USC

Integrating Machine Learning and Discrete Optimization

Bistra Dilkina

Assistant Professor of Computer Science, USC
Associate Director, USC Center of AI in Society

Invited Talk at CP 2019
Oct 3, 2019

AI for Sustainability and Social Good

Biodiversity Conservation

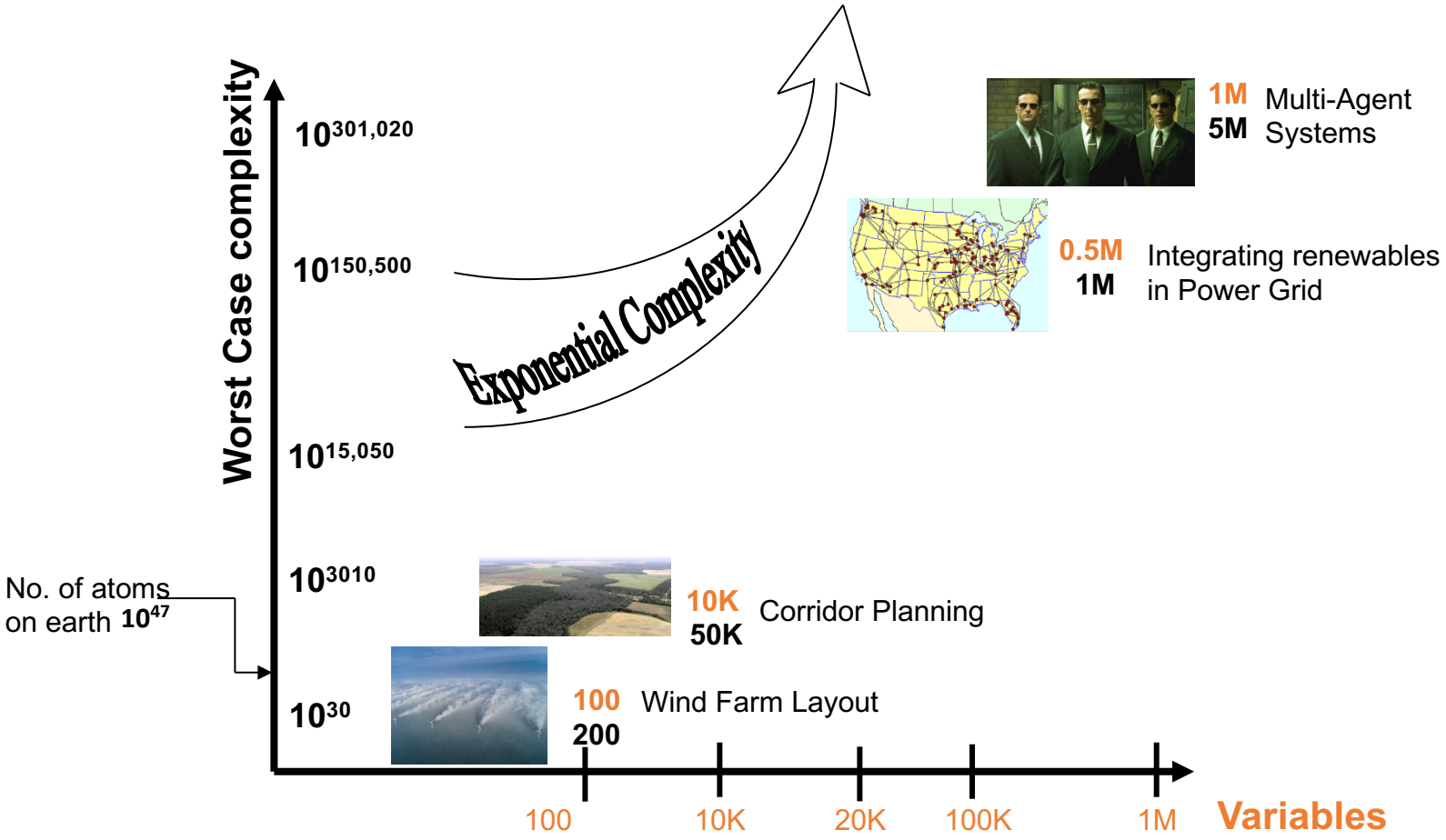
Disaster resilience

Public Health & Well-being

Design of policies to manage limited resources for best impact translate into
large-scale decision / optimization and learning problems, combining discrete and continuous effects

Constraint Reasoning and Optimization

Decision making problems of larger size and new problem structure drive the continued need to improve combinatorial solving methods



ML ⟺ Combinatorial Optimization

- Exciting and growing research area
- Design discrete optimization algorithms with learning components
- Learning methods that incorporate the combinatorial decision making they inform

