2-4 papers** discussed in class. For each paper:

1. Summarize a paper that **the paper covered in class cites**  
*How does the paper covered in class build on the older paper?*
2. Summarize a paper that **cites the paper covered in class**  
*How does the more recent paper build on the paper covered in class?*
3. Imagine you're a **new researcher** working in this area
  - Propose an imaginary follow-up project
  - Not just based on the paper covered in class...  
but **only possible** due to the existence and success of that paper

# Working in groups

- Welcome to work in groups on the final project
- Groups should include:
  - At most three students if it's a **research** project
  - At most two students if it's a **survey** project
- Group of two must put twice as much work into project
  - Similarly for groups of three
- The **paper length** for final write-up is:
  - 3 if solo-authored,
  - 5 if there are two authors, and
  - 7 if there are three authors

# Milestones

**May 1:** Submit a short progress report of 1-2 pages  
Describe your project and partial progress

**June 5:** Students will present their final project during class

**June 12:** Each group will submit their final report

# Class format

## Whiteboard!

- Studies show that students learn better from whiteboard vs. slides
- Writing down notes helps you learn
  - As opposed to just following along in slides
- I automatically go slower

**Please ask questions in class!**

# 2-minute anonymous surveys

- Watch out for an email about a 2-min anonymous survey
- Random set of students asked each week
  - You'll be asked 2-3 times during the quarter to fill it out
- It's so useful for us!

Please use it to tell us:

- What's going well 😊
- What you're confused about 🤔
- How we can best help you learn!



# OAE

Let me know if you have an OAE letter as soon as possible

***Thanks!***

# Outline

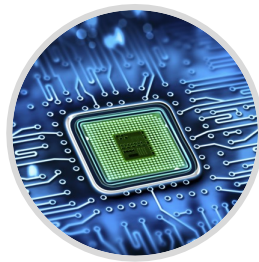
1. Introduction
2. Course logistics
- 3. Applied topics**
  - a. Pointer networks for the traveling salesman problem (TSP)**
  - b. Graph neural networks
  - c. Transformers and LLMs
4. Theoretical topics

# Traveling salesman problem

- One of the most famous *NP-hard* problems
- **Input:** network with  $n$  nodes, representing a map with  $n$  cities
- **Goal:** compute the shortest-distance tour  
*Should pass through each city exactly once*



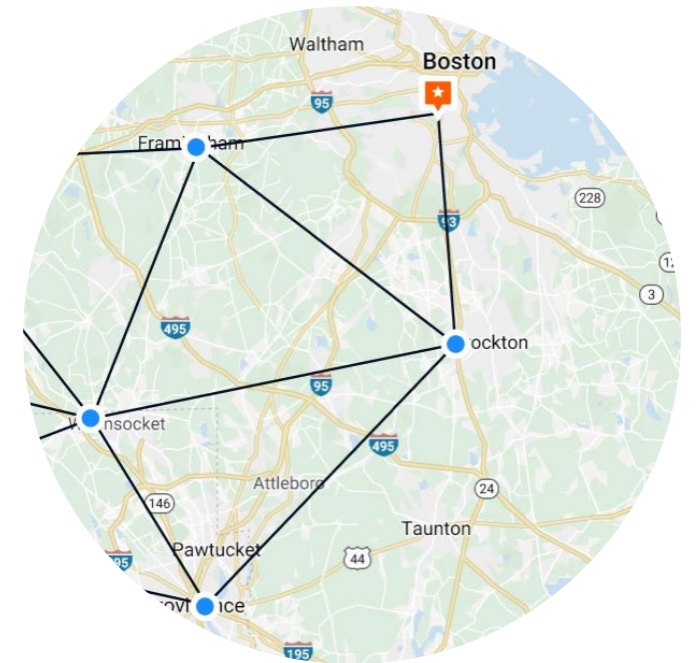
Routing



Manufacturing



Planning

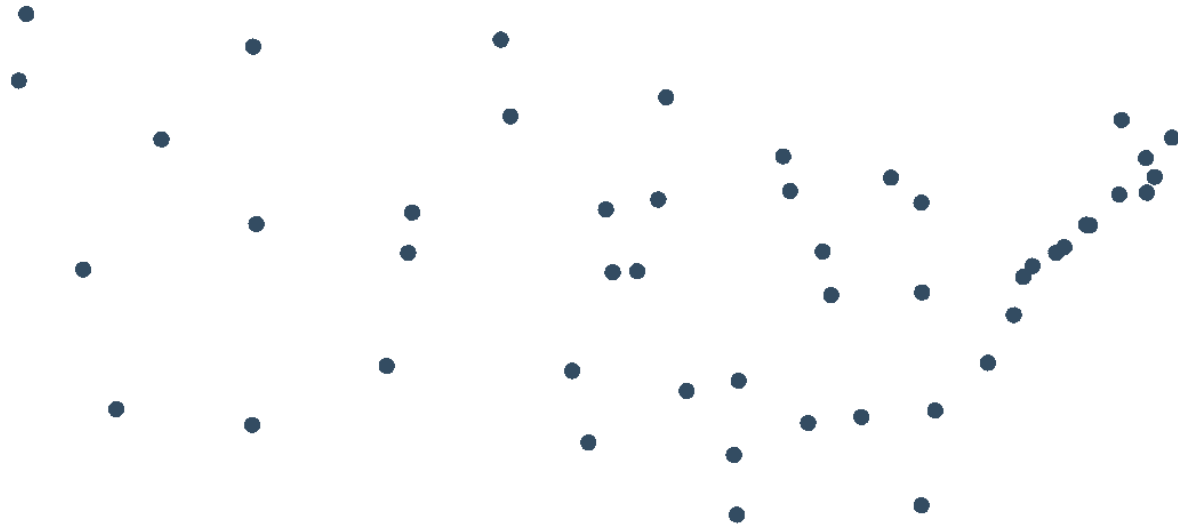


# Farthest-first heuristic

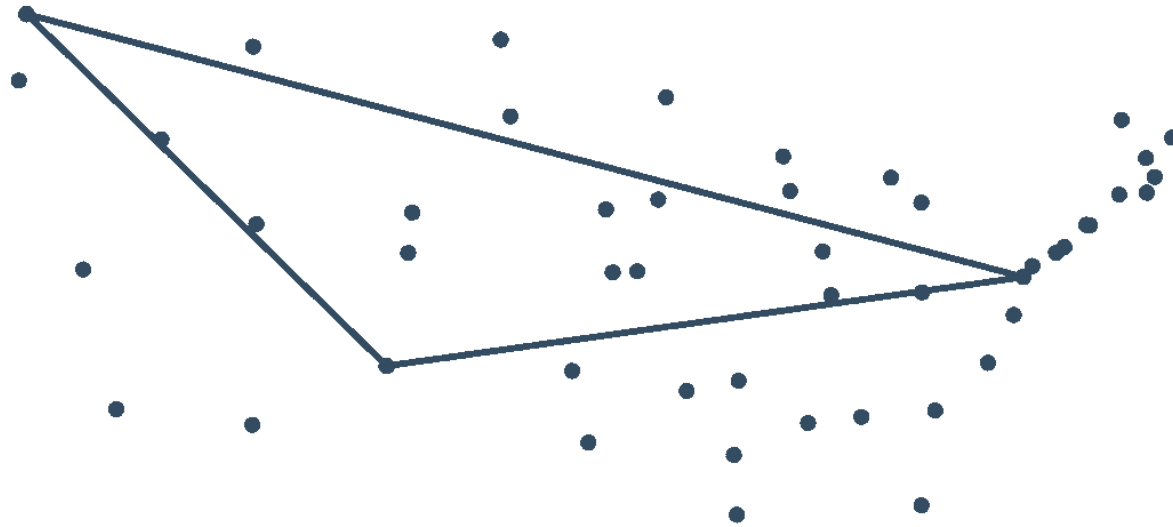
Many **heuristics** for TSP

1. Start with subtour {1}
2. Among all cities not in the subtour:  
Choose the one that's *farthest* from any city in the subtour
3. Insert it into the subtour  
Position: where it causes the smallest tour length increase

