

Welcome to
**Machine Learning for
Algorithm Design!**

About me



Ellen Vitercik

Assistant Professor at Stanford

*Management Science & Engineering
Computer Science*

Research revolves around

- Machine learning for algorithm design
- 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 **algorithm design**?

O **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 **algorithm design**?

O **Algorithm configuration**

How to tune an algorithm's parameters?

O **Algorithm selection**

Given a variety of algorithms, which to use?

O **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_BTTLIM 40
CPX_PARAM_NODESEL 102	CPX_PARAM_TUNINGDISPLAY 162	CPX_PARAM_REINV 131	CPXPARAM_MIP_Strategy_Branch 39	CPX_PARAM_FPHEUR 72	CPX_PARAM_CALCQCPDUALS 41
CPX_PARAM_NUMERICALPHASIS 102	CPX_PARAM_TUNINGMEASURE 163	CPX_PARAM_RELAXPREIND 132	CPXPARAM_MIP_Strategy_MIQCPStrat 93	CPX_PARAM_FRACAND 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_OBLLIM 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_CPUTMASK 48
CPX_PARAM_POLISHAFTERDETTIME 111	CPXPARAM_Benders_Strategy 30	CPX_PARAM_SCRIND 143	CPXPARAM_Preprocessing_Linear 120	CPX_PARAM_ITLIM 80	CPX_PARAM_CRAIND 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_SIFTITLIM 145	CPXPARAM_Read_DataCheck 54	CPX_PARAM_LPMETHOD 136	CPX_PARAM_CUTSFATOR 52
CPX_PARAM_POLISHAFTERNODE 115	CPXPARAM_CPUmask 48	CPX_PARAM_SIMDISPLAY 145	CPXPARAM_Read_Scale 142	CPX_PARAM_MCPFCUTS 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	CPX_PARAM_DATACHECK 54
CPX_PARAM_POLISHTIME (deprecated) 116	CPXPARAM_LPMETHOD 136	CPX_PARAM_SOLNPOOLAGAP 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_LocalImpplied 77	CPX_PARAM_SOLNPOOLGAP 148	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_PRELINEAR 120	CPXPARAM_MIP_Limits_CutsFactor 52	CPX_PARAM_SOLUTIONTARGET	CPXPARAM_SolutionType 152	CPX_PARAM_MIPORDIND 90	CPX_PARAM_EACHCUTLIM 60
CPX_PARAM_PREPASS 121	CPXPARAM_MIP_Limits_RampupDefTimeLimit 127	deprecated: see CPXPARAM_OptimalityTarget 106	CPXPARAM_Threads 157	CPX_PARAM_MIPORDTYPE 91	CPX_PARAM_EPAGAP 61
CPX_PARAM_PRESLVND 122	CPXPARAM_MIP_Limits_Solutions 79	CPX_PARAM_SOLUTIONTYPE 152	CPXPARAM_TimeLimit 159	CPX_PARAM_MIPSEARCH 92	CPX_PARAM_EPGAP 61
CPX_PARAM_PRICELIM 123	CPXPARAM_MIP_Limits_StrongCand 154	CPX_PARAM_STARTALG 139	CPXPARAM_Tune_DefTimeLimit 160	CPX_PARAM_MIQCPSTRAT 93	CPX_PARAM_EPINT 62
CPX_PARAM_PROBE 123	CPXPARAM_MIP_Limits_StrongIt 154	CPX_PARAM_STRONGCANDLIM 154	CPXPARAM_Tune_Display 162	CPX_PARAM_MIRCUTS 94	CPX_PARAM_EPMRK 64
CPX_PARAM_PROBEDETTIME 124	CPXPARAM_MIP_Limits_TreeMemory 160	CPX_PARAM_STRONGITLIM 154	CPXPARAM_Tune_Measure 163	CPX_PARAM_MPSSLONGNUM 94	CPX_PARAM_EPOPT 65
CPX_PARAM_PROBTIME 124	CPXPARAM_MIP_OrderType 91	CPX_PARAM_SUBALG 99	CPXPARAM_Tune_Repeat 164	CPX_PARAM_NETDISPLAY 95	CPX_PARAM_EPPER 65
CPX_PARAM_QPMAKEPSDIND 125	CPXPARAM_MIP_Pool_AbsGap 146	CPX_PARAM_SUBMIPNODELIMIT 155	CPXPARAM_Tune_TimeLimit 165	CPX_PARAM_NETEPOPT 96	CPX_PARAM_EPRELAX 66
CPX_PARAM_QPMETHOD 138	CPXPARAM_MIP_Pool_Capacity 147	CPX_PARAM_SYMMETRY 156	CPXPARAM_WorkDir 167	CPX_PARAM_NETEPRS 96	CPX_PARAM_EPRIHS 67
CPX_PARAM_QPNZREADLIM 126	CPXPARAM_MIP_Pool_Intensity 149	CPX_PARAM_THREADS 157	CPXPARAM_WorkMem 168	CPX_PARAM_NETFIND 97	CPX_PARAM_FEASOPTMODE 68
		CPX_PARAM_TILIM 159	CraInd 50	CPX_PARAM_NETITLIM 98	CPX_PARAM_FILEENCODING 69

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 **algorithm design**?

O **Algorithm configuration**

How to tune an algorithm's parameters?

O **Algorithm selection**

Given a variety of algorithms, which to use?

O **Algorithm design**

Can machine learning guide algorithm discovery?

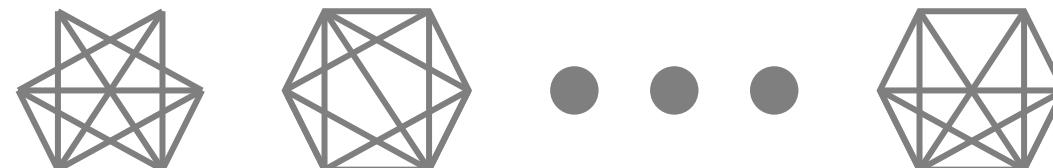
Algorithm selection in theory

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

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

In practice:

Instances solved in **past** are similar to **future** instances...



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



Routing problems a shipping company solves

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



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



Scheduling problems an airline solves

Course topics

Range of techniques for integrating ML into algorithm design

1. Applied topics

- i. Graph neural networks
- ii. Integer programming and SAT
- iii. Reinforcement learning
- iv. Data structures

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
5. Plan for the next 2 weeks

Course overview

Website: vitercik.github.io/ml4algs

Office hours:

- Tuesday 11am-12pm in Huang 250
- Or by appointment, please feel free to reach out!

Course setup

1. Lectures given by the instructor

- Key techniques for integrating ML into algorithm design
- *E.g., graph neural networks, reinforcement learning, theoretical ML*

2. Discussions led by students

- Covering influential papers in the field

Discussion led by students

- 10 paper discussion classes
- Each student will take on a **presenter role** for 5 discussions
 - Archaeologist
 - Researcher
 - Industry R&D expert
 - Private investigator
 - NeurIPS reviewer
 - (Based on a course design by [Alec Jacobson and Colin Raffel](#))
- (Students may need to pair up depending on class size)

Discussion led by students

- Presentations will be approximately 7 minutes + 5 min Q&A
- I'll distribute a Google spreadsheet next week to select roles

Presenter role: Archaeologist

- Determine where the paper sits in the context of previous and subsequent work
- Find and report on:
 1. One older paper cited by the current paper, and
 2. One newer paper citing this current paper



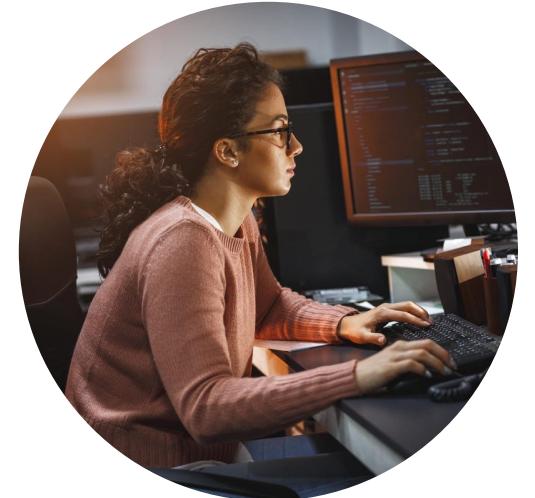
Presenter role: Researcher

- Propose a follow-up project on the current paper
 - Should only be possible due to the paper's existence and success



Presenter role: Industry R&D expert

- Convince your industry bosses that it's worth your time and money to implement this paper into the company's pipeline
- Choose an appropriate company and product or application



Presenter role: Private investigator

- Find out background information on one of the paper authors
 - Where have they worked?
 - What did they study?
 - What previous projects might have led to working on this one?



Presenter role: NeurIPS reviewer

Answer the questions on the NeurIPS review form
Originality, quality, clarity, significance, etc.



Non-presenter assignment

By 1pm on the day of class, post to Ed discussion:

at least one question about the paper. E.g.

- Something you're confused about
- Something you'd like to hear discussed more

Course project

- All students will write a "mini-paper" as a final project
- Can be empirical, theoretical, or both

Project policies

- Encouraged to work in groups!
 - Up to 3 people (except with special permission)
- Groups of 2 should put twice as much work into the final project than for a sole-author project
 - Similarly for groups of 3
- Paper length for a final project write-up is $3 + n$ where n is the number of people in the group that worked on the project
 - Not including references or the contributions paragraph
- Required to include a “contributions” paragraph in final paper that concretely lists each author's contributions

Milestones

April 17-21: All groups meet with me to discuss project ideas

- Please come prepared with ideas/interests!
- Look out for an email about scheduling this meeting

May 5: Submit a progress report of 1-2 pages

- Describe your project and partial progress

May 11: Short presentation about a paper related to your project

June 6-8: Present your final project during class

June 12: Submit your final report

Grading

Out of 100 points:

- Discussion: 60 points
 - Each **presentation** is worth 10 points
 - Each **non-presenter assignment** is worth 2 points
- Project: 40 points
 - **Progress report**: 7 points
 - **Midterm presentation**: 8 points.
 - **Novelty**: 5 points
 - Project should propose something new (new application, method, perspective)
 - **Writing**: 10 points
 - Final paper should be readable and complete and situate itself among related work
 - **Final presentation**: 10 points
 - Final presentation should be clear and provide a solid picture of what you did

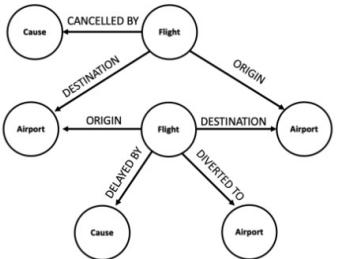
Prerequisites

- Introductory algorithms class
- Machine learning class is helpful but not required

Outline

1. Introduction
2. Course logistics
- 3. Applied topics**
 - i. **Graph neural networks**
 - ii. Integer programming and SAT
 - iii. Reinforcement learning
 - iv. Data structures
4. Theoretical topics
5. Plan for the next 2 weeks

Many types of data are graphs

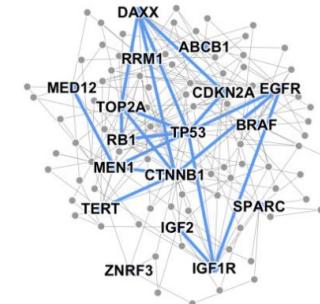


Event Graphs



Image credit: [SalientNetworks](#)

Computer Networks



Disease Pathways

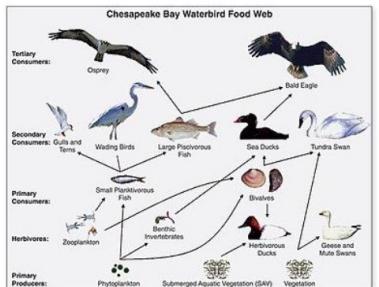


Image credit: [Wikipedia](#)

Food Webs



Image credit: [Pinterest](#)