ML-Driven Discrete Optimization

Infusing Discrete Optimization with Machine Learning

Infusing ML with Constrained Decision Making

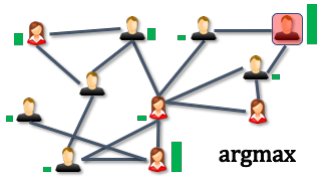
ML Paradigm

Self-Supervised Learning

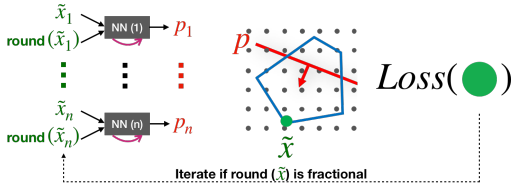
Reinforcement Learning

Supervised Learning

Greedy Heuristic

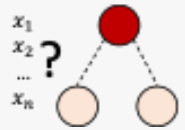


General IP Heuristic

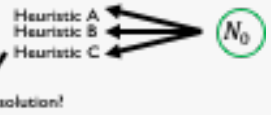


Exact Solving

Branching



Heuristic Selection



MIP as a layer in Neural Networks

Decision-focused learning for submodular optimization and LP

Graph Optimization

Integer Programming

Problem Type

Opportunity:

Tailor algorithms to
a **family** of **instances** to
discover novel **search strategies**

A realistic setting

- Same problem is solved repeatedly with slightly different data
- Delivery truck in Los Angeles:
  - Daily routing in the same area with slightly different customers
- Typical approach:
  - Customize branch-and-bound / approx. / heuristic

Mixed Integer Linear Programs (MIP) & their Applications

$$(MIP) \quad z^* = \min\{c^T x \mid Ax \leqslant b, x \in \mathbb{R}^n, x_j \in \mathbb{Z}\ \forall j \in I\}$$

Widely used to model real-world decision-making scenarios:

Conservation planning

Road Infrastructure Planning

Kidney exchange

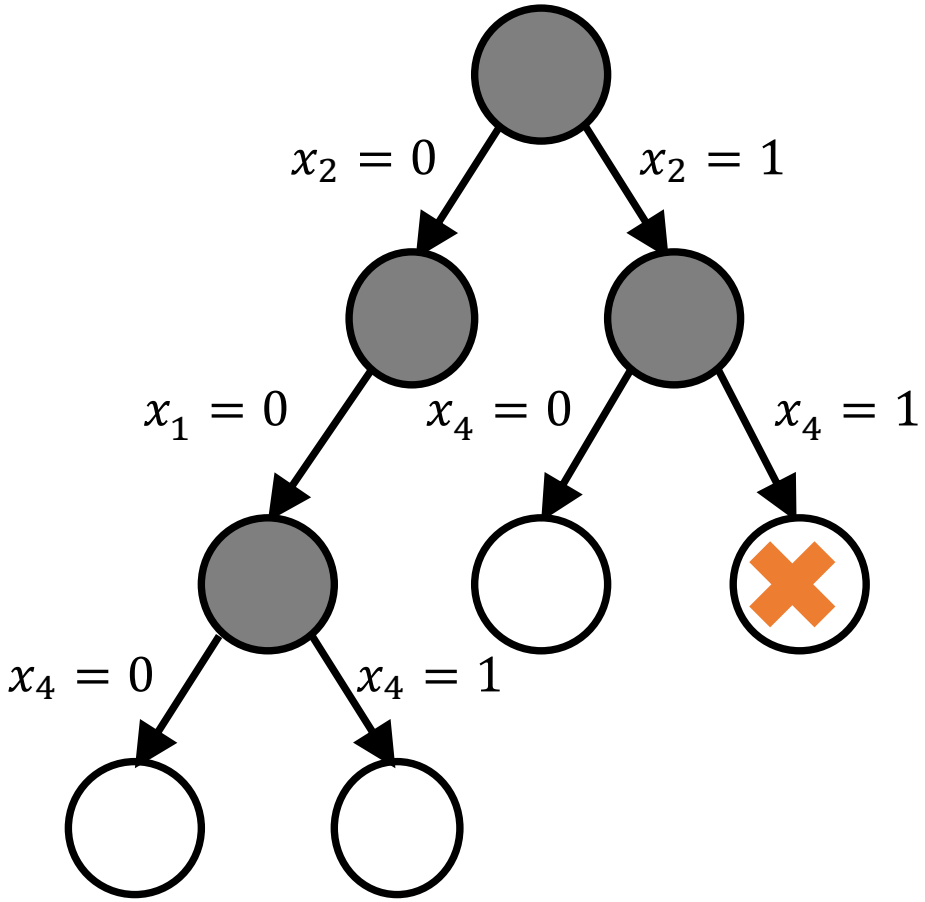
Airline fleet and crew scheduling

Branch-and-Bound in a Nutshell

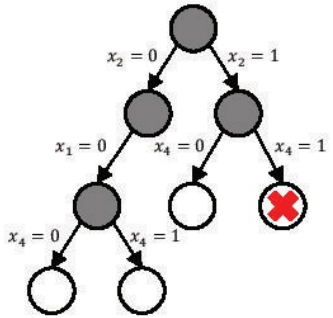
$$\min_{x} c^T x \; s.t. \; Ax \leq b, x \in \{0,1\}^n$$

Repeat:

- Select Node
- Solve Relaxation
- Add Cuts
- Run Heuristics
- Branch



Opportunities for Machine Learning in B&B

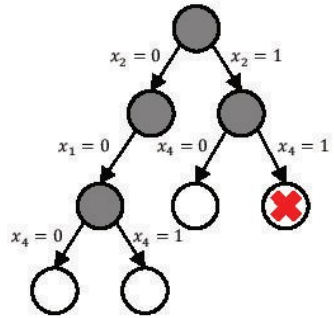


Task	Issue	Current Approach
Select Branching Variable	what selection rule?	single **hand-designed** ranking metric
Add Cuts	which cuts to add?	**hand-designed** ranking formula
Run Heuristics	which heuristics to run?	run each every k nodes (**fixed parameter** k)
Select Node	what selection rule?	best-first

Improve B&B with ML-driven strategies for these tasks.