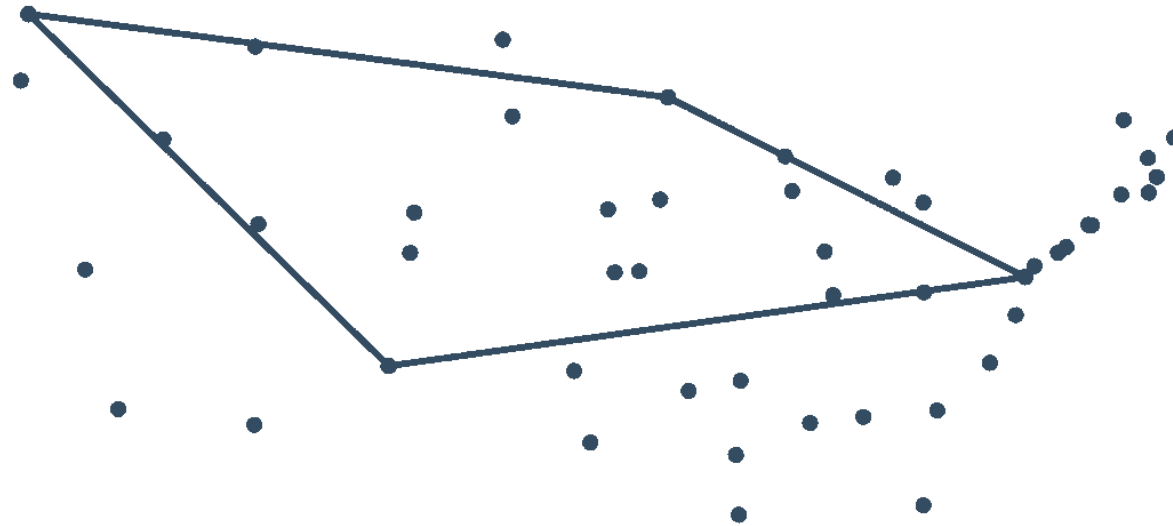
# Farthest-first heuristic



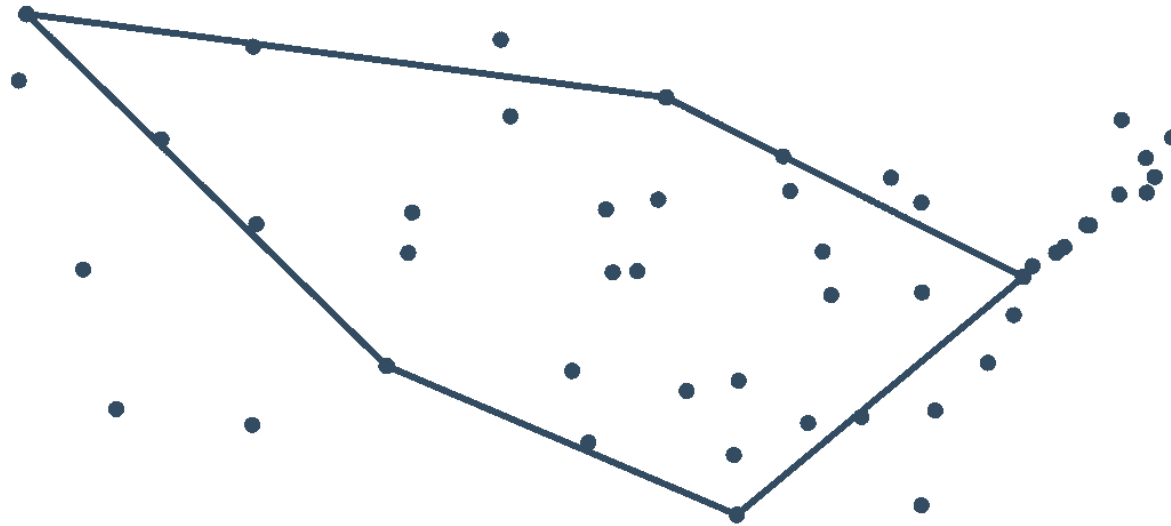
# Farthest-first heuristic



# Farthest-first heuristic

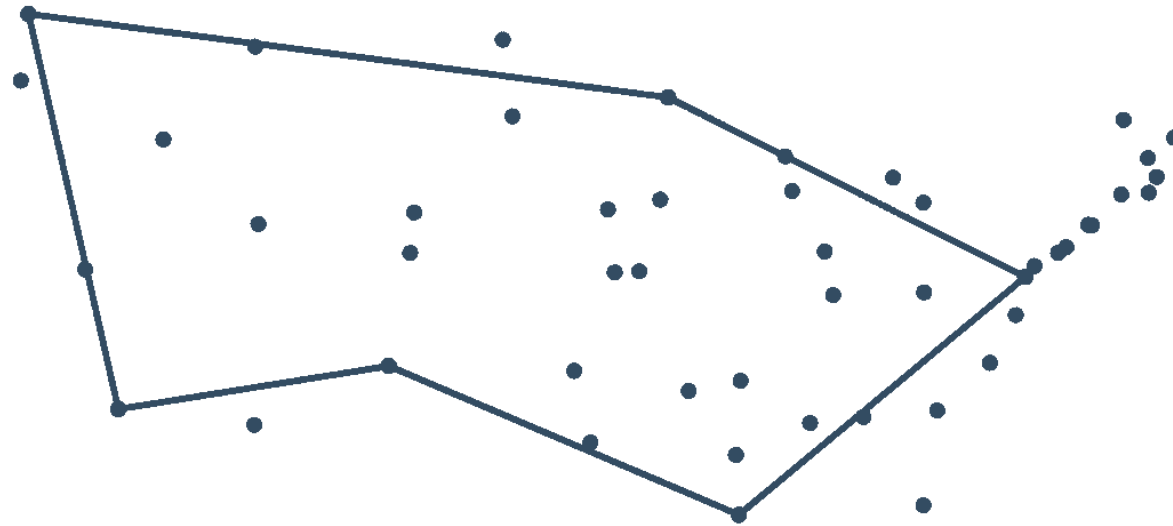


# Farthest-first heuristic

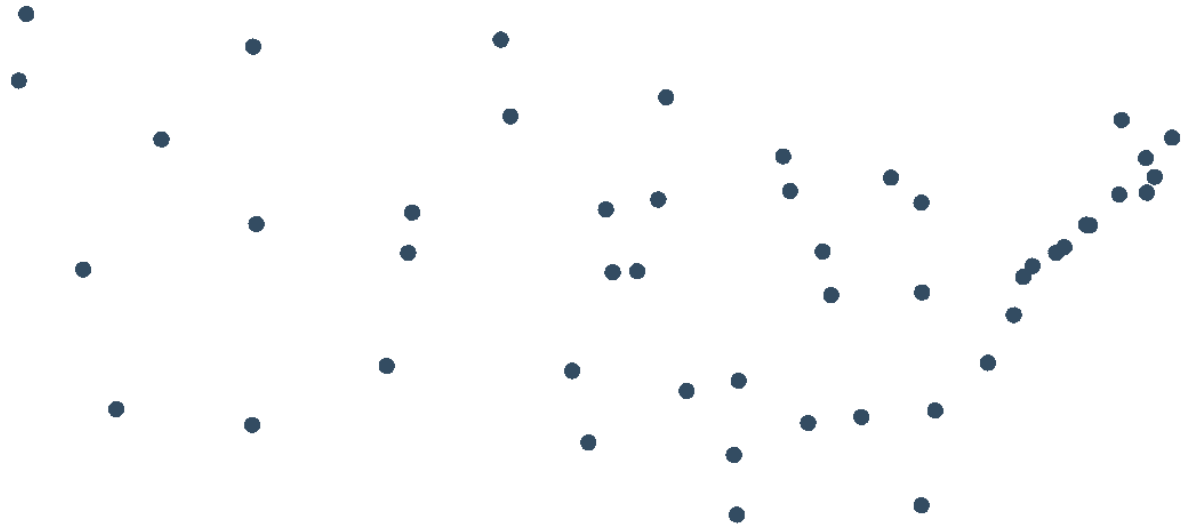




# Farthest-first heuristic



# Farthest-first heuristic



# Heuristics for TSP

Many **heuristics** for TSP

- Nearest neighbor
- Nearest insertion
- Cheapest insertion
- Random insertion
- Christofides algorithm
- ...

**Goal:** use ML to uncover a better heuristic

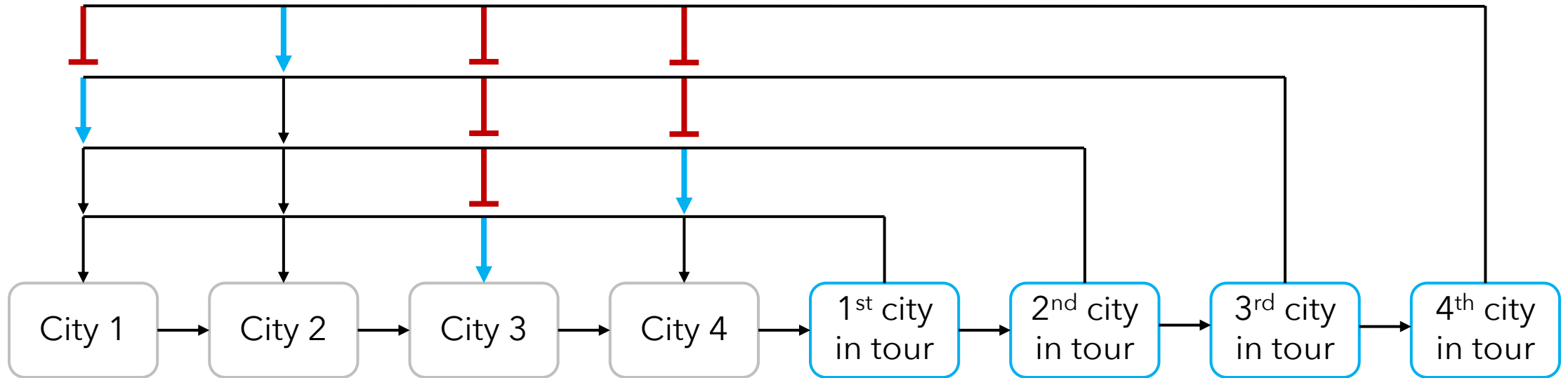
# Topics

**This module:** early approaches to DL for discrete optimization  
*circa '15-'17*

We'll cover:

- **Recurrent neural networks**  
*Long-short term memories (LSTMs)*
- **Pointer networks:** use LSTMs to output a permutation
- **Training** pointer networks for TSP (*policy gradient*)

# Pointer networks



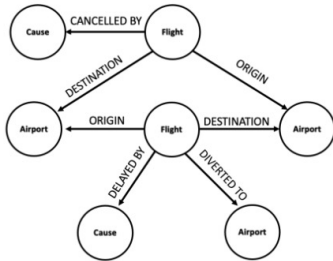
Shows promise that ML can be useful for discrete optimization

But treats cities as a sequence, losing the network structure 🤔

# Outline

1. Introduction
2. Course logistics
3. Applied topics
  - a. Pointer networks for the traveling salesman problem (TSP)
  - b. Graph neural networks**
  - c. Transformers and LLMs
4. Theoretical topics

# Many types of data are graphs

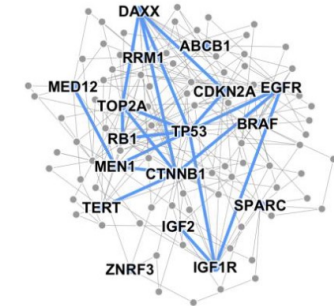


**Event Graphs**



Image credit: [SalientNetworks](#)

**Computer Networks**



**Disease Pathways**

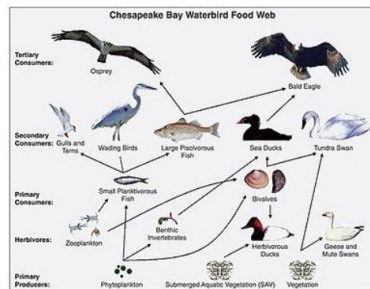


Image credit: [Wikipedia](#)

**Food Webs**



Image credit: [Pinterest](#)