Particle Networks



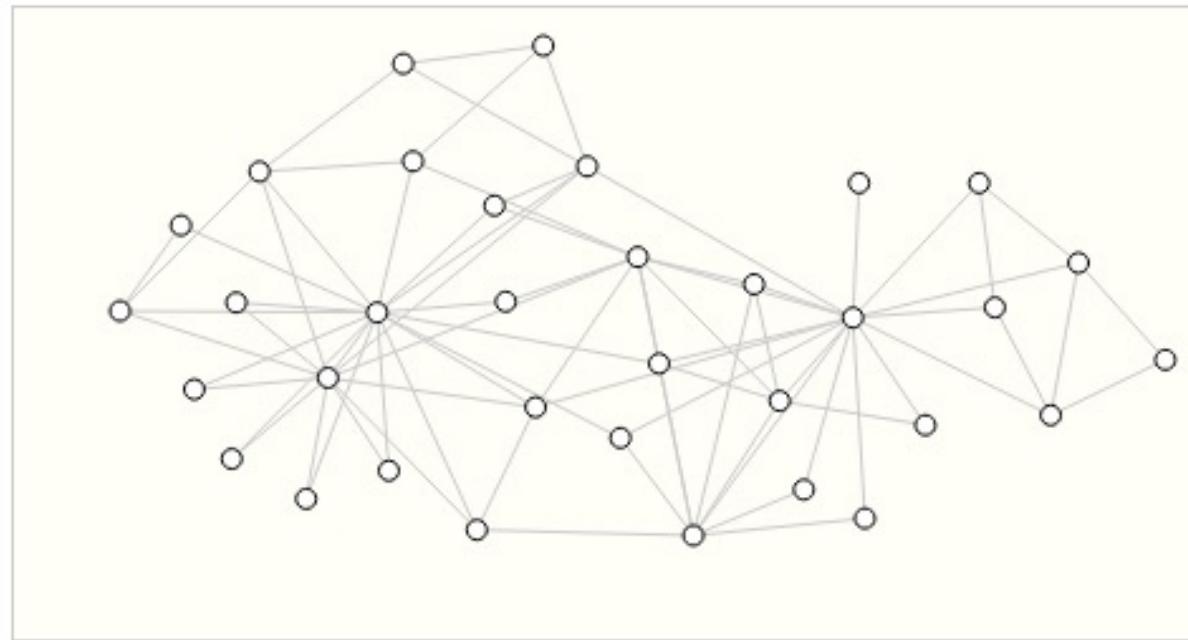
Image credit: [visitlondon.com](#)

Underground Networks

GNN motivation

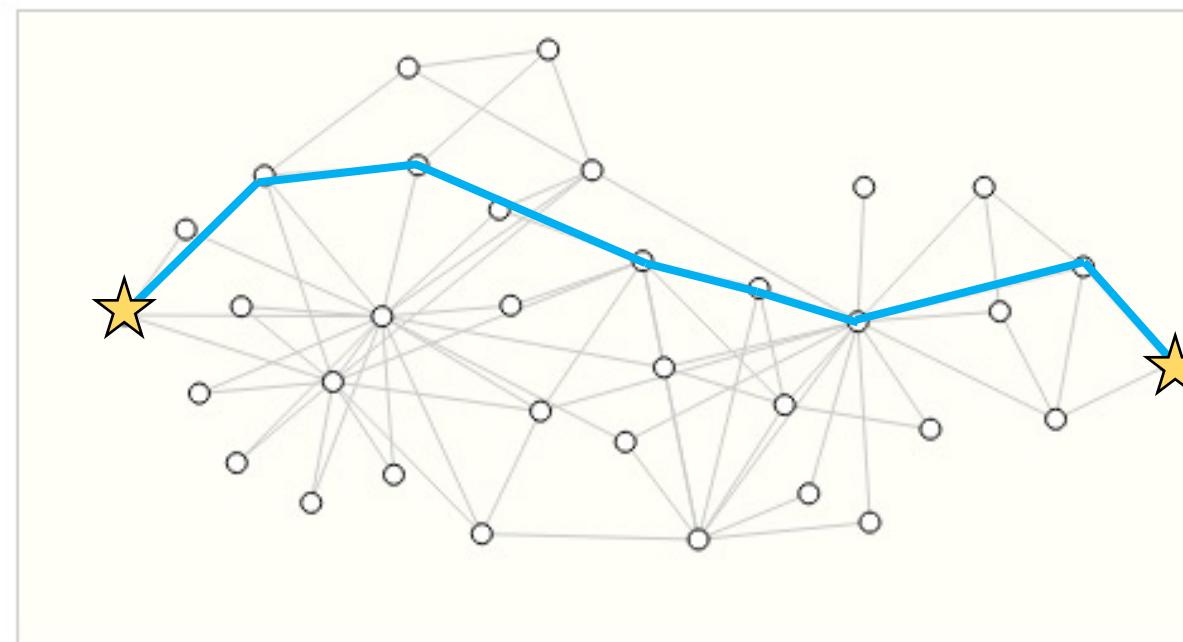
Special type of NN architecture for tasks involving graphs

How to utilize relational structure for better prediction?



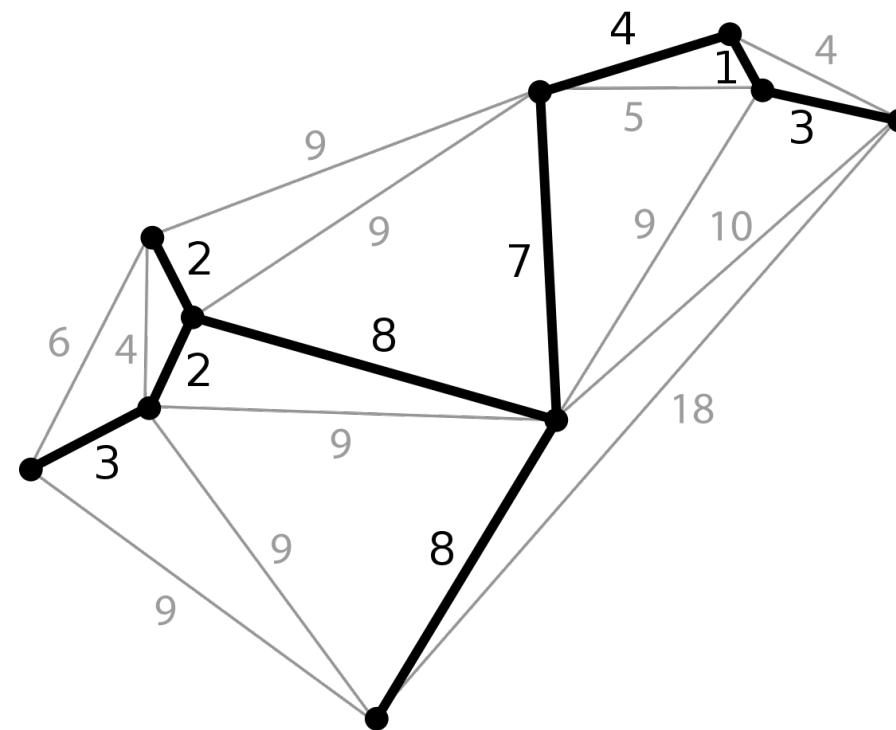
Shortest path prediction

Example: predicting the shortest path in a graph



MST prediction

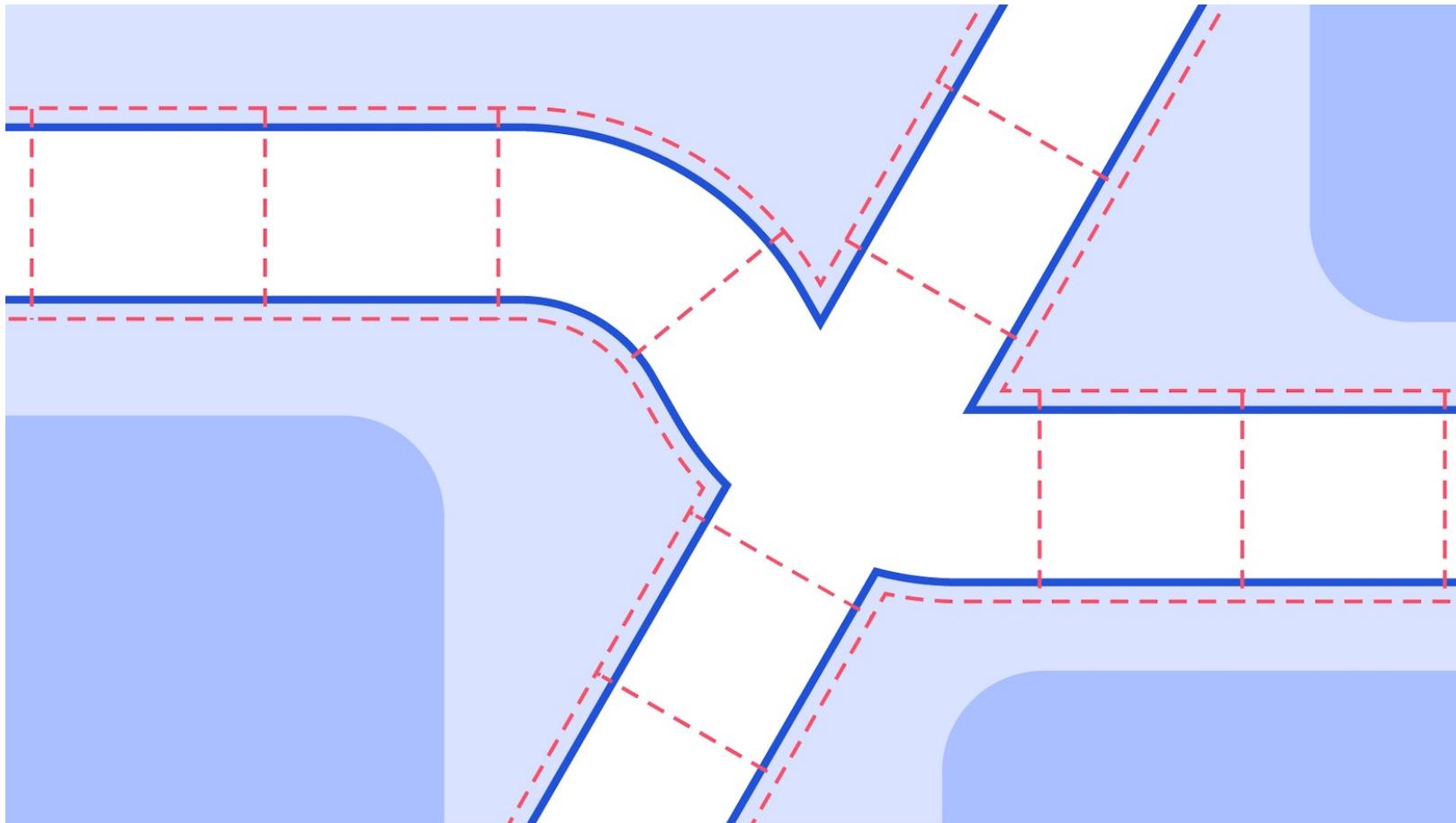
Example: predicting a minimum spanning tree