- Most related work uses ML at a "meta level", e.g. algorithm portfolios, parameter tuning, strategy selection
- Here: use ML to discover **novel** strategies that **dynamically** guide the behavior of the BnB algorithm

Opportunities for Machine Learning in B&B

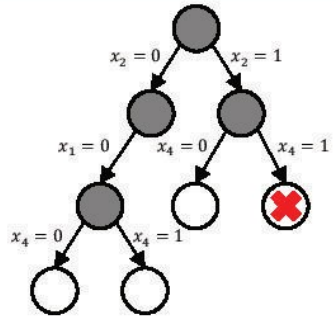


Task	Issue	Current Approach
Select Branching Variable	what selection rule?	single **hand-designed** ranking metric
Add Cuts	which cuts to add?	**hand-designed** ranking formula
Run Heuristics	which heuristics to run?	run each every k nodes (**fixed parameter** k)
Select Node	what selection rule?	best-first

Improve B&B with ML-driven strategies for these tasks.

- Most related work uses ML at a "meta level", e.g. algorithm portfolios, parameter tuning, strategy selection
- Here: use ML to discover **novel** strategies that **dynamically** guide the behavior of the BnB algorithm

Opportunities for Machine Learning in B&B



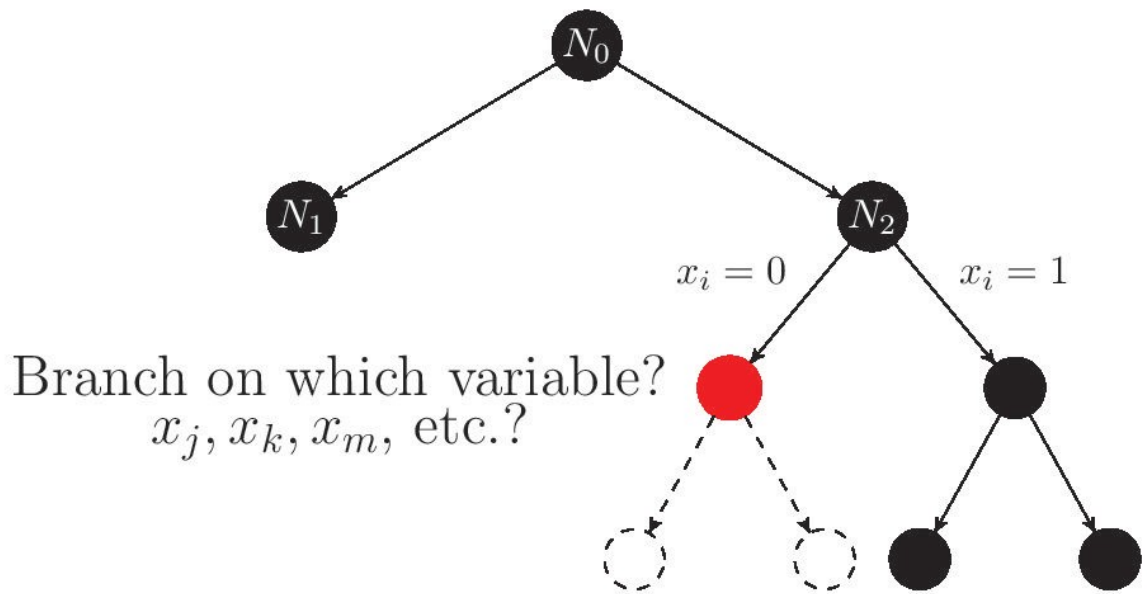
Task	Issue	Current Approach
Select Branching Variable	what selection rule?	single **hand-designed** ranking metric
Add Cuts	which cuts to add?	**hand-designed** ranking formula
Run Heuristics	which heuristics to run?	run each every k nodes (**fixed parameter** k)
Select Node	what selection rule?	best-first

Improve B&B with ML-driven strategies for these tasks.

- Most related work uses ML at a "meta level", e.g. algorithm portfolios, parameter tuning, strategy selection
- Here: use ML to discover **novel** strategies that **dynamically** guide the behavior of the BnB algorithm