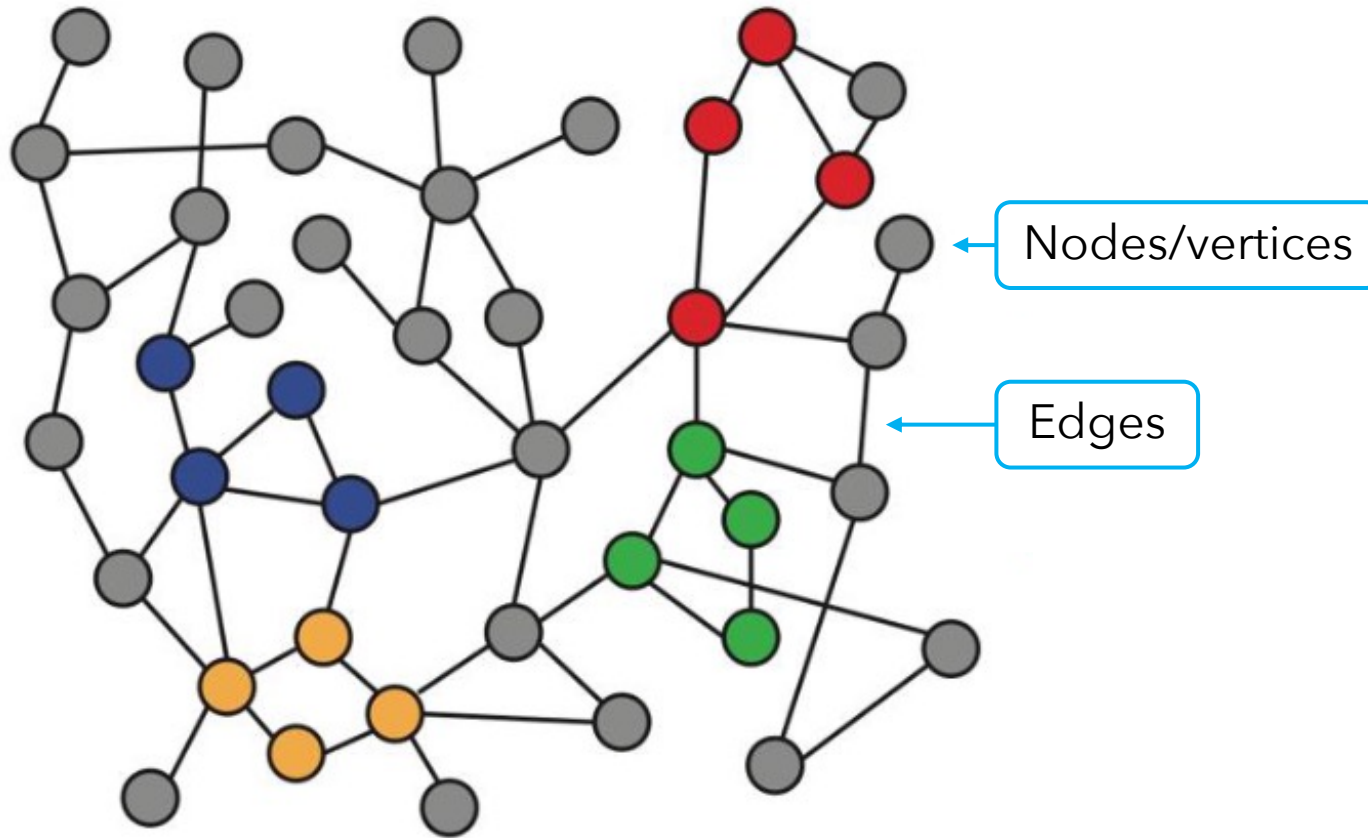
**Particle Networks**



Image credit: [visitlondon.com](#)

**Underground Networks**

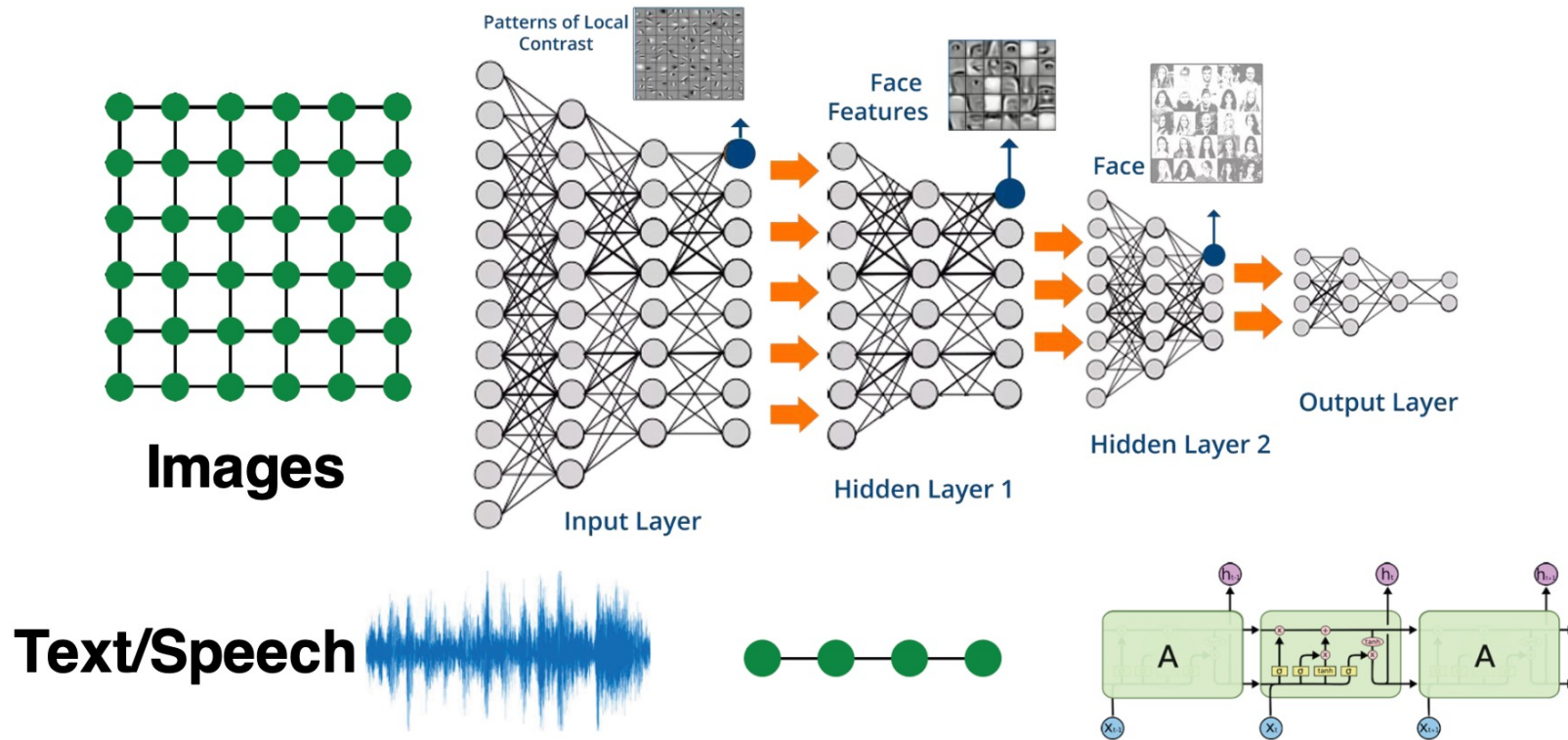
# Graph terminology





# Today: Modern ML toolbox

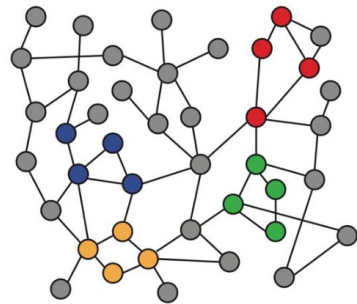
Modern DL toolbox is designed for simple sequences & grids



# Why is graph deep learning hard?

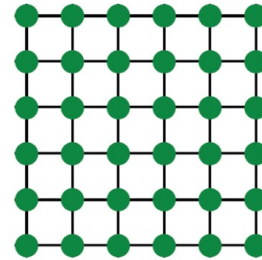
Networks are complex

- Arbitrary size and complex topological structure



**Networks**

versus



**Images**



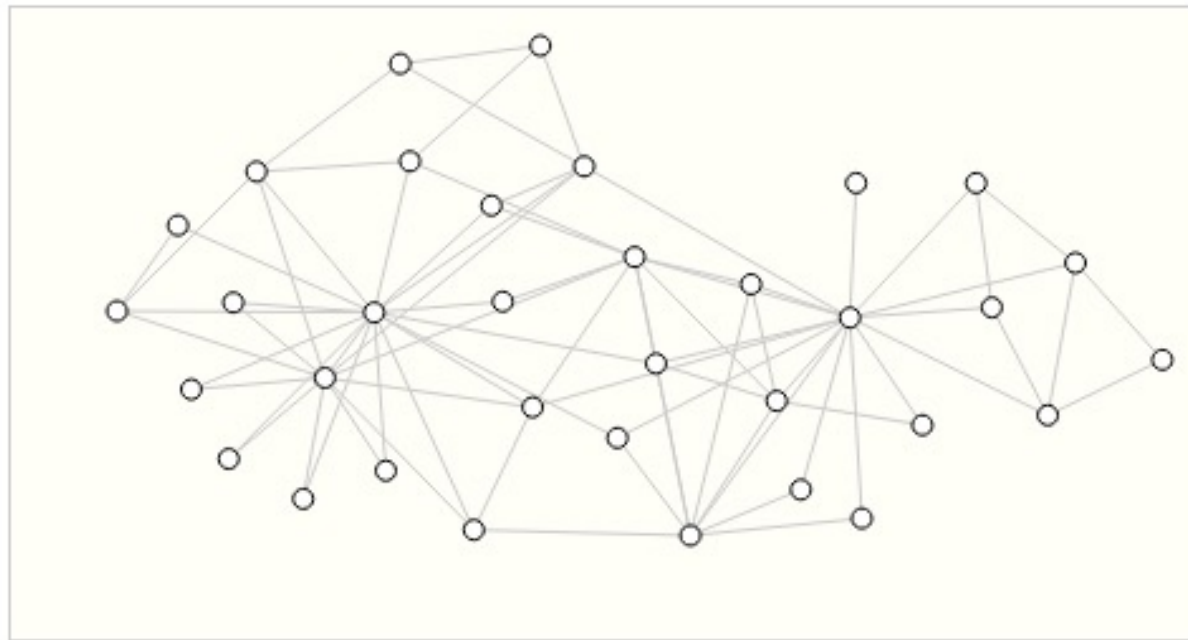
**Text**

- No fixed node ordering or reference point
- Often dynamic and have multimodal features