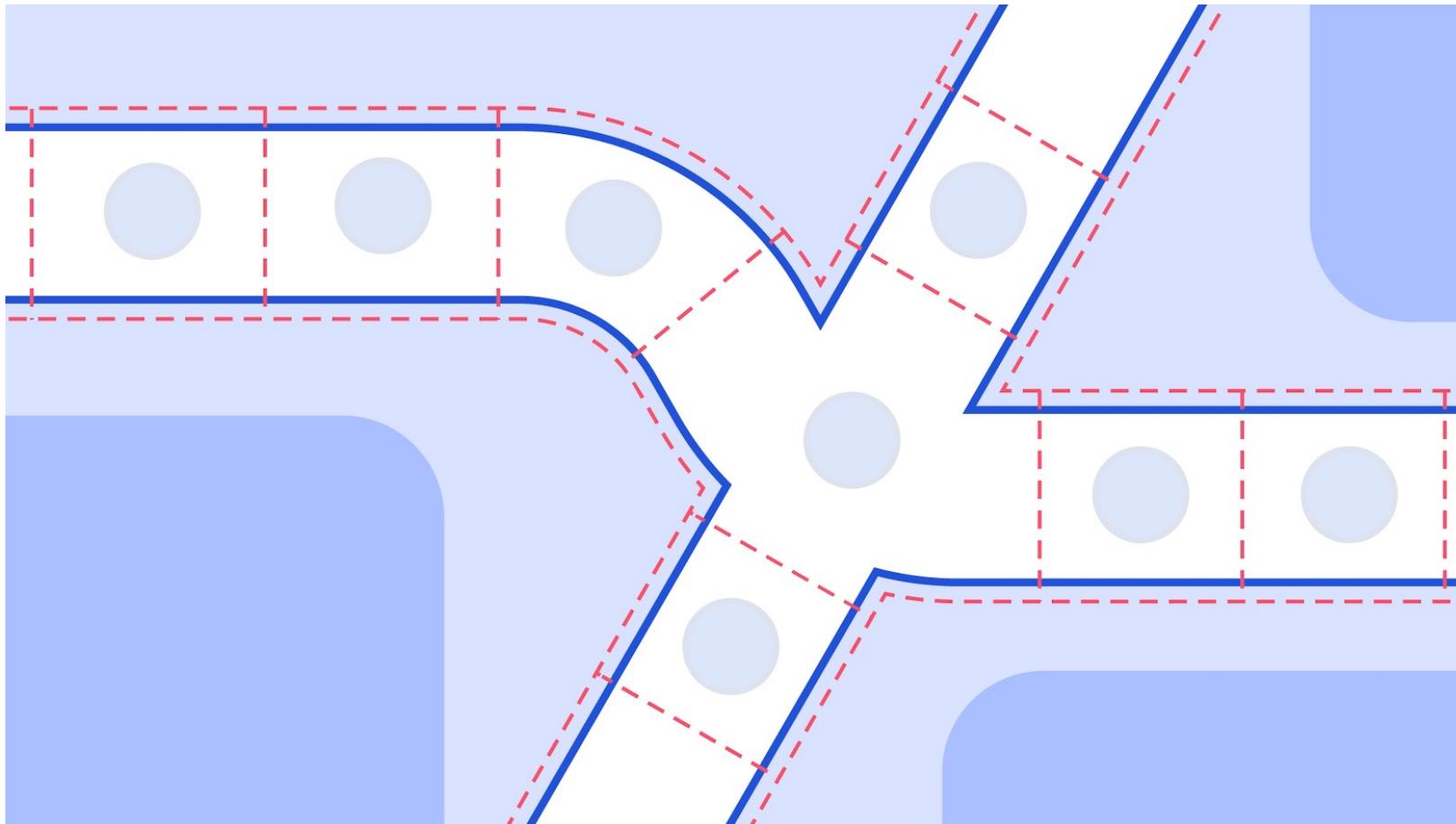
GNN: Message passing



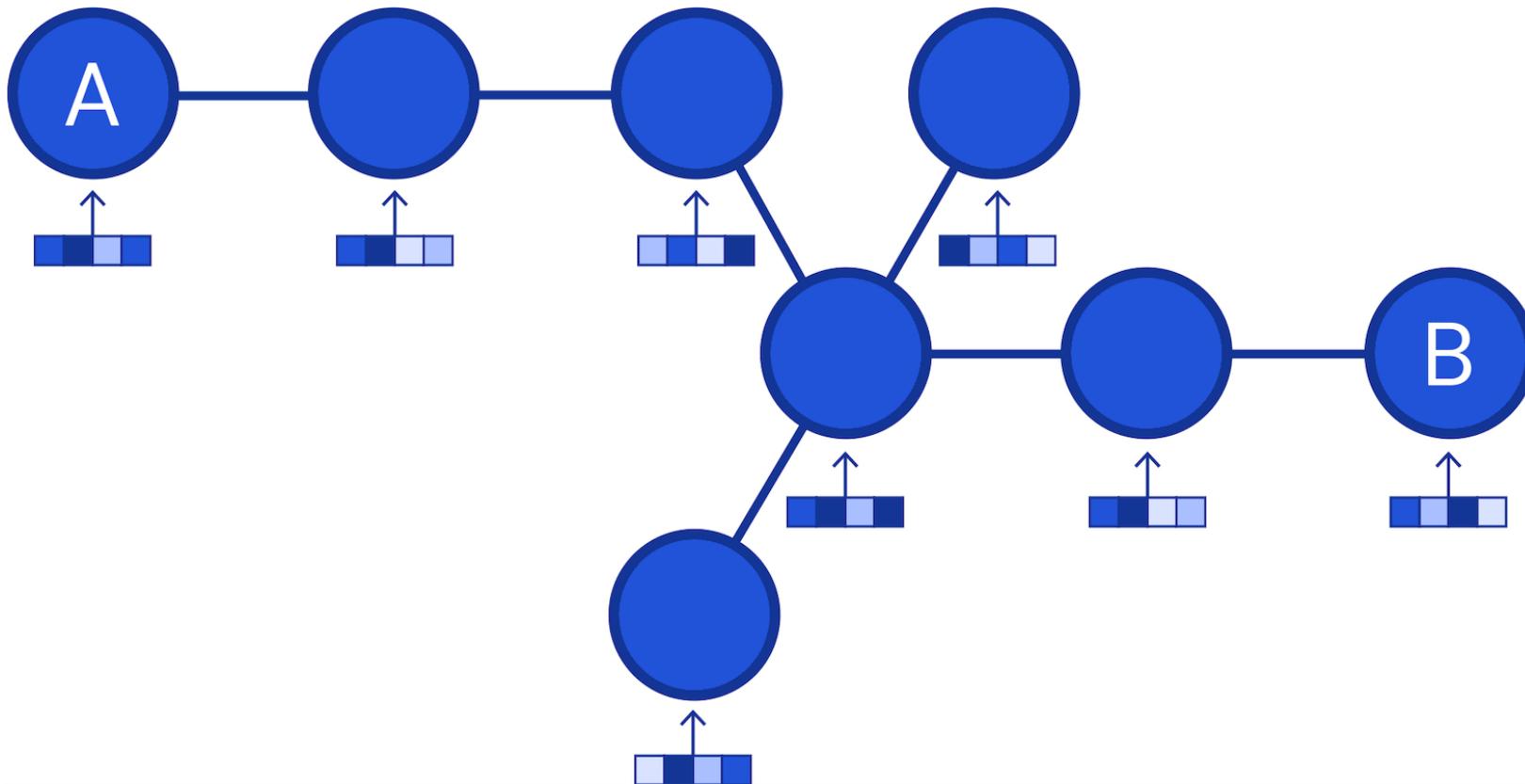
GNN: Message passing



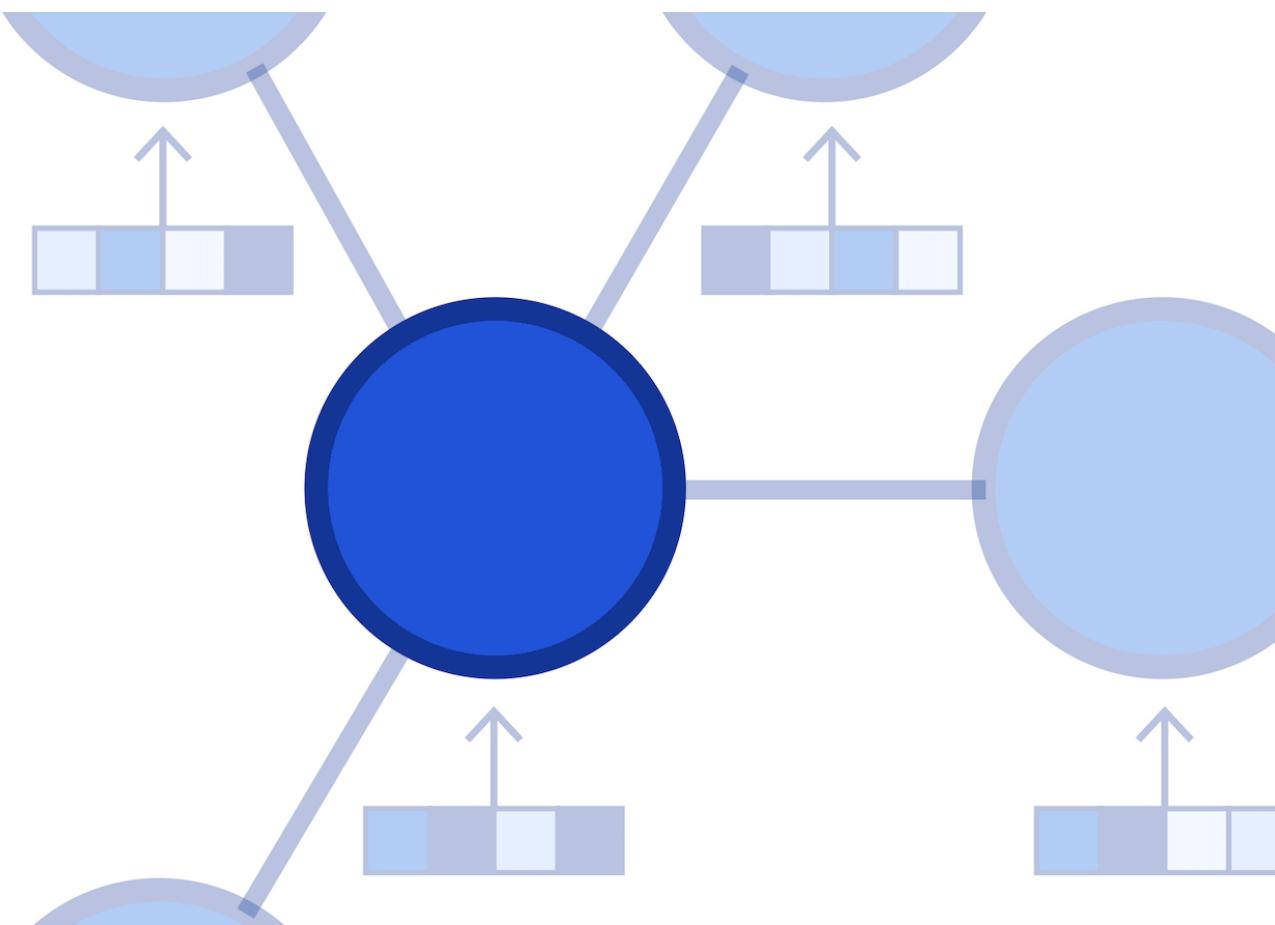
GNN: Message passing



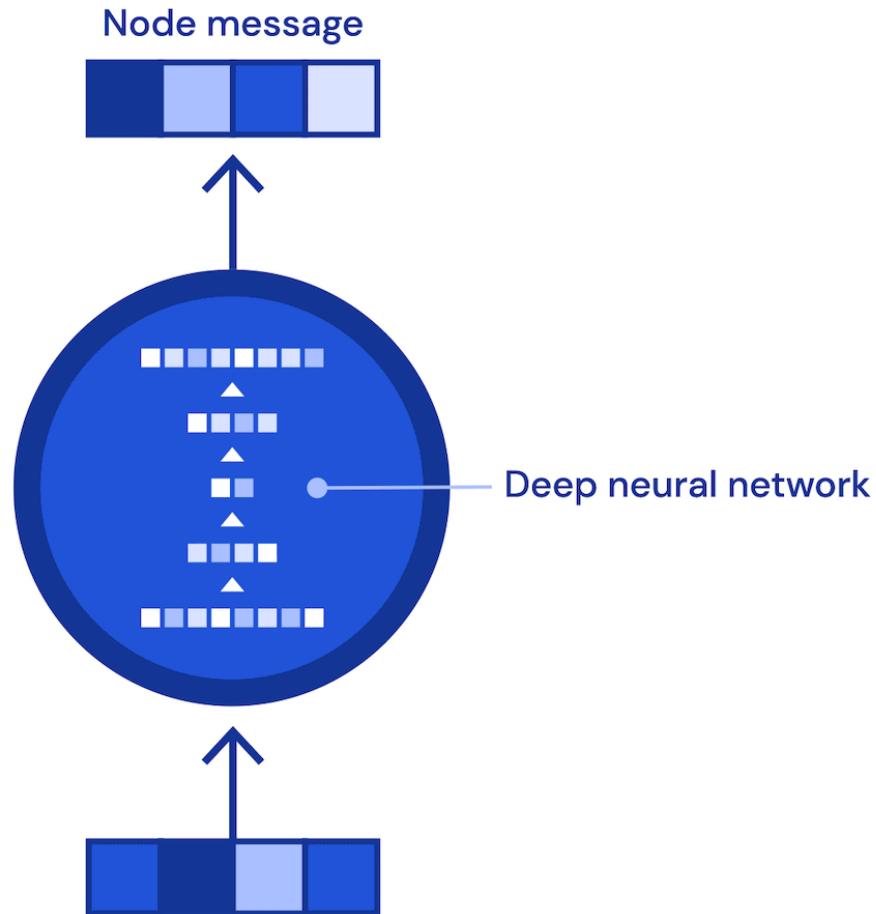
GNN: Message passing



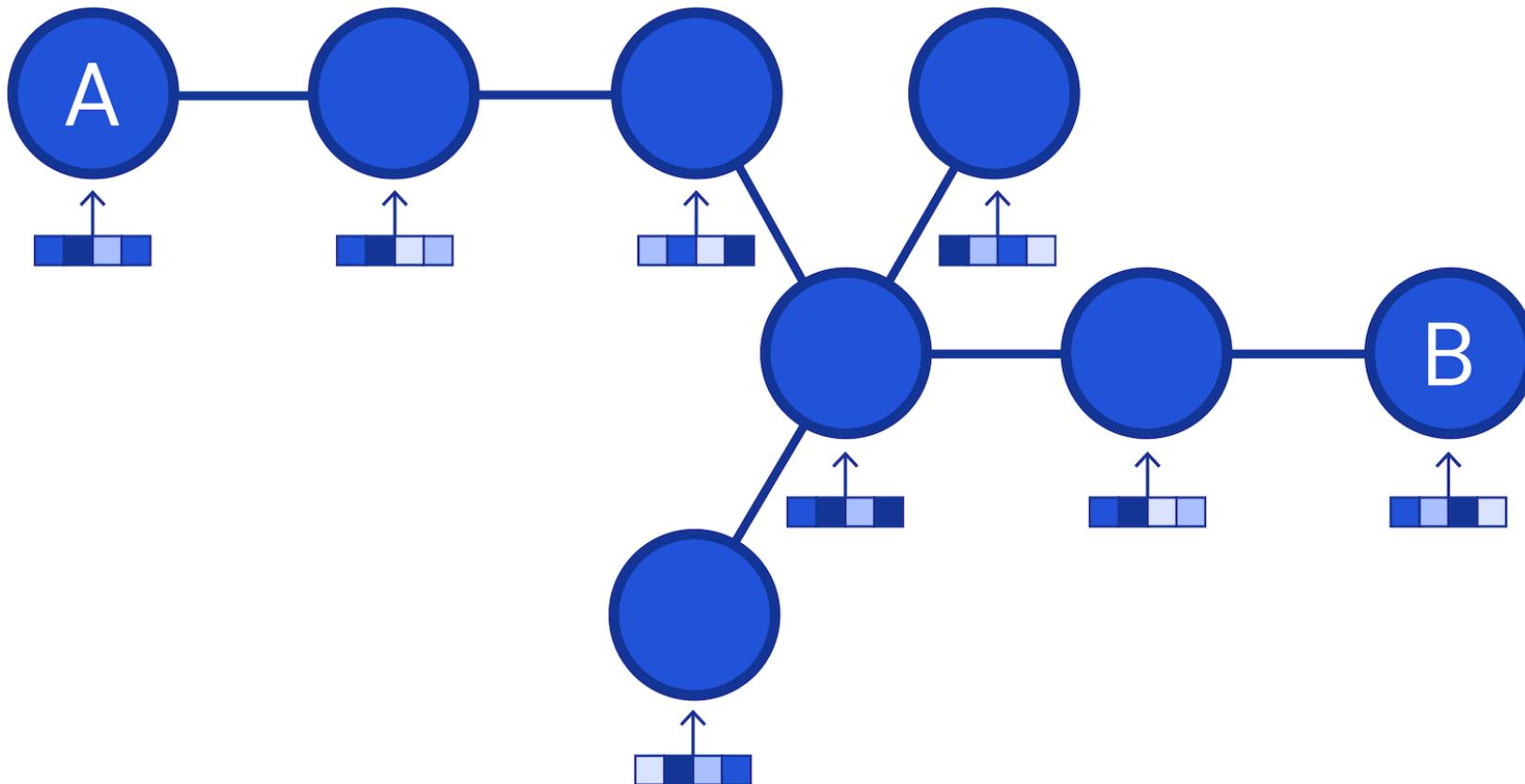