Learning to Branch

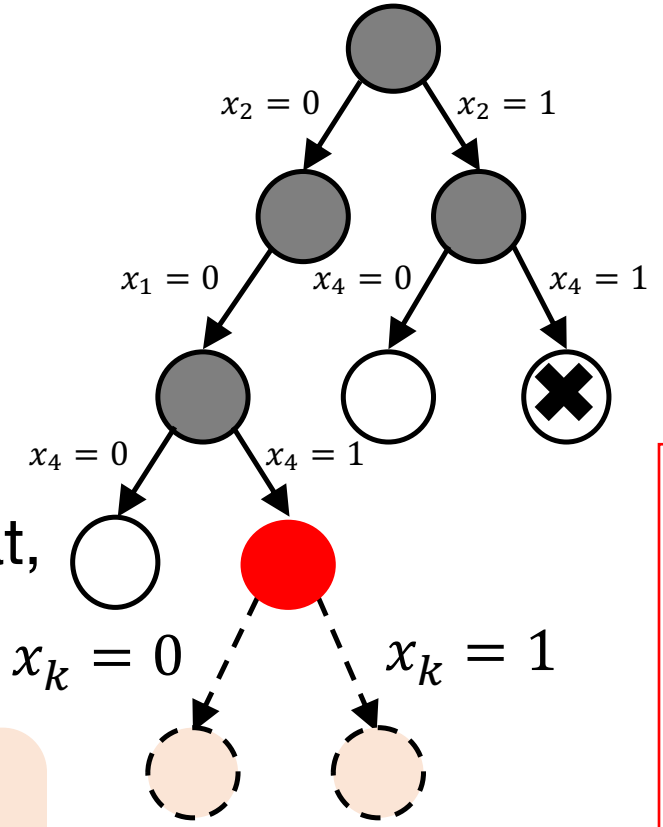
- Branching on the "right" variables can have a dramatic impact on the number of nodes in B&B tree
  - e.g. small backdoors in MIPs [Dilkina et al, CPAIOR 2009]



Joint work with my PhD student **Elias Khalil**, and our collaborators George Nemhauser, Le Song and Pierre Le Bodic [AAAI 2016]

Learning to Branch

Ideally, select variables that lead to small sub-tree ↔ many infeasible nodes ✖



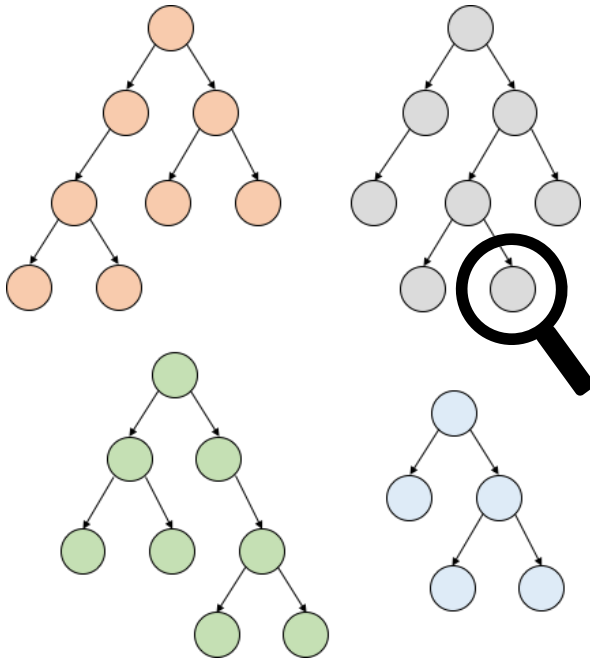
x_3?
x_5?
...
x_n?

Strong Branching (SB) achieves that, but is extremely costly

Given: dataset of
(variable features, Strong Branching score)

Learn: a ranking model that imitates SB

Learning to Branch



Given: dataset of
(variable features, Strong Branching score)

Learn: a ranking model that imitates SB

		y_j^i	feature 1	feature 2	feature 3	feature 4
N_1	x_2	0	0.4	0.4	0.3	0.9
	x_4	1	0.3	0.9	0.4	0.9
	x_5	0	0.2	0.2	0.6	0.7
	x_6	1	0.5	0.8	0.4	0.4

72 features, e.g.: objective coefficient, pseudocosts, statistics for constraint degrees **in node subproblem**

+ pairwise product of features

labels: **1** if $SB(x_i)$ close to max. SB at node; **0** o.w.