# GNN motivation

Special type of NN architecture for tasks involving graphs

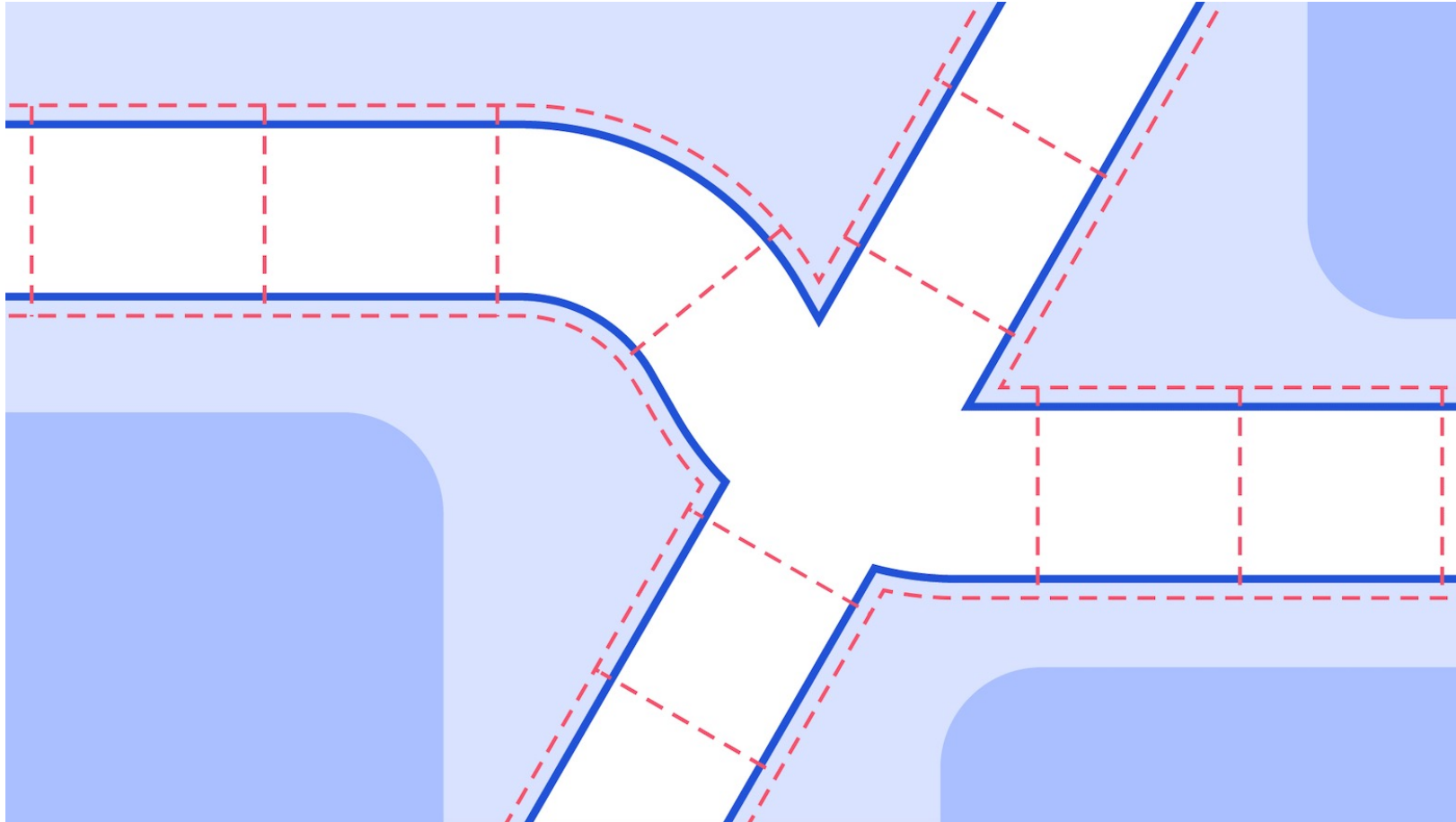
*How to utilize relational structure for better prediction?*