GNN: Message passing



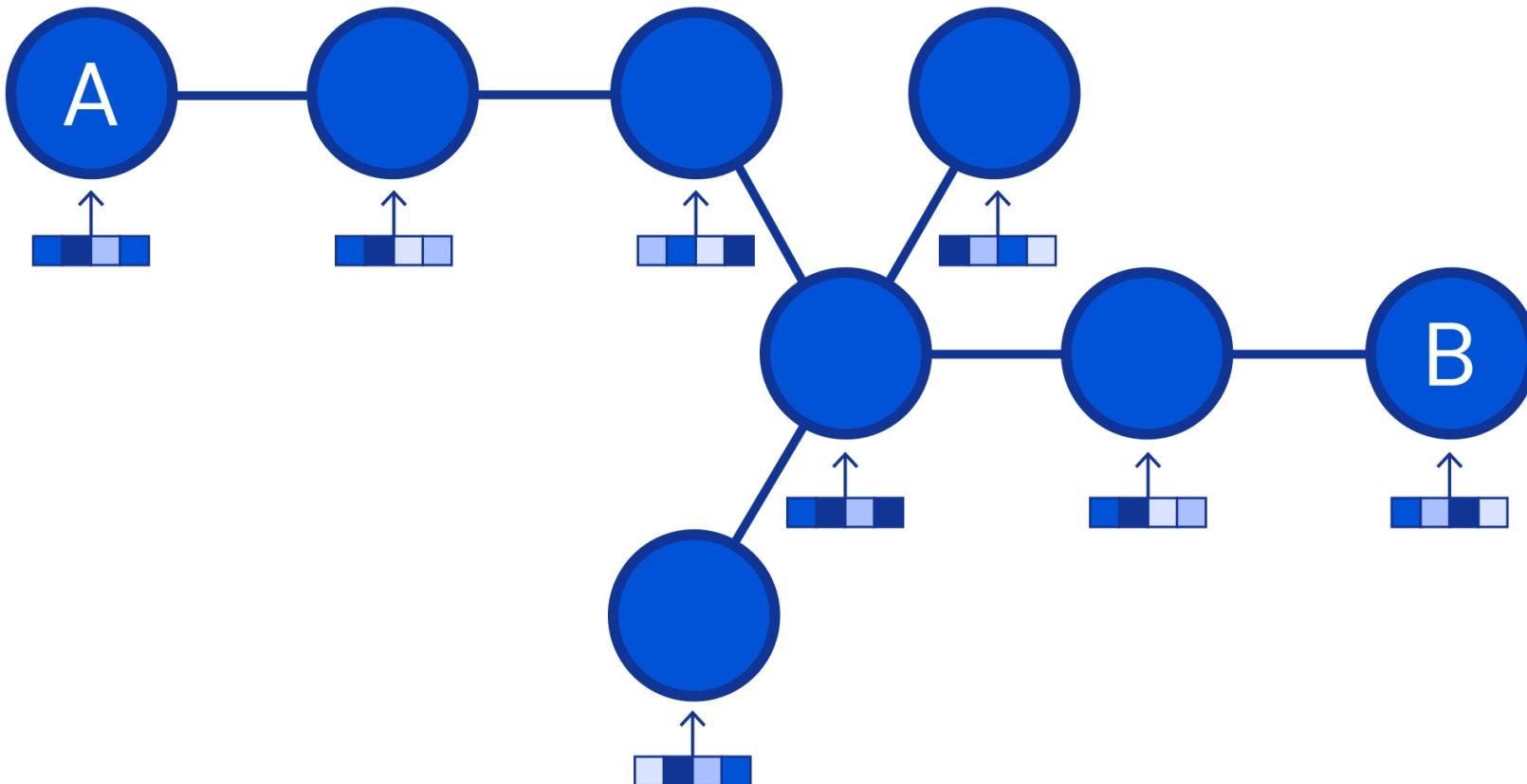
GNN: Message passing



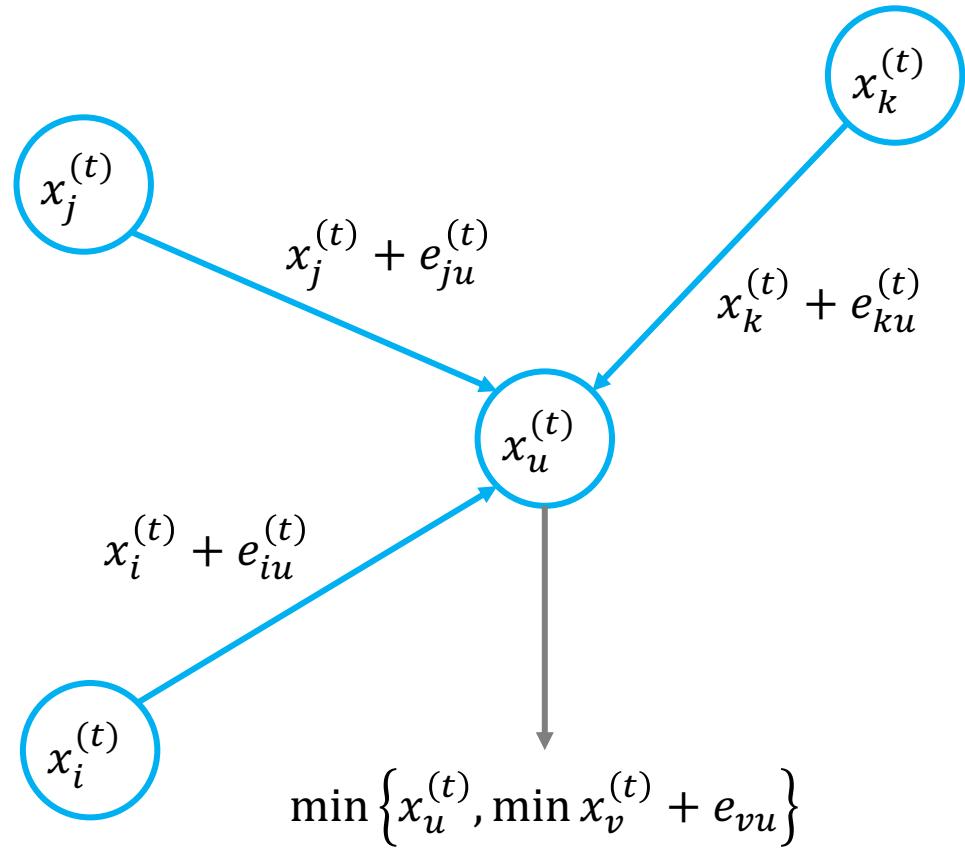
GNN: Message passing



GNN: Message passing

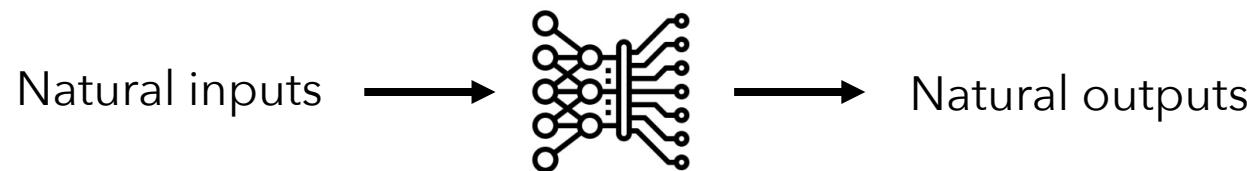


Bellman-Ford: Message passing



Why use GNNs for algorithm design?