Model Learning

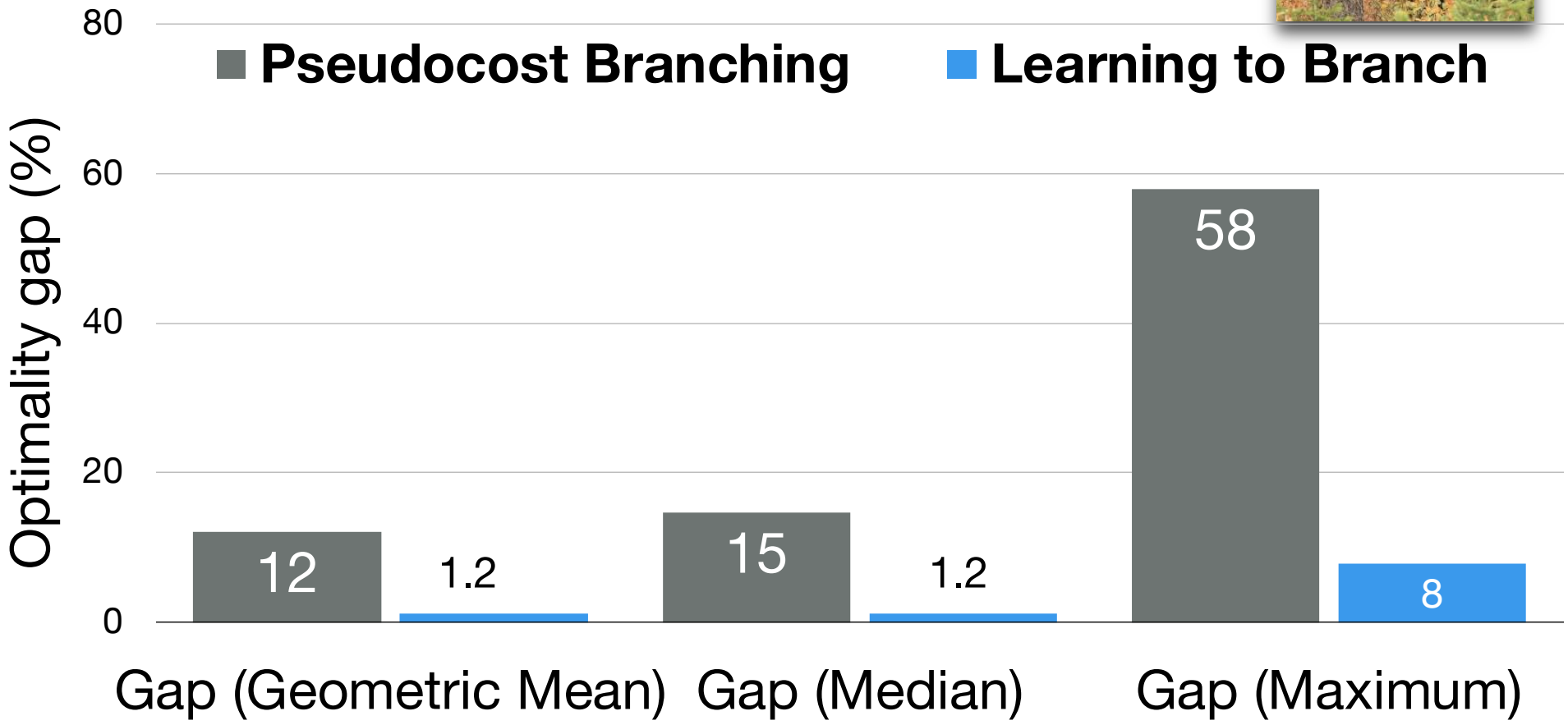
Goal

Learn a function of the features that **ranks variables** with better labels higher than other variables.

- "Learning to rank with pairwise loss", popular in web search
- Learn an f that minimizes the number of wrong pairwise orderings between variables of the same node
- Ranking function f is linear in the features
-
- NP-hard but can optimize efficiently an upper bound as a convex SVM optimization problem [Joachims2006].
- Open-source implementation SVM^{rank}:
  `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