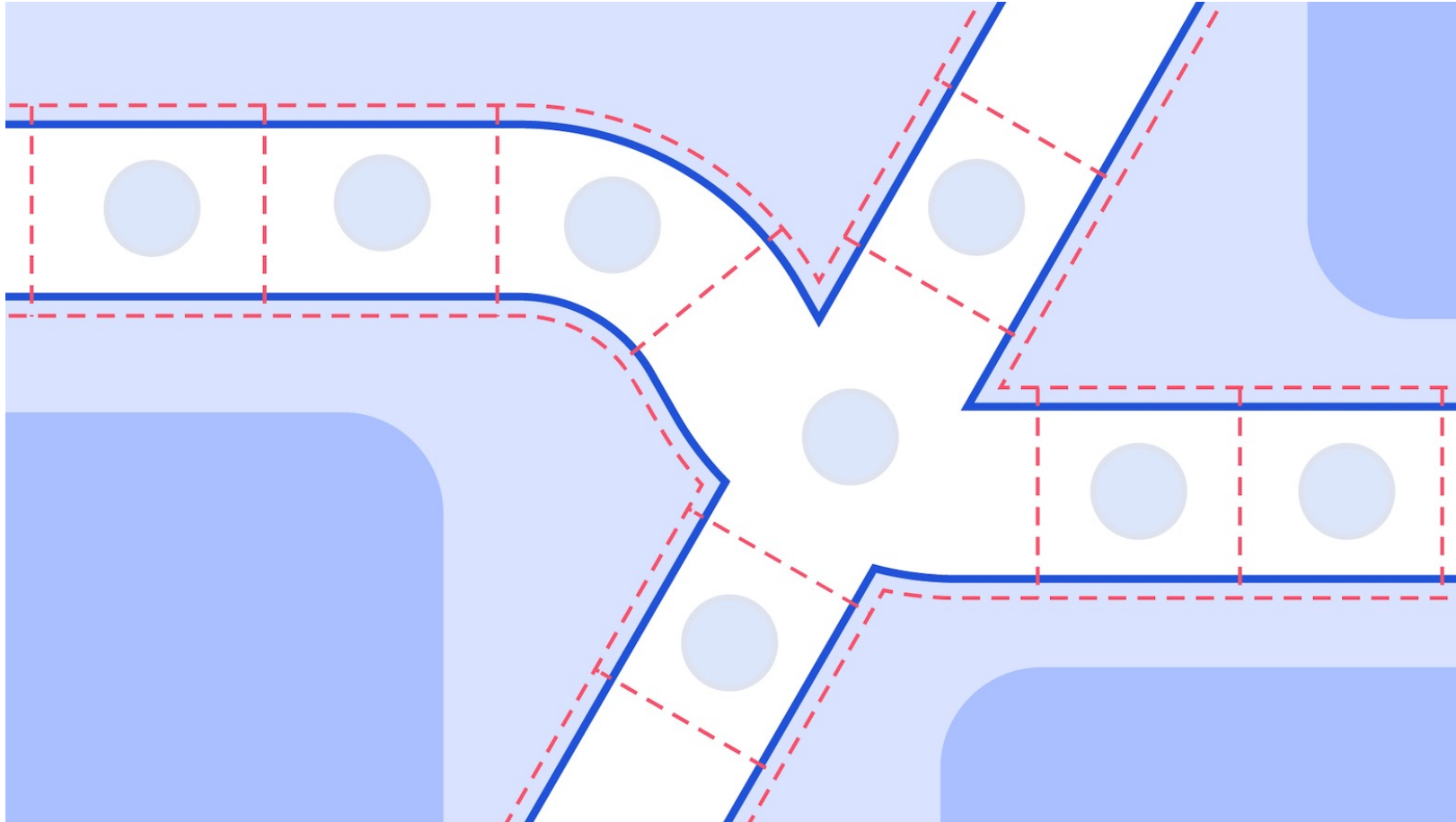
# GNN overview



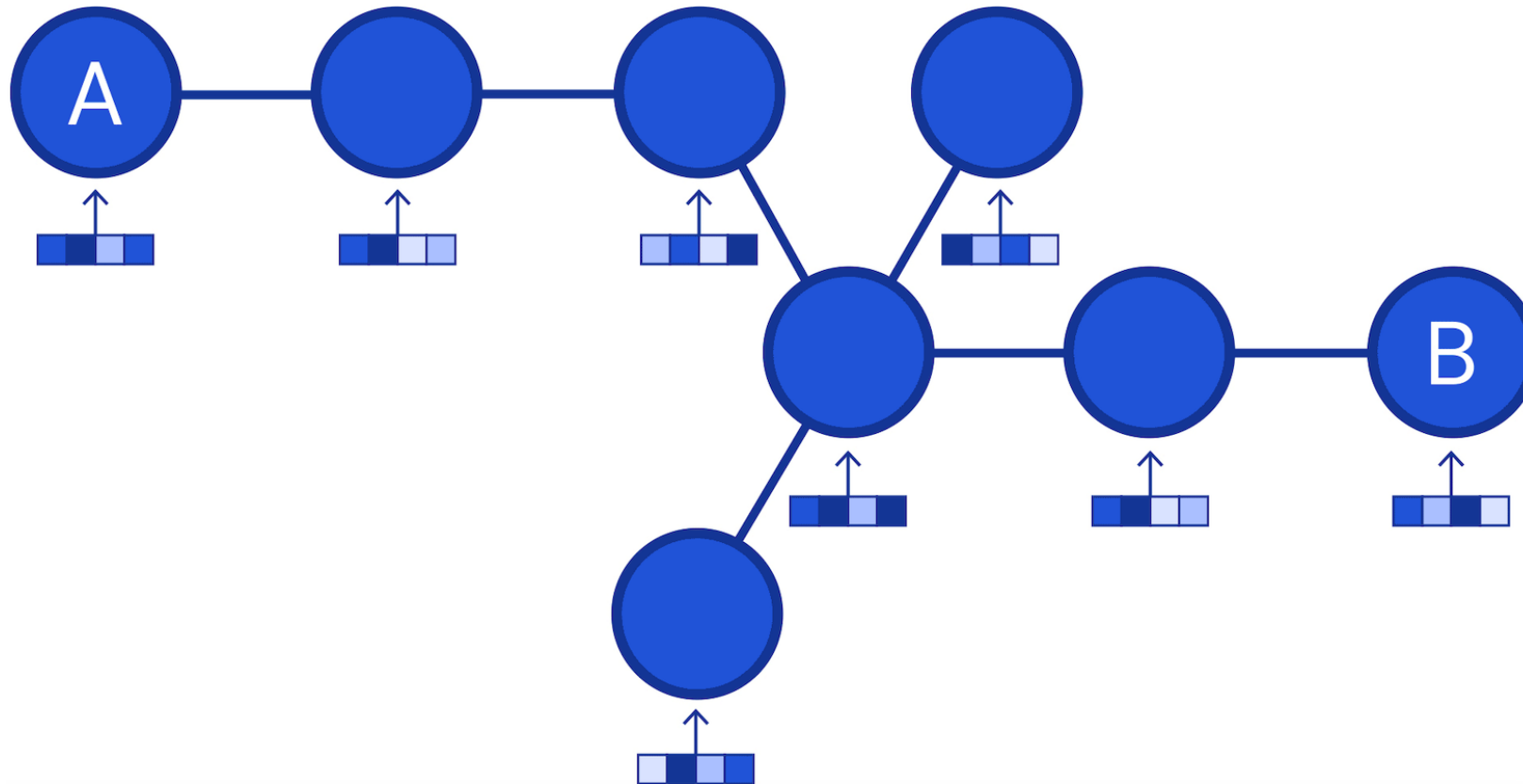
# GNN overview



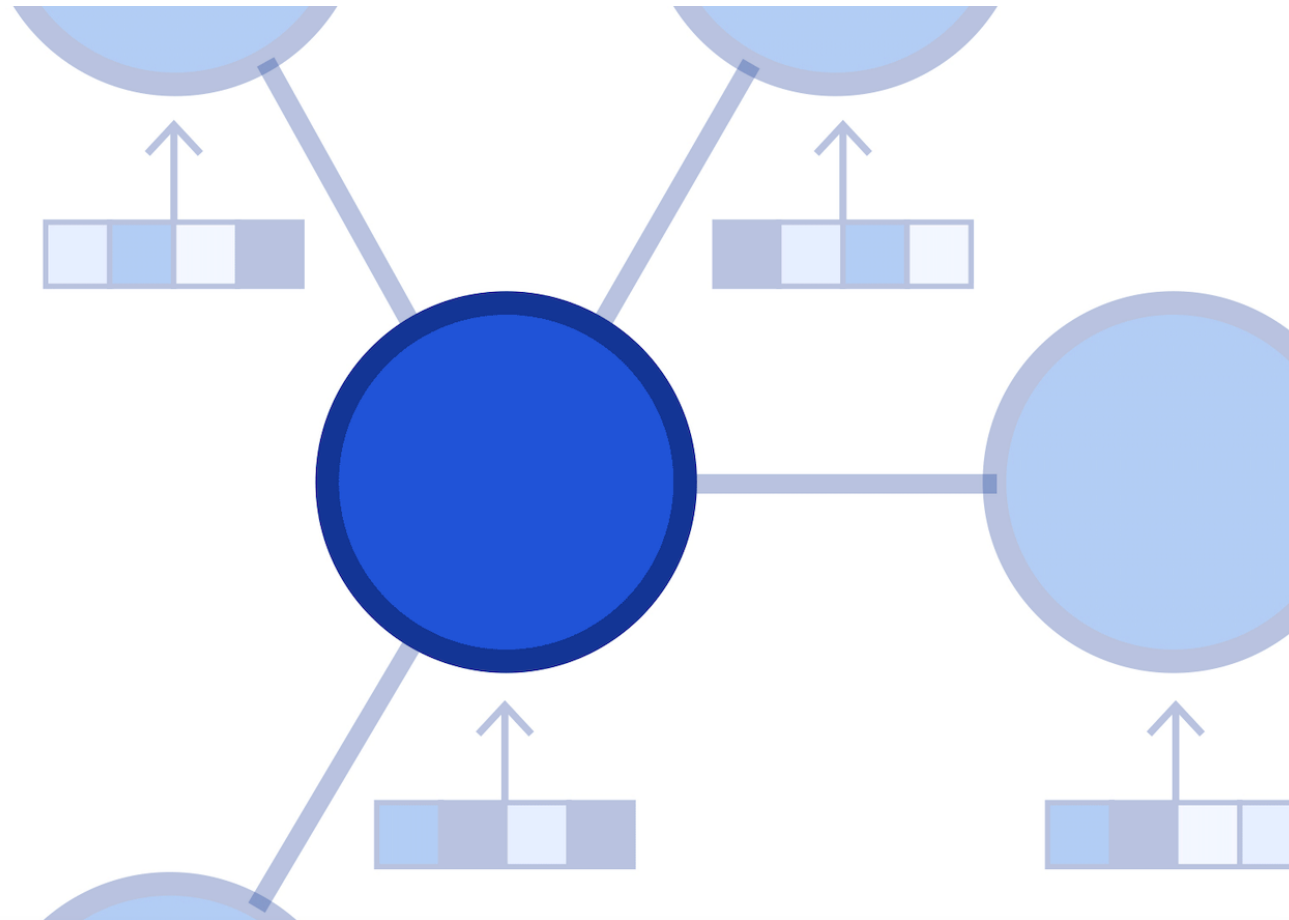
# GNN overview



# GNN overview

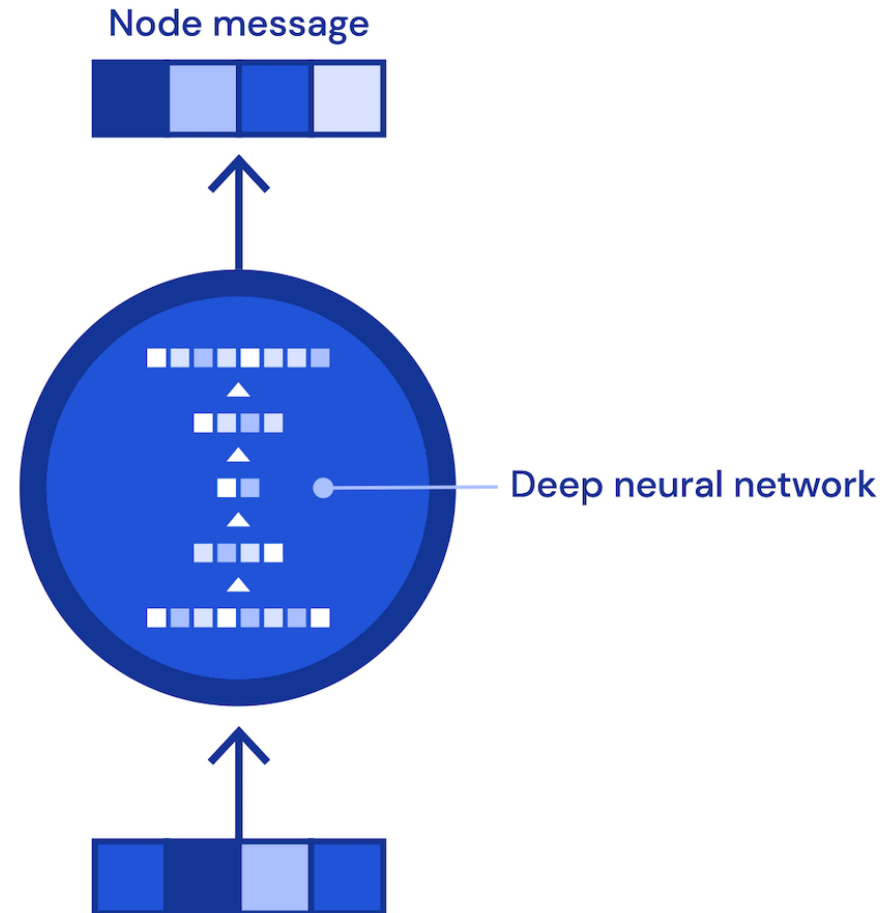


# GNN overview

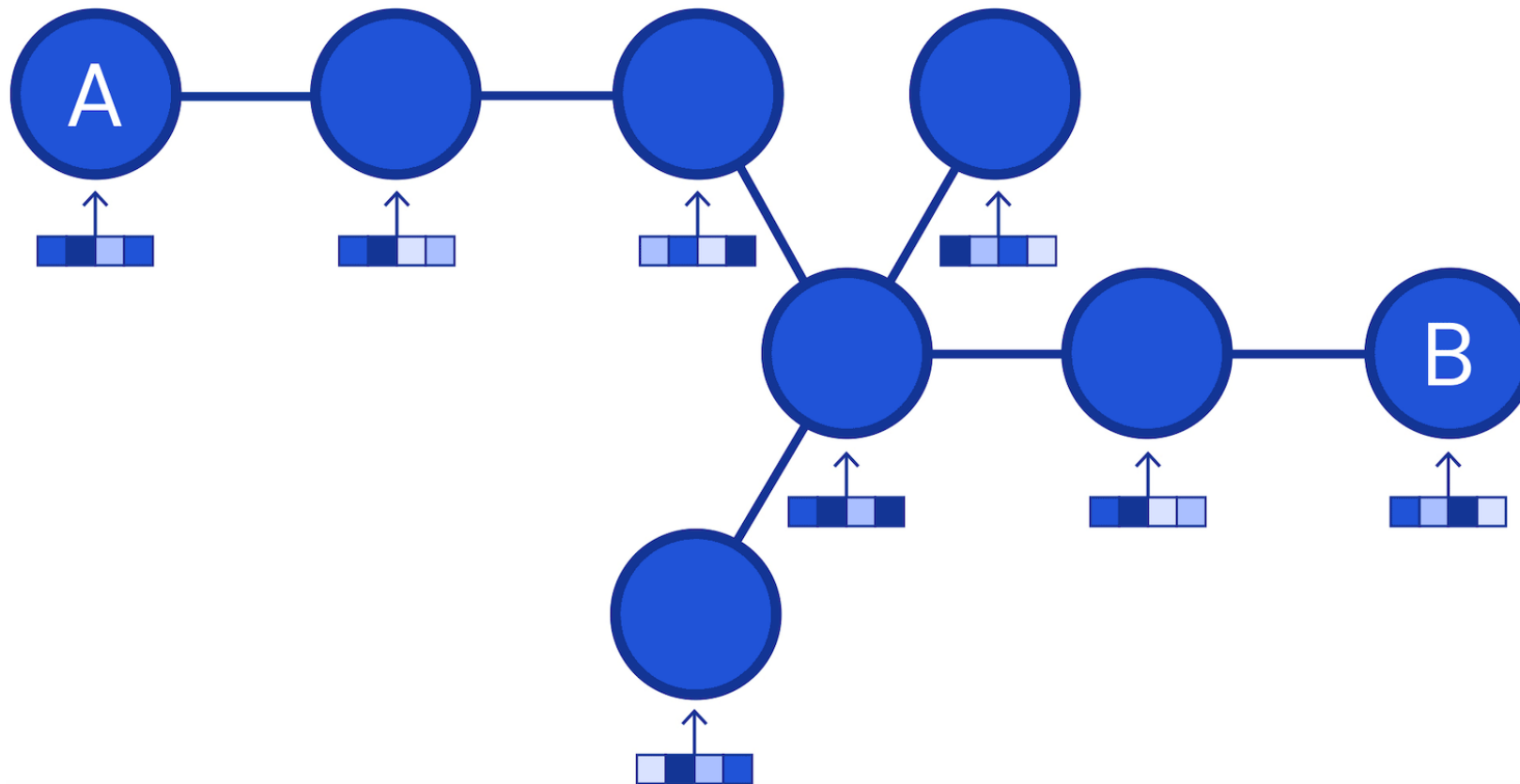




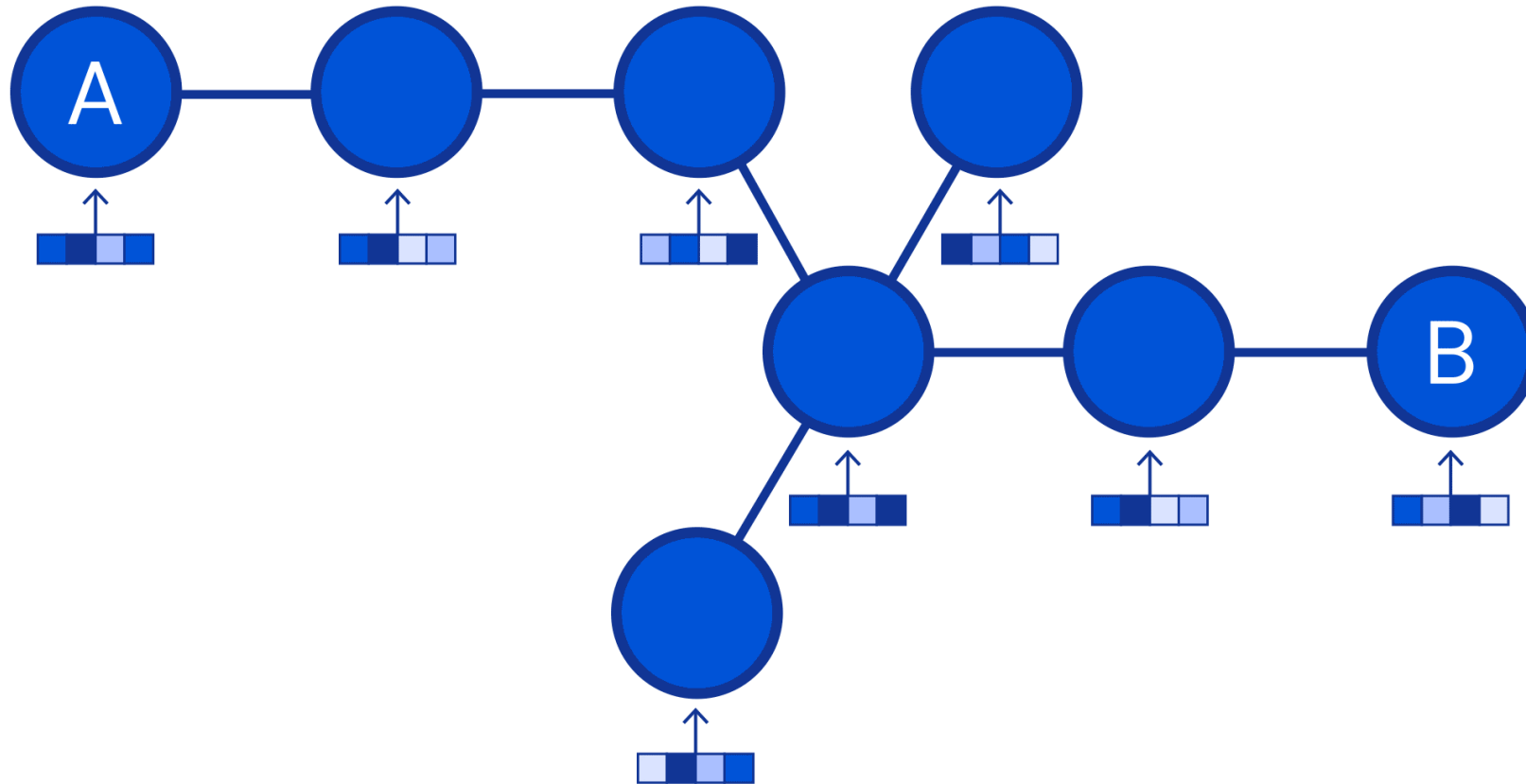
# GNN overview



# GNN overview



# GNN overview



# Outline

1. Introduction
2. Course logistics
3. Applied topics
  - a. Pointer networks for the traveling salesman problem (TSP)
  - b. Graph neural networks
    - i. **Greedy algorithms**
    - ii. Integer programming and SAT
  - c. Transformers and LLMs
4. Theoretical topics

# Greedy algorithms

Many graph problems are **NP-hard**

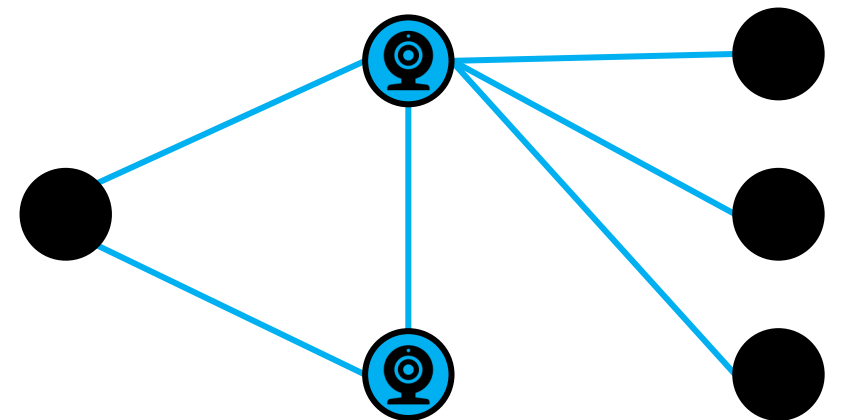
**Greedy algorithms:** a common type of fast heuristic algorithm

# Example: minimum vertex cover

Find smallest vertex subset such that each edge is covered

## **Example application:**

Installing cameras in corners covering all hallways on a floor

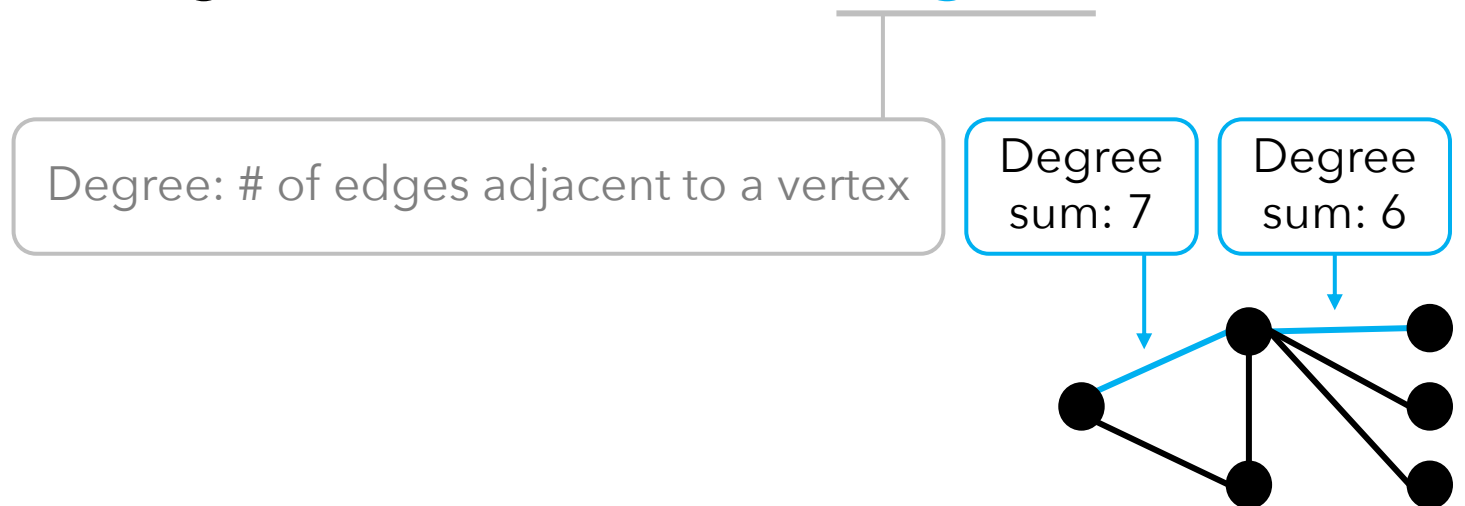


# Example: minimum vertex cover

Find smallest vertex subset such that each edge is covered