- Classical algorithms are designed with abstraction in mind
 - Enforce their inputs to conform to stringent preconditions
- Challenges:
 - Natural inputs may be only partially observable
 - Manually converting natural inputs into abstract inputs leads to information loss
- **Goal:** end-to-end neural pipeline which is fully differentiable



Papers we'll read

Veličković, Petar, et al. "Neural execution of graph algorithms."
ICLR. 2020.

- GNNs don't work off-the-shelf for combinatorial tasks
- How to **align** GNN architectures to these tasks

Cappart, Quentin, et al. "Combinatorial optimization and reasoning with GNNs." *arXiv*.

- **Broad overview** of the field; current & future directions

Outline

1. Introduction
2. Course logistics
3. Applied topics
 - i. Graph neural networks
 - ii. Integer programming and SAT**
 - iii. Reinforcement learning
 - iv. Data structures
4. Theoretical topics
5. Plan for the next 2 weeks

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**?

Integer program

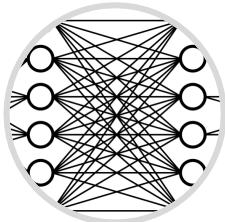
Integer program (IP)

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

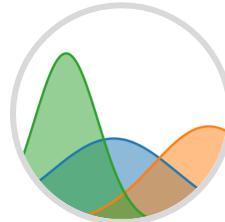
$$\text{s.t.} \quad A\mathbf{z} \leq \mathbf{b}$$

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

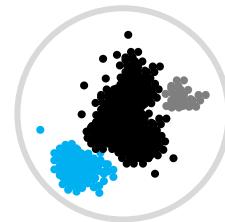
Tons of applications:



Robust ML



MAP estimation



Clustering



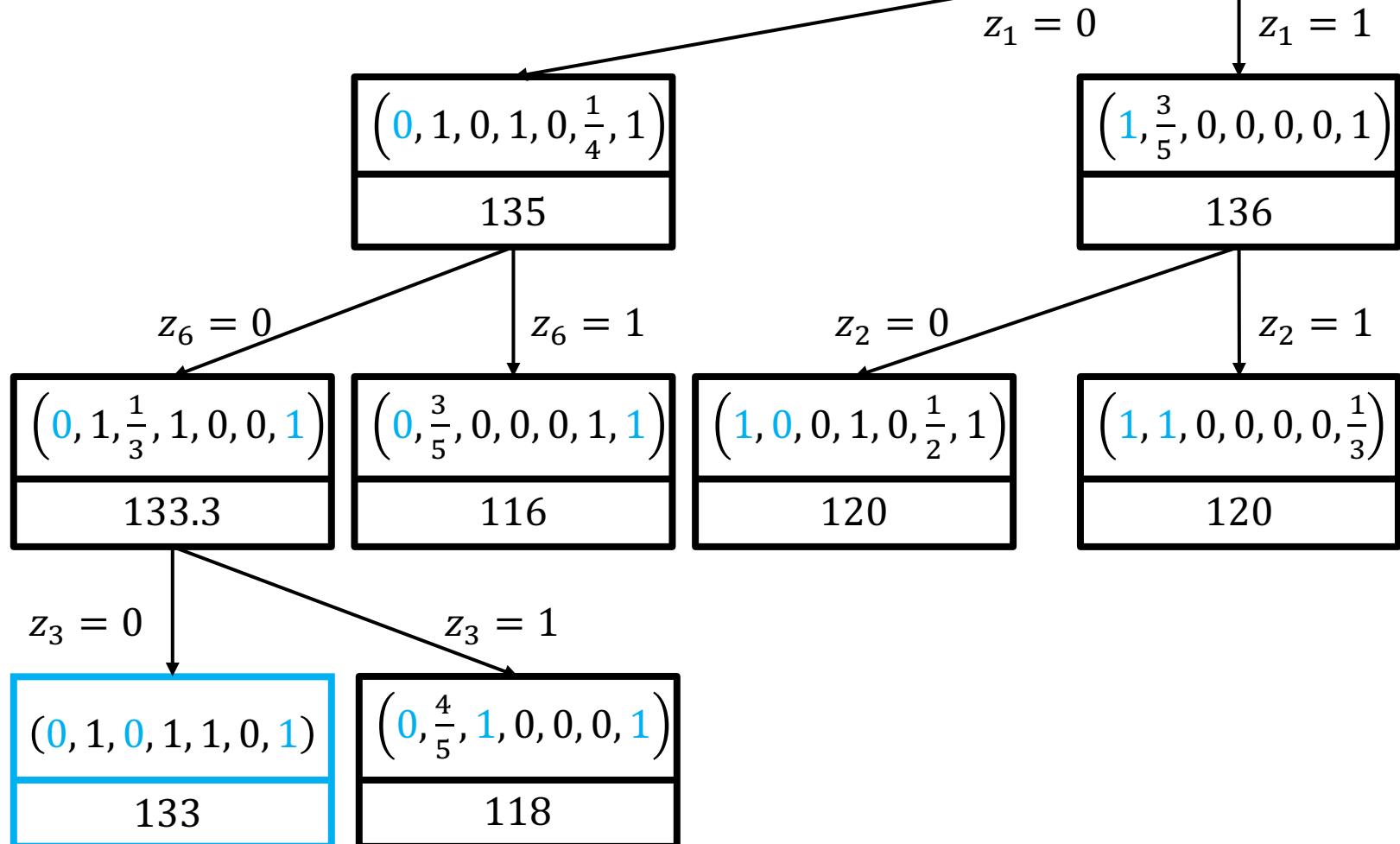
Routing



Scheduling

$$\begin{aligned}
 \max \quad & (40, 60, 10, 10, 3, 20, 60) \cdot \mathbf{z} \\
 \text{s.t.} \quad & (40, 50, 30, 10, 10, 40, 30) \cdot \mathbf{z} \leq 100 \\
 \mathbf{z} \in \{0,1\}^7
 \end{aligned}$$

$$\begin{array}{c}
 \mathbf{z} = \left(\frac{1}{2}, 1, 0, 0, 0, 0, 1 \right) \\
 \hline
 140
 \end{array}$$

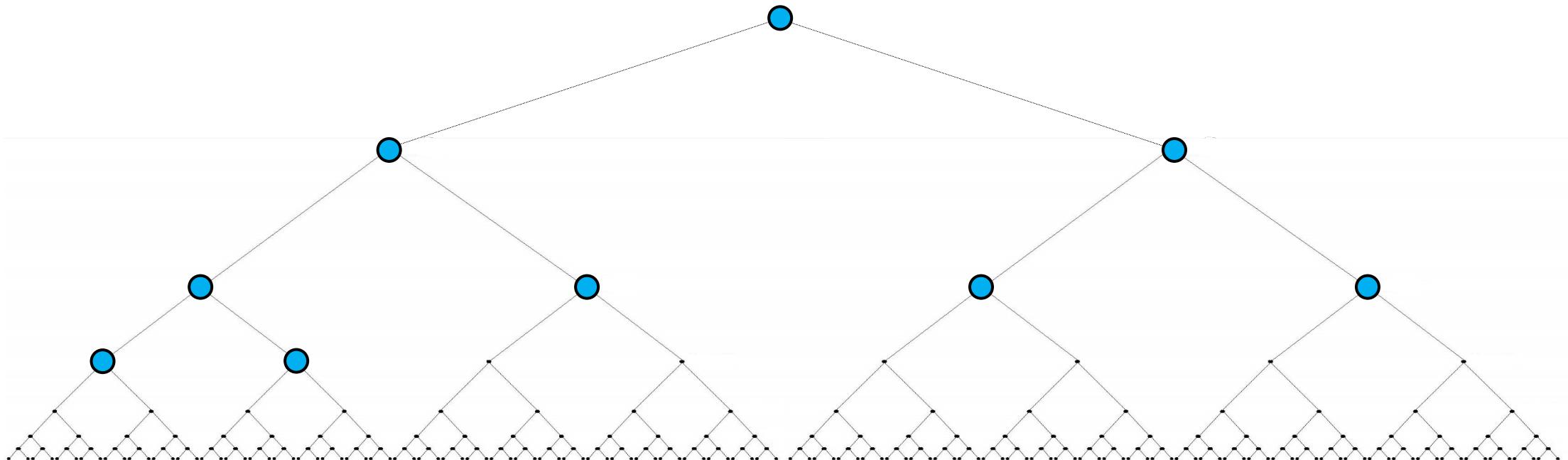


Branch and bound (B&B)

Tree-building policies

Tree-building policies can have a huge effect on tree size

E.g., node selection, variable selection,



Example: variable selection policies

Score-based variable selection policies:

At leaf Q , branch on variable z_i maximizing $\text{score}(Q, i) \in \mathbb{R}$

Many options! Little known about which to use when

Gauthier, Ribi  re, Math. Prog. '77; Beale, Annals of Discrete Math. '79; Linderoth, Savelsbergh, INFORMS JoC '99; Achterberg, Math. Prog. Computation '09; Gilpin, Sandholm, Disc. Opt. '11; ...

Example: variable selection policies

Score-based variable selection policies:

At leaf Q , branch on variable z_i maximizing $\mathbf{score}(Q, i) \in \mathbb{R}$

Given d scoring rules $\mathbf{score}_1, \dots, \mathbf{score}_d$, possible to
learn best convex combination $\rho_1 \mathbf{score}_1 + \dots + \rho_d \mathbf{score}_d$?

History: For a specific \mathbf{score}_1 and \mathbf{score}_2 :

- $\frac{1}{2} \mathbf{score}_1 + \frac{1}{2} \mathbf{score}_2$ Gauthier and Ribi  re '79
- \mathbf{score}_1 B  nichou et al. '71 and Beale '71
- $\frac{1}{3} \mathbf{score}_1 + \frac{2}{3} \mathbf{score}_2$ Linderoth and Savelsbergh '99
- $\frac{1}{6} \mathbf{score}_1 + \frac{5}{6} \mathbf{score}_2$ Achterberg '09