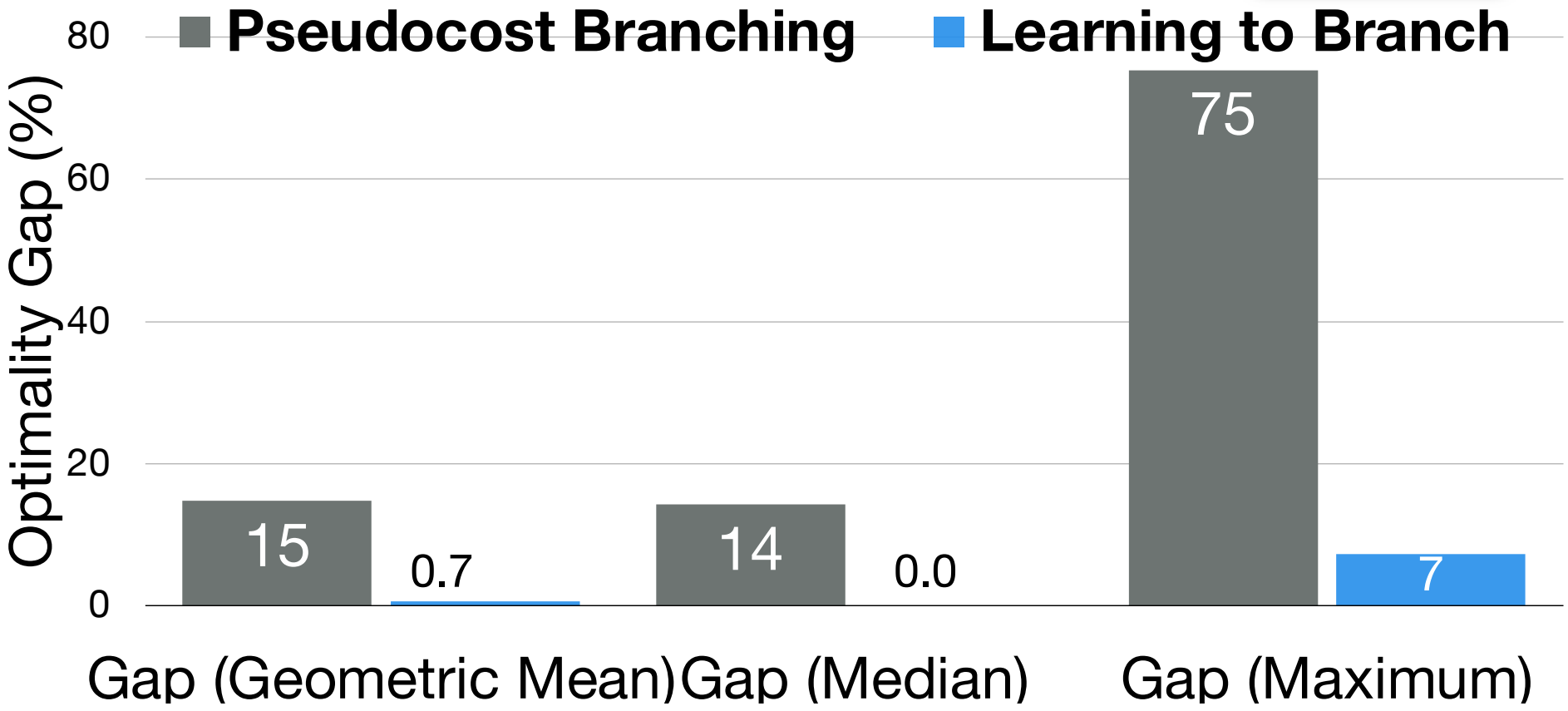
Learning to Branch **in Practice**

Wildlife Corridor Design

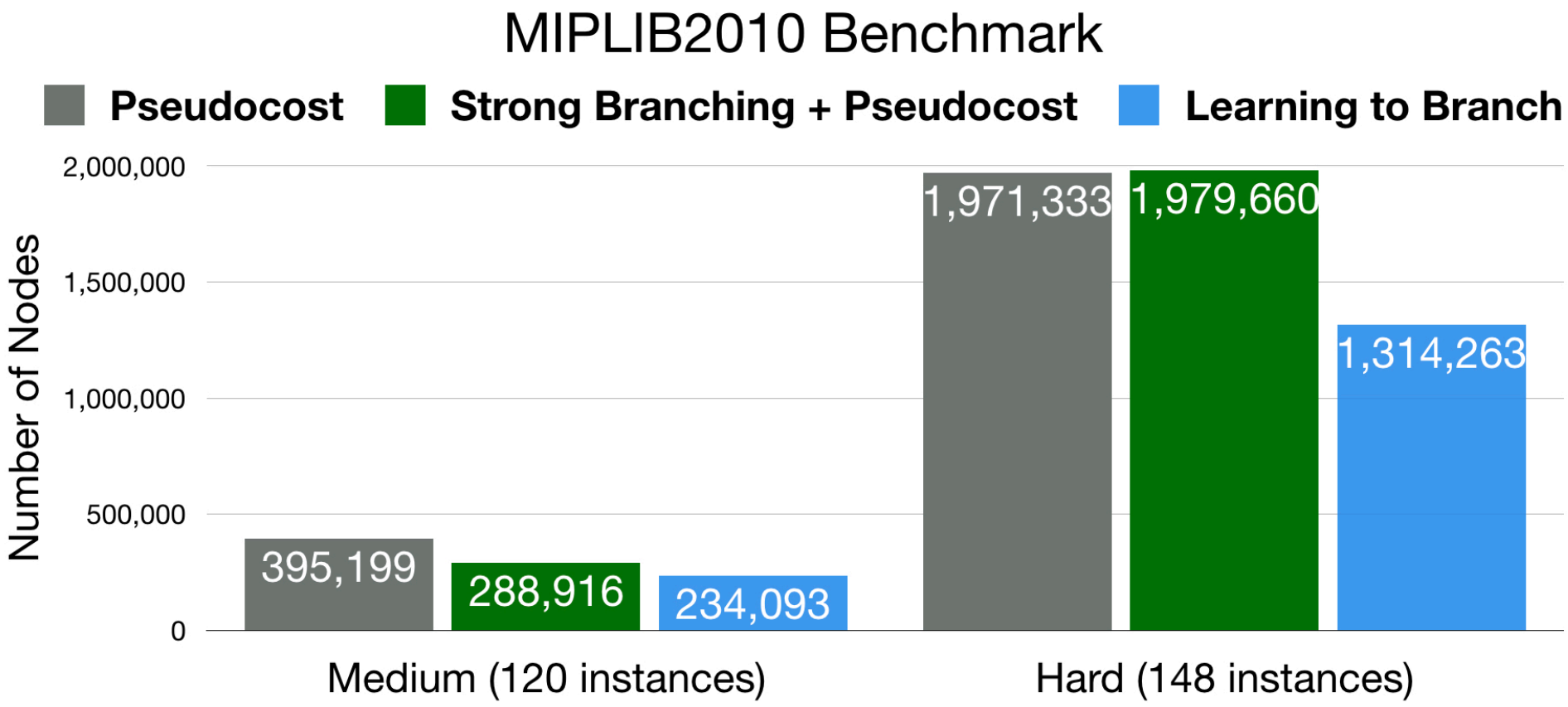


Learning to Branch **in Practice**

Road Network Design for Flooding



Learning to Branch **On-The-Fly (Per Instance)**

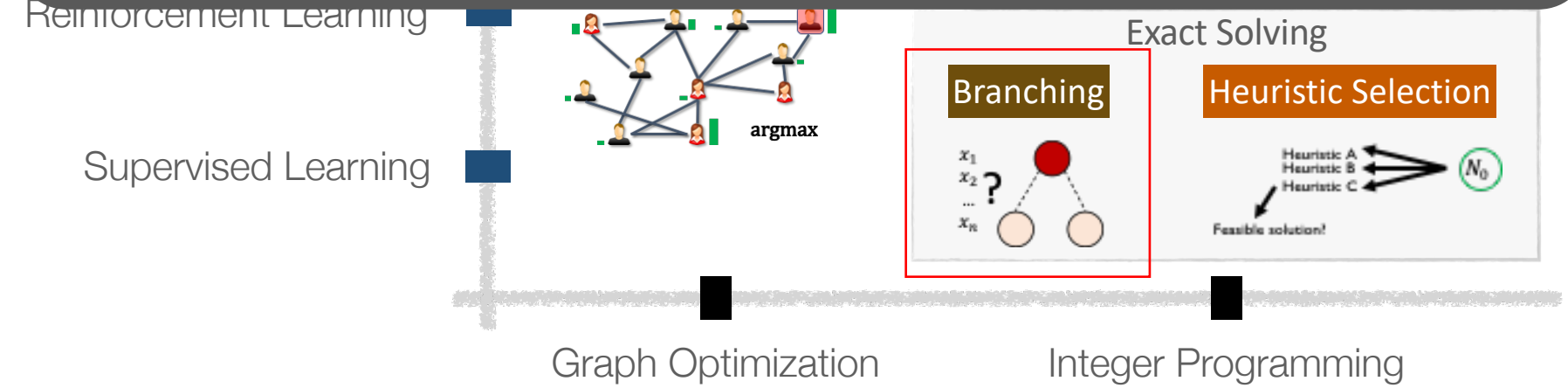


Compared to PC on Hard instances:
33% fewer nodes, **14%** less time