## Classic greedy algorithm:

Greedily add vertices of edge with **maximum degree sum**



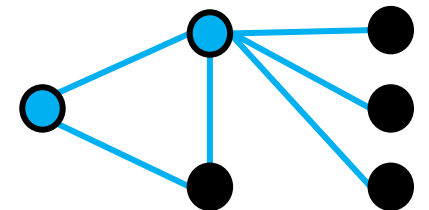
# Example: minimum vertex cover

Find smallest vertex subset such that each edge is covered

## Classic greedy algorithm:

Greedily add vertices of edge with maximum degree sum

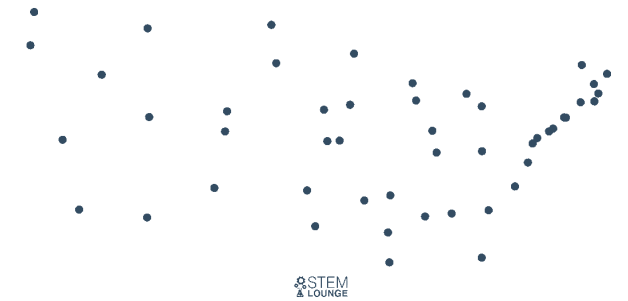
Scoring function that guides greedy algorithm





# Example: TSP

1. Among all cities not in the subtour:  
Choose the one that's farthest from any city in the subtour
  2. Insert it into the subtour
- Scoring function** that guides greedy algorithm

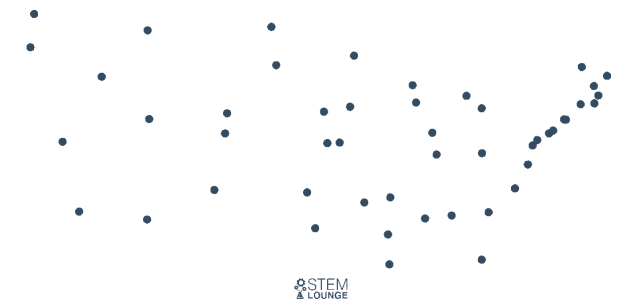


# RL for combinatorial optimization

**Goal: learn** a scoring function to guide greedy algorithm

Represented by a GNN

We'll see how to use *Q-learning* to train GNN



# ML as a toolkit for theory?

E.g., Dai et al. [NeurIPS'17] write that their RL alg discovered:  
“New and interesting” greedy strategies for MAXCUT and MVC  
“which **intuitively make sense** but have **not been analyzed** before,”  
thus could be a “good **assistive tool** for discovering new algorithms.”