ML + algorithm design: Potential impact

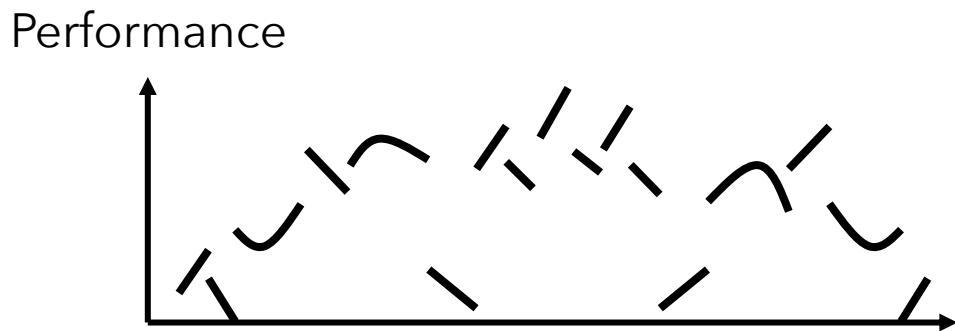
Example: integer programming

- 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



Primary challenge

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



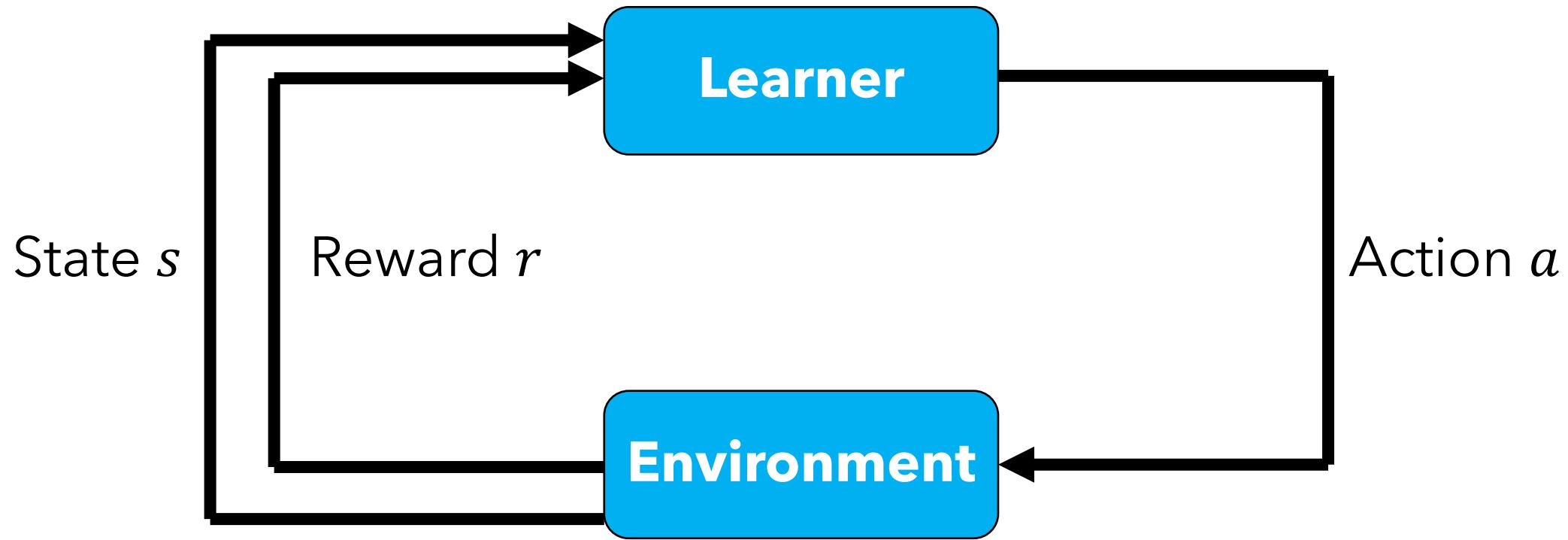
Papers we'll read

- Hutter, Frank, et al. "ParamILS: an automatic algorithm configuration framework." *JAIR* 36 (2009): 267-306.
 - Methods for **searching** through combinatorial parameter space
- Xu, Lin, et al. "SATzilla: portfolio-based algorithm selection for SAT." *JAIR* 32 (2008): 565-606.
 - How to compile a **portfolio** of algorithm configurations
 - At runtime, use **ML** to **select** a configuration from portfolio
- Gasse, Maxime, et al. "Exact combinatorial optimization with graph convolutional neural networks." *NeurIPS*. (2019).
 - Use **GNNs** to design **variable selection** policies

Outline

1. Introduction
2. Course logistics
3. Applied topics
 - i. Graph neural networks
 - ii. Integer programming and SAT
 - iii. Reinforcement learning**
 - iv. Data structures
4. Theoretical topics
5. Plan for the next 2 weeks

Learner interaction with environment



Markov decision process

S : set of states

A : set of actions

Transition probability distribution $P(s'|s, a)$

Probability of entering state s' from state s after taking action a

Reward function $R: S \rightarrow \mathbb{R}$

Goal: Policy $\pi: S \rightarrow A$ that maximizes total (discounted) reward

RL for combinatorial optimization

[Dai et al., NeurIPS'17]

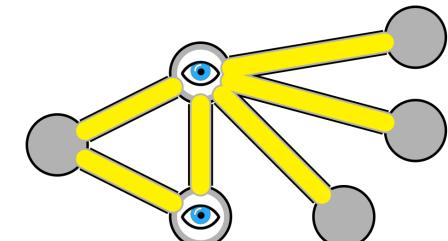
Minimum vertex cover:

Find smallest vertex subset such that each edge is covered

2-approximation:

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

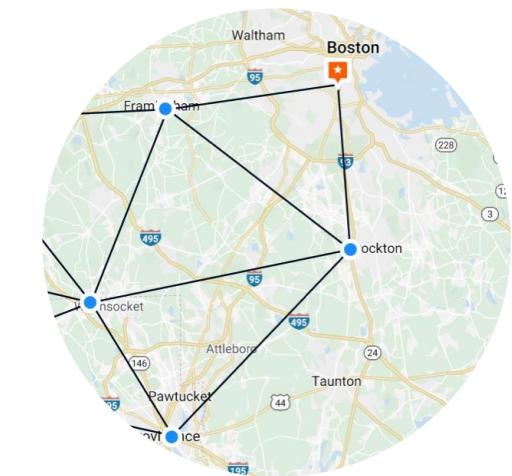
Scoring function that guides greedy algorithm



RL for combinatorial optimization

Goal: learn a scoring function to guide greedy algorithm

Problem	Greedy operation
Minimum vertex cover	Insert node into cover
Maximum cut	Insert node into subset
Traveling salesman problem	Insert node into sub-tour

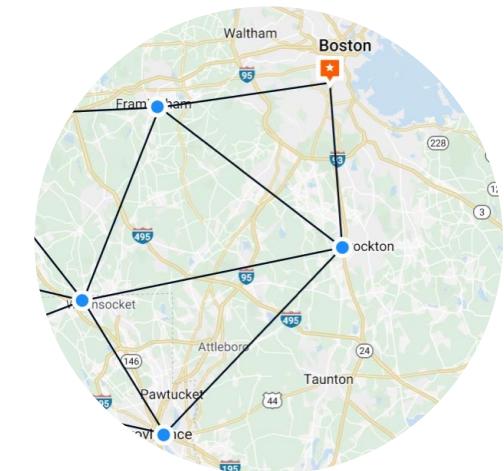


RL for combinatorial optimization

Greedy algorithm	Reinforcement learning
Partial solution	State
Scoring function	Q-function
Select best node	Greedy policy

Repeat until all edges are covered:

1. Compute node scores
2. Select best node with respect to score
3. Add best node to partial solution



Paper we'll read

Dai, Hanjun, Khalil, Elias, et al. "Learning combinatorial optimization algorithms over graphs." NeurIPS'17.

- Develop RL algorithms for a variety of combinatorial problems
- Suggest RL could be used for **algorithm discovery**
 - "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
 - i. Graph neural networks
 - ii. Integer programming and SAT
 - iii. Reinforcement learning
 - iv. Data structures**
4. Theoretical topics
5. Plan for the next 2 weeks

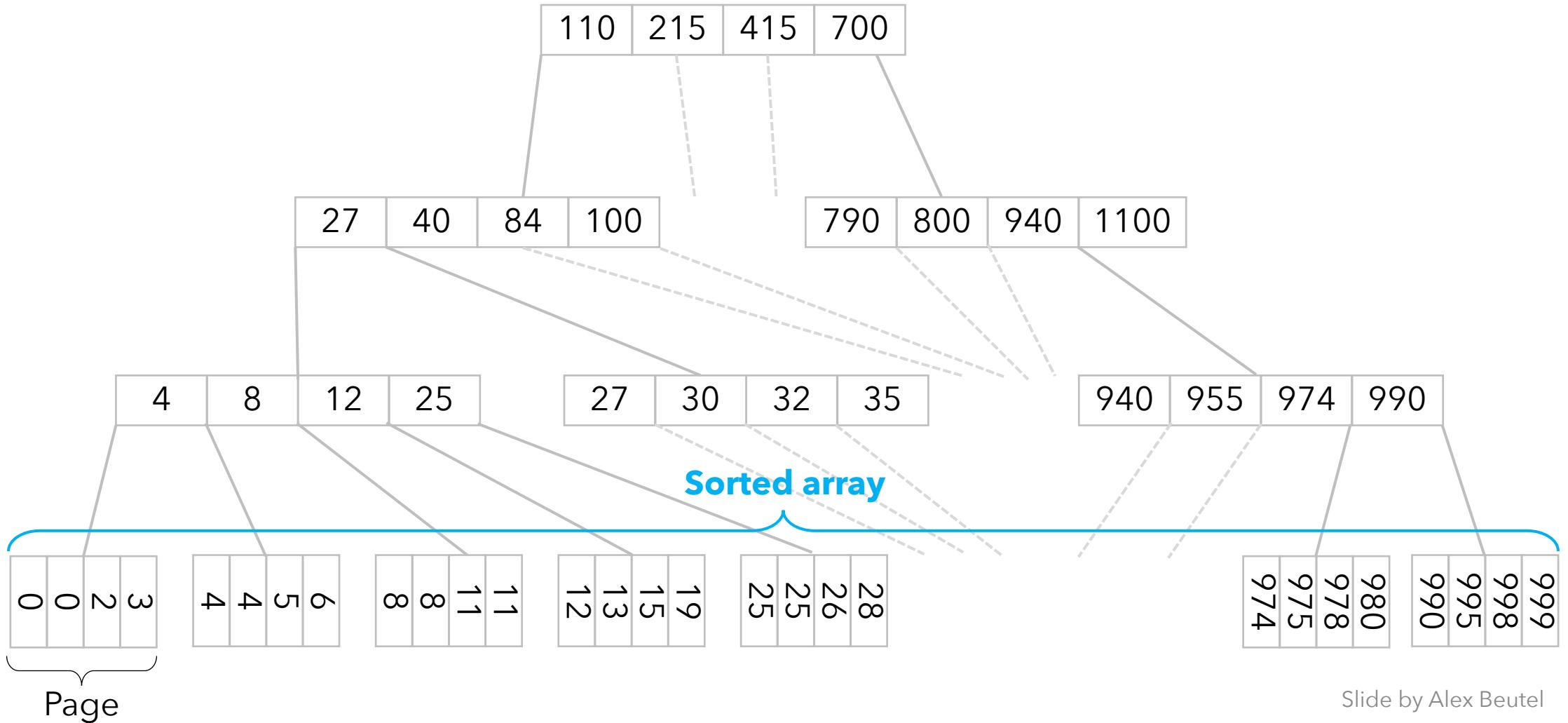
Classical databases

In classical data structures,
databases are **general purpose**. 1-size-fits all.