Learning-Driven Algorithm Design

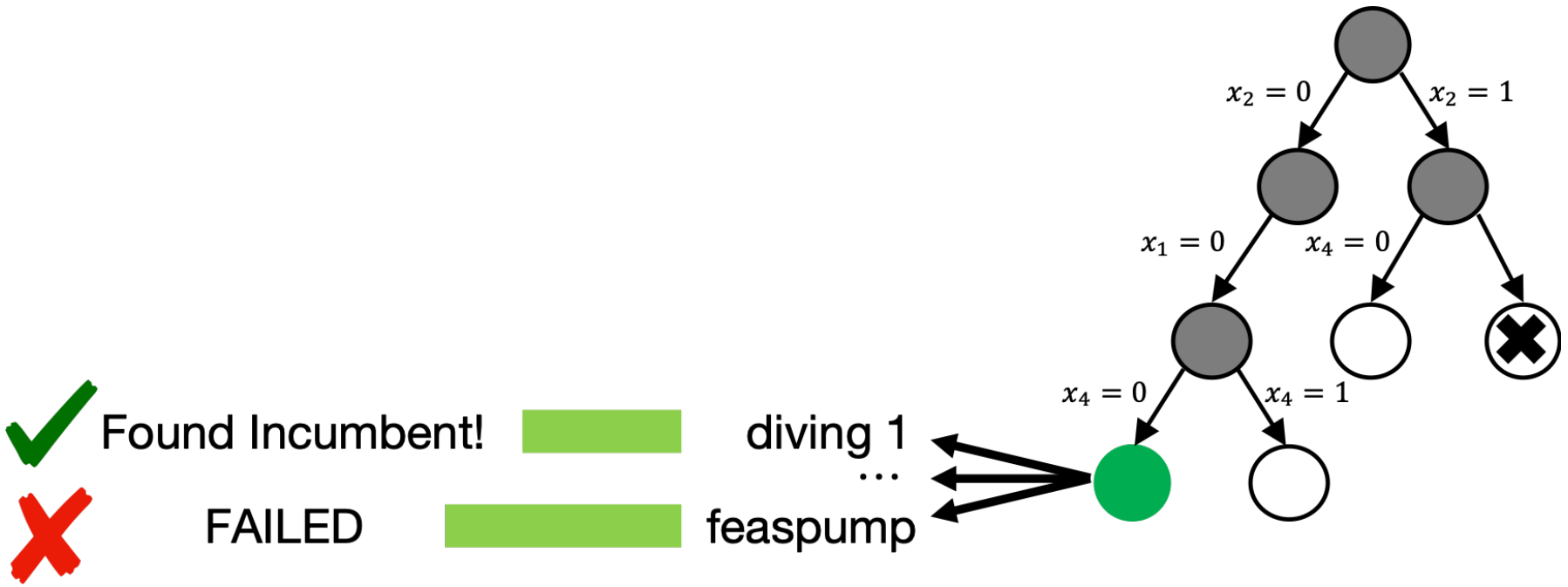
Takeaways

- First ML framework for branching
- Feature Engineering + Linear Ranking Model
- Significant improvements on families of instances
- On-the-fly version for limited data settings



Learning to Run Heuristics

In many applications, obtaining **good solutions quickly** is equally or more important than proving optimality.