# Outline

1. Introduction
2. Course logistics
3. Applied topics
  - a. Pointer networks for the traveling salesman problem (TSP)
  - b. Graph neural networks
    - i. Greedy algorithms
    - ii. Integer programming and SAT**
  - c. Transformers and LLMs
4. Theoretical topics

# Integer programming

## Integer program (IP)

$$\max \mathbf{c} \cdot \mathbf{z}$$

$$\text{s.t. } \mathbf{Az} \leq \mathbf{b}$$

$$\mathbf{z} \in \mathbb{Z}^n$$

**Tons** of applications:



Routing



Manufacturing



Scheduling



Planning

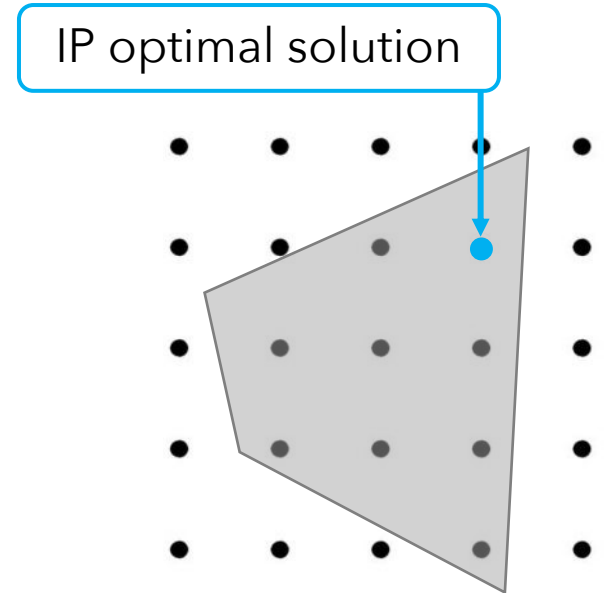


Finance

# Integer programming

## Integer program (IP)

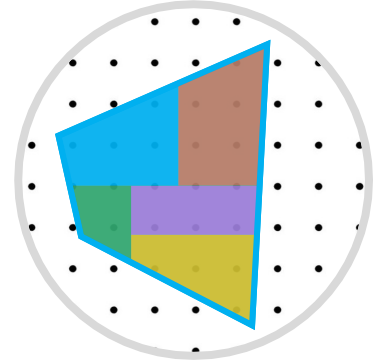
$$\begin{aligned} \max \quad & \mathbf{c} \cdot \mathbf{z} \\ \text{s.t.} \quad & A\mathbf{z} \leq \mathbf{b} \\ & \mathbf{z} \in \mathbb{Z}^n \end{aligned}$$



# Integer programming

Recursively partitions feasible region

Partition organized with a **tree data structure**



Partitioning policy has a big impact on runtime

**Goal:** Use a GNN to guide tree search

*Large improvements over leading open-source solver*



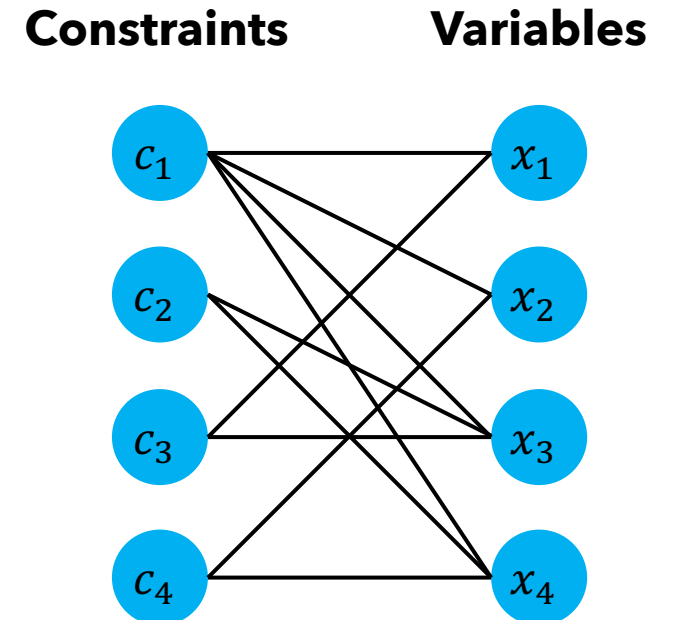
# GNNs for integer programming

**Key insight:** IP can be encoded as a **graph**

$$\begin{aligned} \max \quad & 9x_1 + 5x_2 + 6x_3 + 4x_4 \\ \text{s.t.} \quad & 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 && (c_1) \\ & x_3 + x_4 \leq 10 && (c_2) \\ & -x_1 + x_3 \leq 0 && (c_3) \\ & -x_2 + x_4 \leq 0 && (c_4) \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned}$$

**Goal:** Use a GNN to guide tree search

*Large improvements over leading open-source solver*





# SAT

$$\begin{aligned} & (x_1 \vee x_4) \\ \wedge & (x_1 \vee \bar{x}_3 \vee \bar{x}_8) \\ \wedge & (x_1 \vee x_8 \vee x_{12}) \\ \wedge & (x_2 \vee x_{11}) \\ \wedge & (\bar{x}_7 \vee \bar{x}_3 \vee x_9) \\ \wedge & (\bar{x}_7 \vee x_8 \vee \bar{x}_9) \\ \wedge & (x_7 \vee x_8 \vee \bar{x}_{10}) \\ \wedge & (x_7 \vee x_{10} \vee \bar{x}_{12}) \end{aligned}$$

**SAT:** Is there an assignment of  $x_1, \dots, x_{12} \in \{0,1\}$  such that this formula evaluates to **True**?

Published as a conference paper at ICLR 2019

## LEARNING A SAT SOLVER FROM SINGLE-BIT SUPERVISION

**Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, David L. Dill**  
Department of Computer Science  
Stanford University  
Stanford, CA 94305  
{dselsam, mlamm, buenz, pliang, dill}@cs.stanford.edu

**Leonardo de Moura**  
Microsoft Research  
Redmond, WA 98052  
leonardo@microsoft.com

# Topics

- Graph fundamentals and famous graph algorithms
- GNNs
- Reinforcement learning (Q-learning)
- Integer programming and SAT solving

# Outline

1. Introduction
2. Course logistics
3. Applied topics
  - a. Pointer networks for the traveling salesman problem (TSP)
  - b. Graph neural networks
  - c. Integer programming
  - d. Transformers and LLMs**
4. Theoretical topics

# Integer programming

$$\begin{aligned} \max \quad & \mathbf{c} \cdot \mathbf{z} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{z} \leq \mathbf{b} \\ & \mathbf{z} \in \mathbb{Z}^n \end{aligned}$$

- Formulating a problem as an IP can be hard
- Requires a lot of domain expertise

**Tons** of applications:



Routing



Manufacturing



Scheduling



Planning



Finance

# LLMs for discrete modeling

$$\begin{array}{ll} \max & \mathbf{c} \cdot \mathbf{z} \\ \text{s.t.} & \mathbf{A}\mathbf{z} \leq \mathbf{b} \\ & \mathbf{z} \in \mathbb{Z}^n \end{array}$$

- Formulating a problem as an IP can be hard
- Requires a lot of domain expertise
- How to use LLMs to for discrete modeling



Guest lecture by  
Madeleine Udell

# Topics

- Transformers
- Transformers as algorithms  
*E.g., can you teach a transformer to add?*
- LLMs for discrete modeling

# Outline

1. Introduction
2. Course logistics
3. Applied topics
- 4. Theoretical topics**
  - i. Statistical guarantees and online algorithm configuration**
  - ii. Algorithms with predictions
5. Plan for the next 2 weeks

# Algorithm configuration

**Example:** IP solvers (CPLEX, Gurobi) have a **ton** parameters

What's the best **configuration** for the application at hand?



Best configuration for **routing** problems  
likely not suited for **scheduling**





# Modeling the application domain

Problem instances drawn from application-specific dist.  $\mathcal{D}$



E.g., **distribution over routing problems**

Widely assumed in applied research, e.g.:

Horvitz, Ruan, Gomez, Kautz, Selman, Chickering

Xu, Hutter, Hoos, Leyton-Brown

He, Daumé, Eisner

UAI'01

JAIR'08

NeurIPS'14

And theoretical research on algorithm configuration, e.g.:

Gupta, Roughgarden

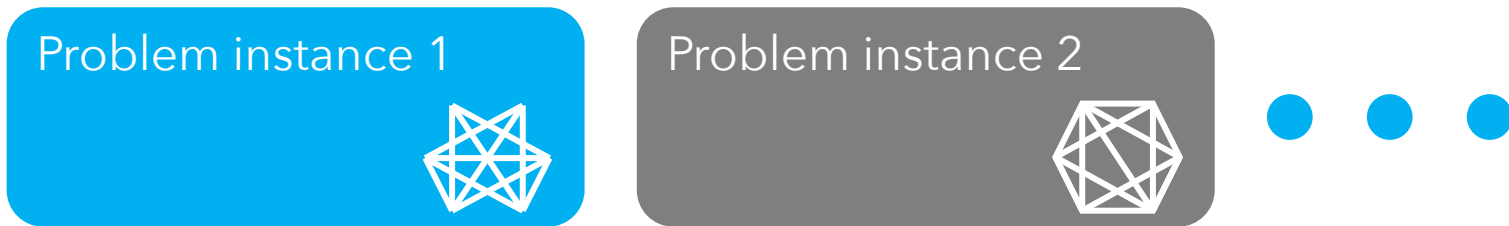
Balcan

ITCS'16

Book Chapter'20

# Automated configuration procedure

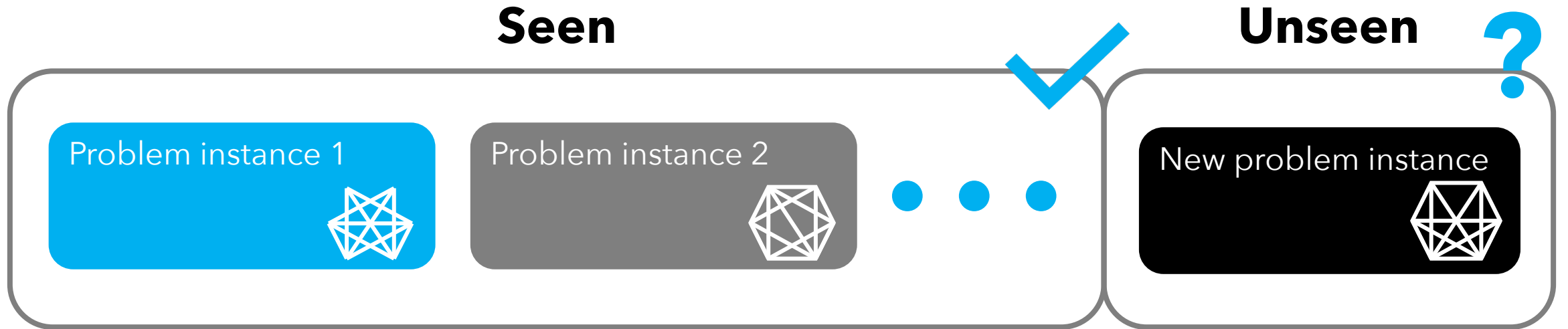
1. Fix parameterized algorithm
2. Receive set of "typical" inputs sampled from unknown  $\mathcal{D}$