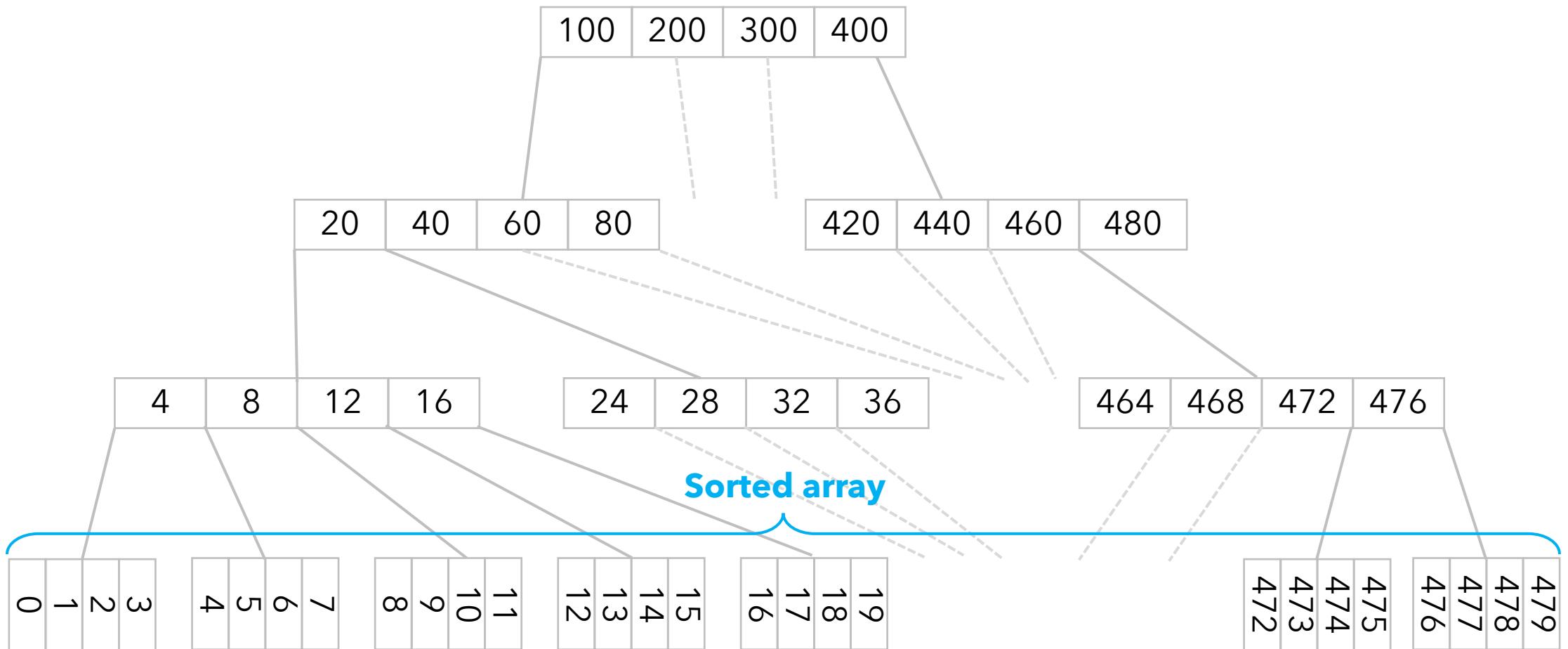
Example: B-trees

- Self-balancing **tree data structure**
- Maintains sorted data
- Searches, insertions, and deletions in **logarithmic time**

B-trees



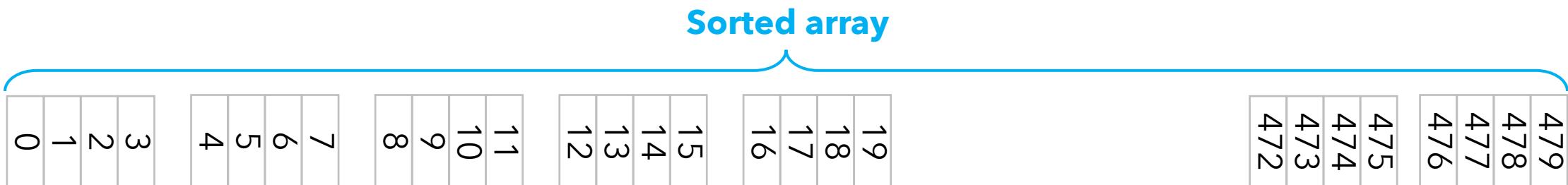
If data is all integers from 0 to 1 million?



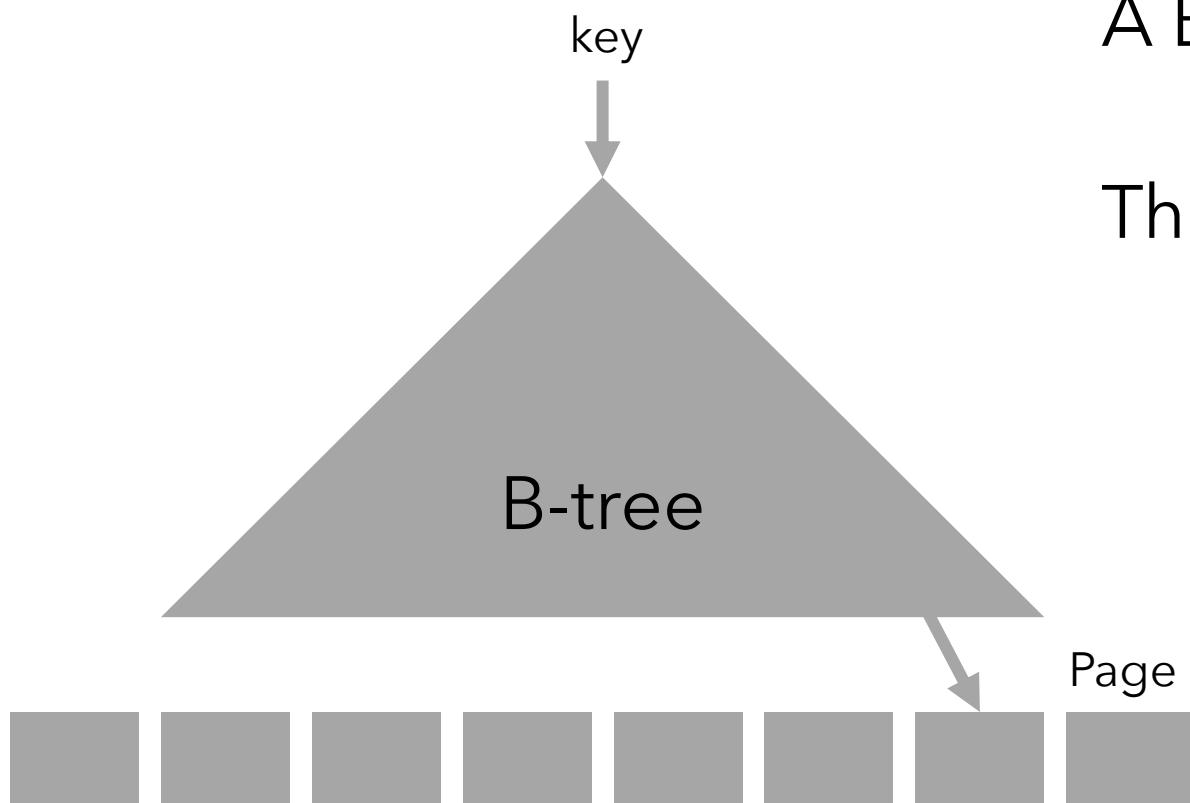
If data is all integers from 0 to 1 million?

No need for B-tree

- $O(1)$ look-up
- $O(1)$ memory



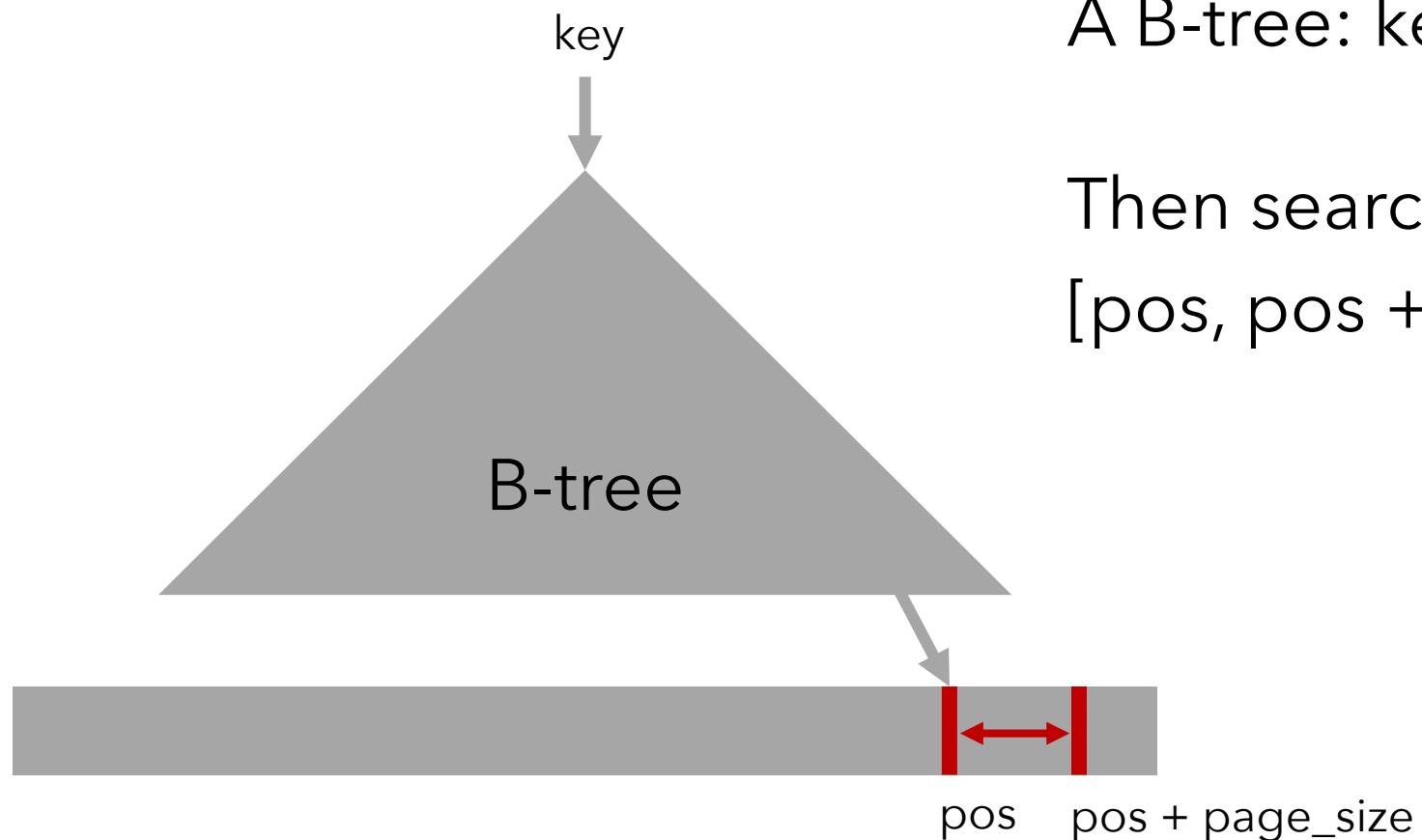
B-trees



A B-tree maps a key to a page

Then searches within the page

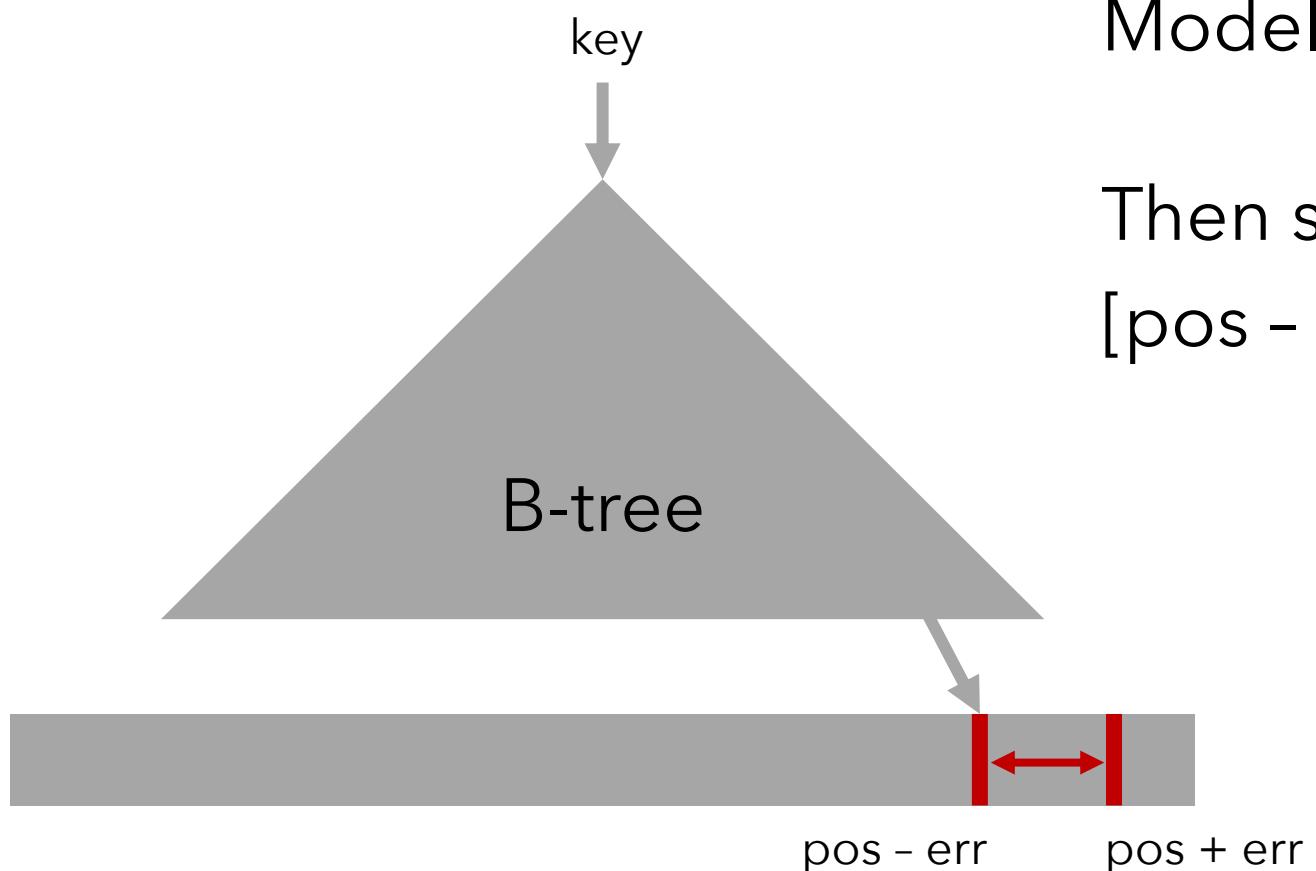
B-trees



A B-tree: key → pos

Then searches from
[pos, pos + page_size]

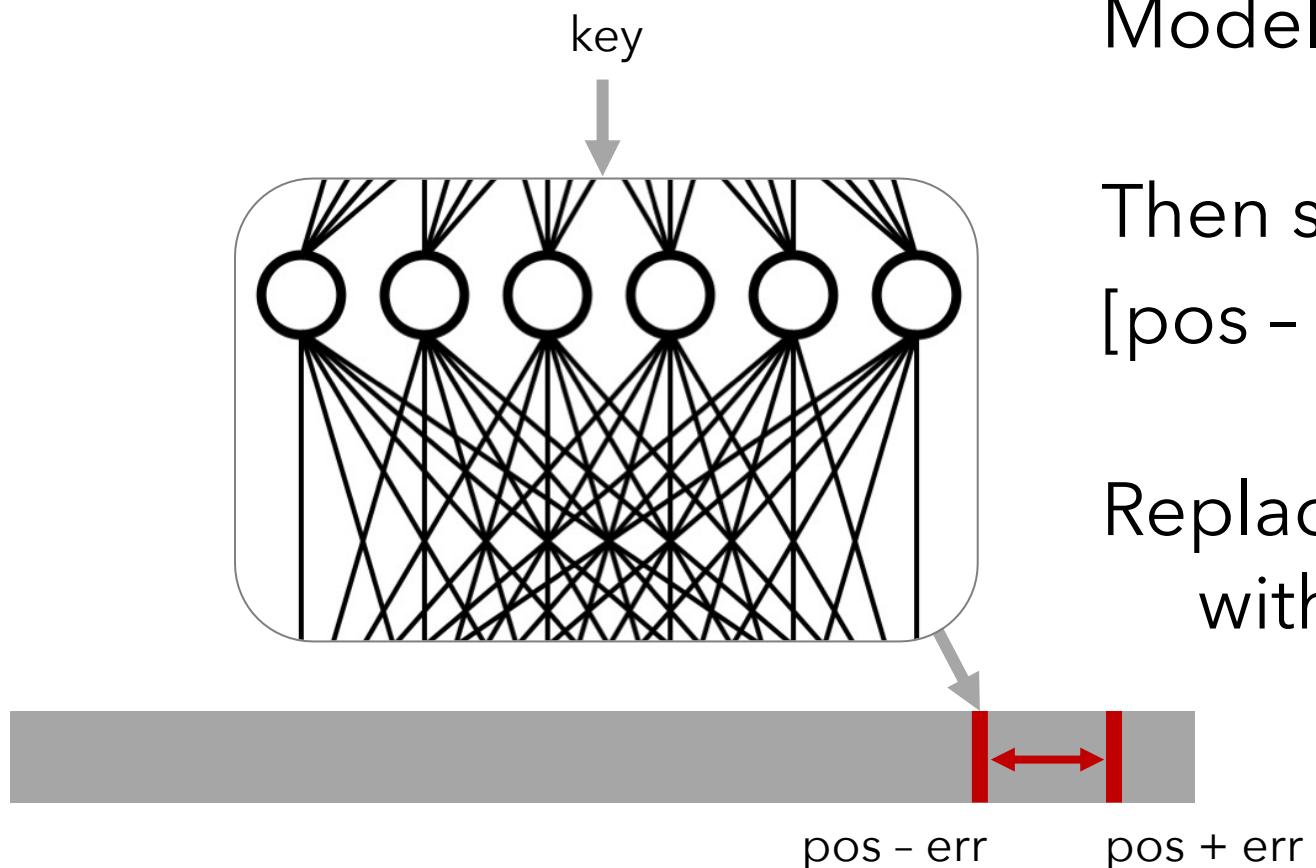
B-trees are models



Model: $f(\text{key}) \rightarrow \text{pos}$

Then searches from
[$\text{pos} - \text{err}$, $\text{pos} + \text{err}$]

B-trees are models



Model: $f(\text{key}) \rightarrow \text{pos}$

Then searches from
[$\text{pos} - \text{err}$, $\text{pos} + \text{err}$]

Replace B-tree
with **neural network?**

Paper we'll read

Kraska, Tim, et al. "The case for learned index structures."
SIGMOD. 2018.

- Naïve approach **fails**
- Investigate how to successfully **integrate** ML into databases:
 - B-trees
 - Hash maps
 - Bloom filters

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

UAI'01

Xu, Hutter, Hoos, Leyton-Brown

JAIR'08

He, Daumé, Eisner

NeurIPS'14

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

Gupta, Roughgarden

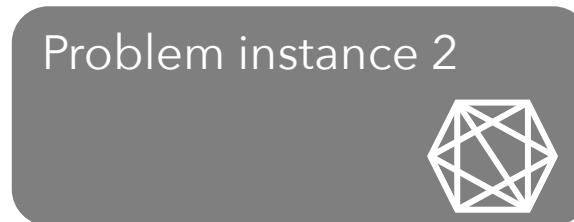
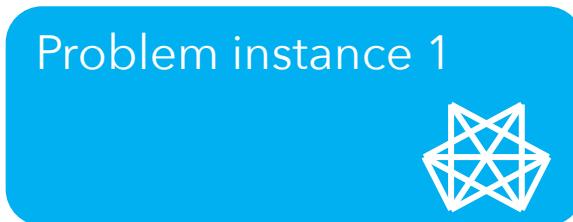
ITCS'16

Balcan

Book Chapter'20

Automated configuration procedure

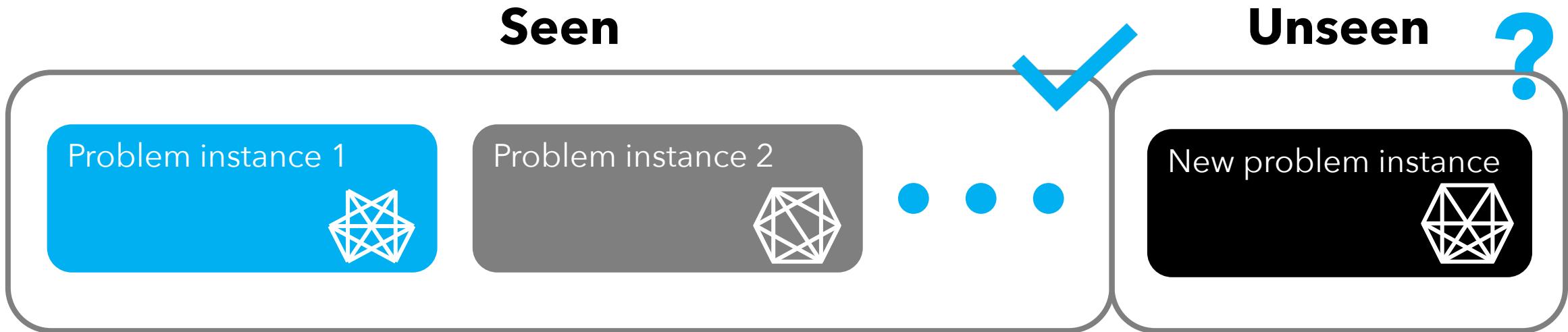
1. Fix parameterized algorithm
2. Receive set of “typical” inputs sampled from unknown \mathcal{D}



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{p} with good avg performance

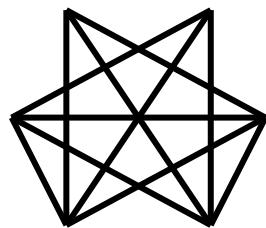
Runtime, solution quality, etc.

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

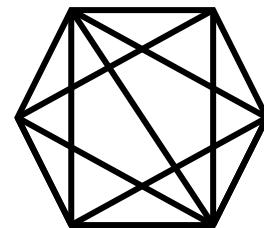
Online algorithm configuration

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

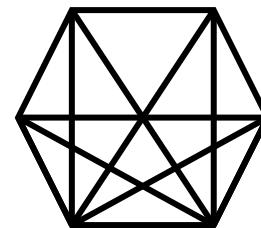
Day 1: ρ_1



Day 2: ρ_2



Day 3: ρ_3



• • •

Goal: Compete with best parameter setting in hindsight

- Impossible in the worst case
- Under what conditions is online configuration possible?

Paper we'll read

Gupta, Rishi, and Tim Roughgarden. "A PAC approach to application-specific algorithm selection." *ITCS'16*.

Statistical guarantees for algorithm configuration

- Greedy algorithms
- Tuning the step-size of gradient decent
- Etc.

Online configuration for max-weight **independent set**

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

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 $\text{OPT} = \min\{x, b\}$
- **Breakeven strategy:** Rent for $b - 1$ days, then buy

$$\text{CR} = \frac{\text{ALG}}{\text{OPT}} = \frac{x\mathbf{1}_{\{x < b\}} + (b-1+b)\mathbf{1}_{\{x \geq b\}}}{\min\{x, b\}} < 2 \text{ (best deterministic)}$$

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 $\lceil \lambda b \rceil$; else buy on start of day $\left\lceil \frac{b}{\lambda} \right\rceil$

Don't jump the gun...

...but don't wait too long

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

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

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, Kleer, Kolev,
NeurIPS'20; ...

Queuing

Mitzenmacher, ACDA'21; ...

Covering problems

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

algorithms-with-predictions.github.io

Outline

1. Introduction
2. Course logistics
3. Applied topics
4. Theoretical topics
- 5. Plan for the next 2 weeks**

Plan for the next 2 weeks

Thursday 4/6: Machine learning crash-course

- Supervised learning model
- Regression
- Classification
- Neural networks (multi-layer perceptrons)

Plan for the next 2 weeks

Thursday 4/6: Machine learning crash-course

Tuesday 4/11: Integer programming crash-course

- Linear programming
- Integer programming solvers
- SAT solving

Plan for the next 2 weeks

Thursday 4/6: Machine learning crash-course

Tuesday 4/11: Integer programming crash-course

Thursday 4/13: GNN crash-course

Plan for the next 2 weeks

Thursday 4/6: Machine learning crash-course

Tuesday 4/11: Integer programming crash-course

Thursday 4/13: GNN crash-course

Starting Tuesday 4/18: GNN paper discussions