3. Return parameter setting  $\hat{\rho}$  with good avg performance

Runtime, solution quality, etc.

# Automated configuration procedure

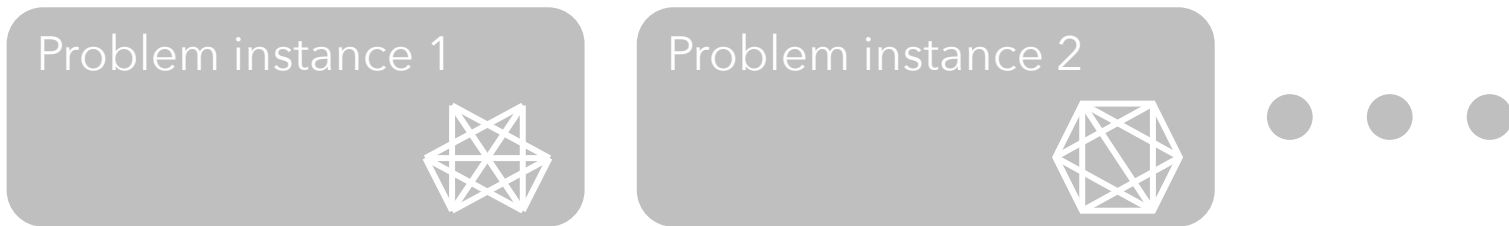


**Statistical question:** Will  $\hat{\rho}$  have good **future** performance?

**More formally:** Is the expected performance of  $\hat{\rho}$  also good?

# Automated configuration procedure

1. Fix parameterized algorithm
2. Receive set of "typical" inputs sampled from unknown  $\mathcal{D}$



3. Return parameter setting  $\hat{\rho}$  with good avg performance

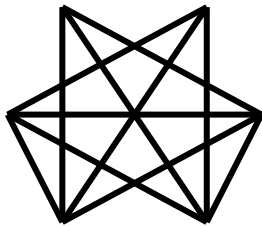
Runtime, solution quality, etc.

Model is known as the "**batch-learning** setting"  
*Optimize over a **batch** of input problem instances*

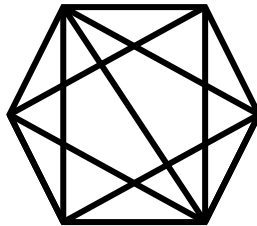
# Online algorithm configuration

What if inputs are not i.i.d., but even adversarial?

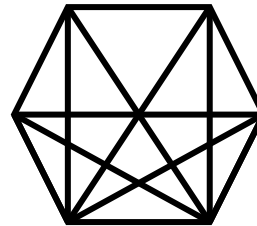
Day 1:  $\rho_1$



Day 2:  $\rho_2$



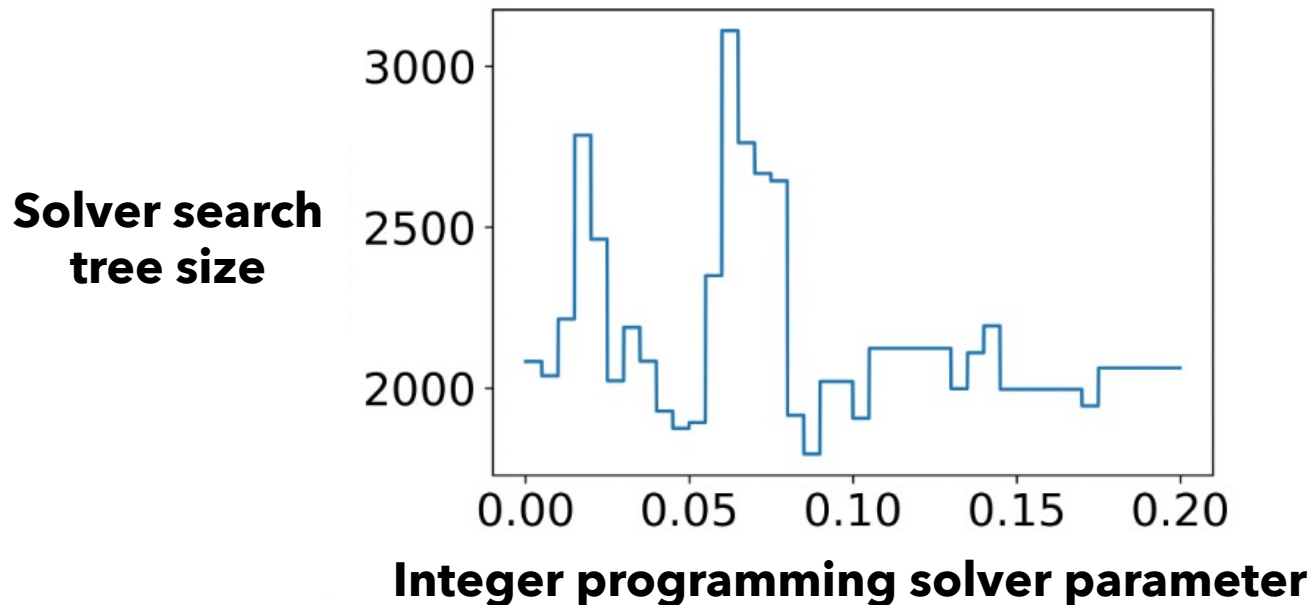
Day 3:  $\rho_3$



- Goal:** Compete with best parameter setting in hindsight
- Impossible in the worst case
  - Under what conditions is online configuration possible?

# Primary challenge

Algorithmic performance is a **volatile** function of parameters  
**Complex** connection between parameters and performance



# Topics

- Statistical learning theory
- Online learning

# Outline

1. Introduction
2. Course logistics
3. Applied topics
4. Theoretical topics
  - i. Statistical guarantees and online algorithm configuration
  - ii. Algorithms with predictions**



# Algorithms with predictions

Assume you have some **predictions** about your problem, e.g.:



Probability any given element is in a huge database

Kraska et al., SIGMOD'18; Mitzenmacher, NeurIPS'18

In caching, the next time you'll see an element

Lykouris, Vassilvitskii, ICML'18

## **Main question:**

How to use predictions to improve algorithmic performance?

# Example: Ski rental problem

- **Problem:** Skier will ski for unknown number of days
  - Can either **rent each day** for \$1/day or **buy** for \$ $b$
  - E.g., if ski for 5 days and then buy, total price is  $5 + b$
- If ski  $x$  days, **opt clairvoyant** strategy pays  $OPT = \min\{x, b\}$
- **Breakeven strategy:** Rent for  $b - 1$  days, then buy

$$CR = \frac{ALG}{OPT}$$

Competitive ratio



# Example: Ski rental problem

Prediction  $y$  of number of skiing days, error  $\eta = |x - y|$

**Algorithm** (with parameter  $\lambda \in (0,1)$ ):

If  $y \geq b$ , buy on start of day  $\lfloor \lambda b \rfloor$ ; else buy on start of day  $\lceil \frac{b}{\lambda} \rceil$

*Don't jump the gun...*

*...but don't wait too long*

**Theorem:** Algorithm has  $\text{CR} \leq \min \left\{ \frac{1+\lambda}{\lambda}, 1 + \lambda + \frac{\eta}{(1-\lambda)\text{OPT}} \right\}$

- If predictor is perfect ( $\eta = 0$ ), **CR is small** ( $\leq 1 + \lambda$ )
- No matter how big  $\eta$  is, setting  $\lambda = 1$  **recovers baseline**  $\text{CR} = 2$

# Design principals

## Consistency:

Predictions are perfect  $\Rightarrow$  recover offline optimal



## Robustness:

Predictions are terrible  $\Rightarrow$  no worse than worst-case

# Many different applications

## Online advertising

Mahdian, Nazerzadeh, Saberi, EC'07;  
Devanur, Hayes, EC'09; Medina,  
Vassilvitskii, NeurIPS'17; ...

## Caching

Lykouris, Vassilvitskii, ICML'18; Rohatgi,  
SODA'19; Wei, APPROX-RANDOM'20; ...

## Frequency estimation

Hsu, Indyk, Katabi, Vakilian, ICLR'19; ...

## Learning low-rank approximations

Indyk, Vakilian, Yuan, NeurIPS'19; ...

## Scheduling

Mitzenmacher, ITCS'20; Moseley,  
Vassilvitskii, Lattanzi, Lavastida, SODA'20; ...

## Matching

Antoniadis, Gouleakis, Kleeer, Kolev,  
NeurIPS'20; ...

## Queuing

Mitzenmacher, ACDA'21; ...

## Covering problems

Bamas, Maggiori, Svensson, NeurIPS'20; ...

[algorithms-with-predictions.github.io](https://algorithms-with-predictions.github.io)

# Outline

1. Applied topics
  - a. Pointer networks for the traveling salesman problem (TSP)
  - b. Graph neural networks
  - c. Transformers and LLMs
  
2. Theoretical topics
  - a. Statistical guarantees and online algorithm configuration
  - b. Algorithms with predictions

**Looking forward to getting to know you this quarter!**