Using Primal Heuristics in Branch-and-Bound

USC

- **Primal heuristic**: an incomplete algorithm that can find a feasible solution (and hence can improve the primal bound).
- **Low Success Rate**: $\frac{\#\text{incumbents found}}{\#\text{runs}}$
- **Time Cost**: time for heuristic could instead be used to solve additional nodes.
- **Current approach in solvers**: run heuristic every k nodes
  - Frequency k: a parameter that is manually tuned.

Goal

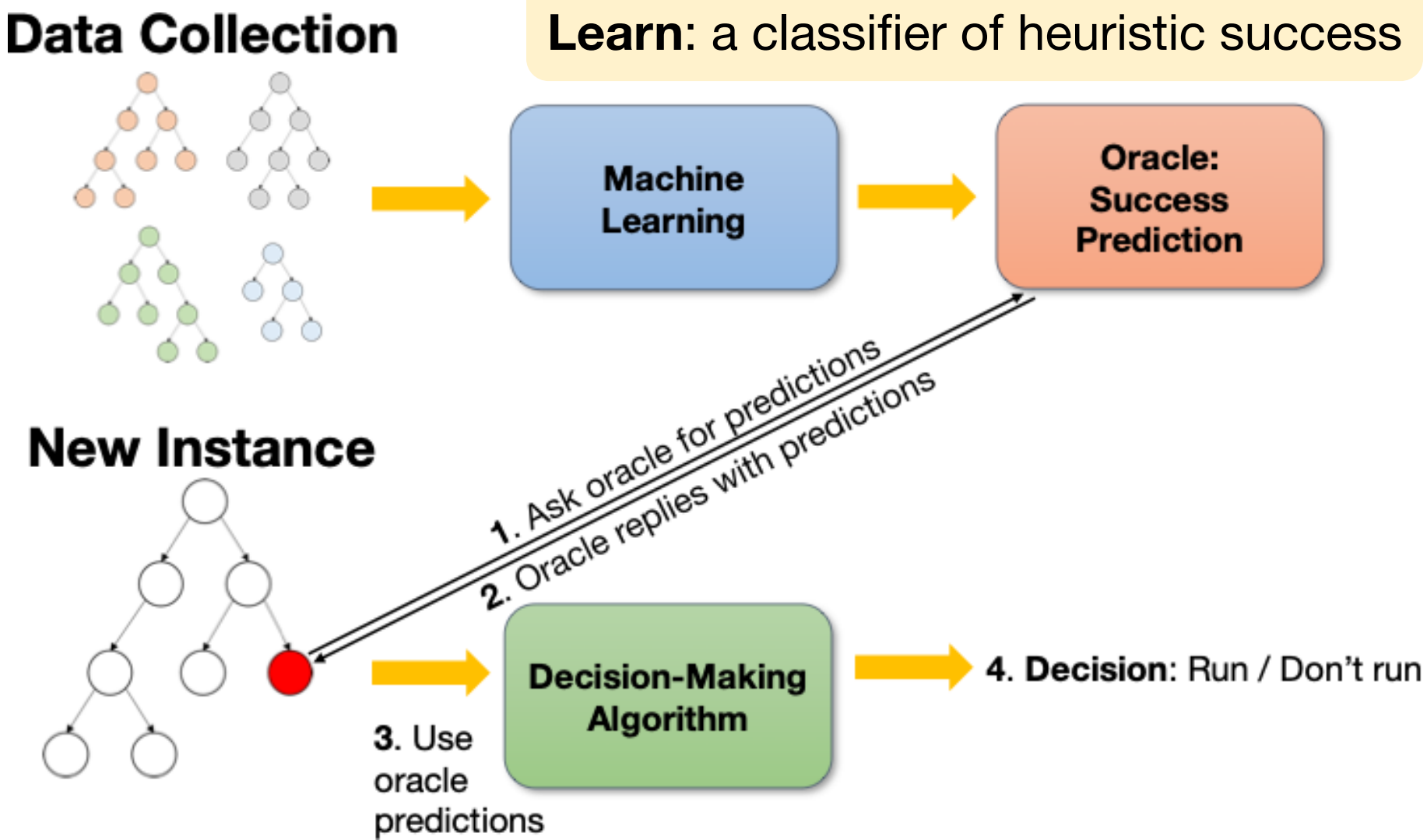
Automate this decision-making task using ML and improve performance

Joint work with my PhD student Elias Khalil, and our collaborators George Nemhauser, Shabbir Ahmed, Yufen Shao [IJCAI 2017]

Learning to Run Heuristics

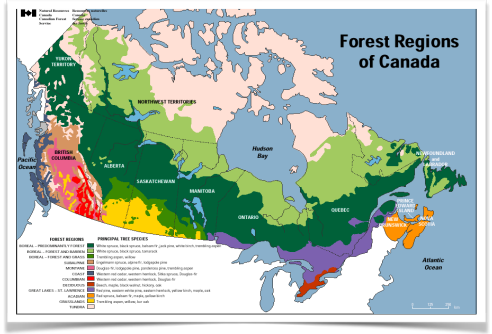
Given: dataset of
(node features, 0/1 success flag)

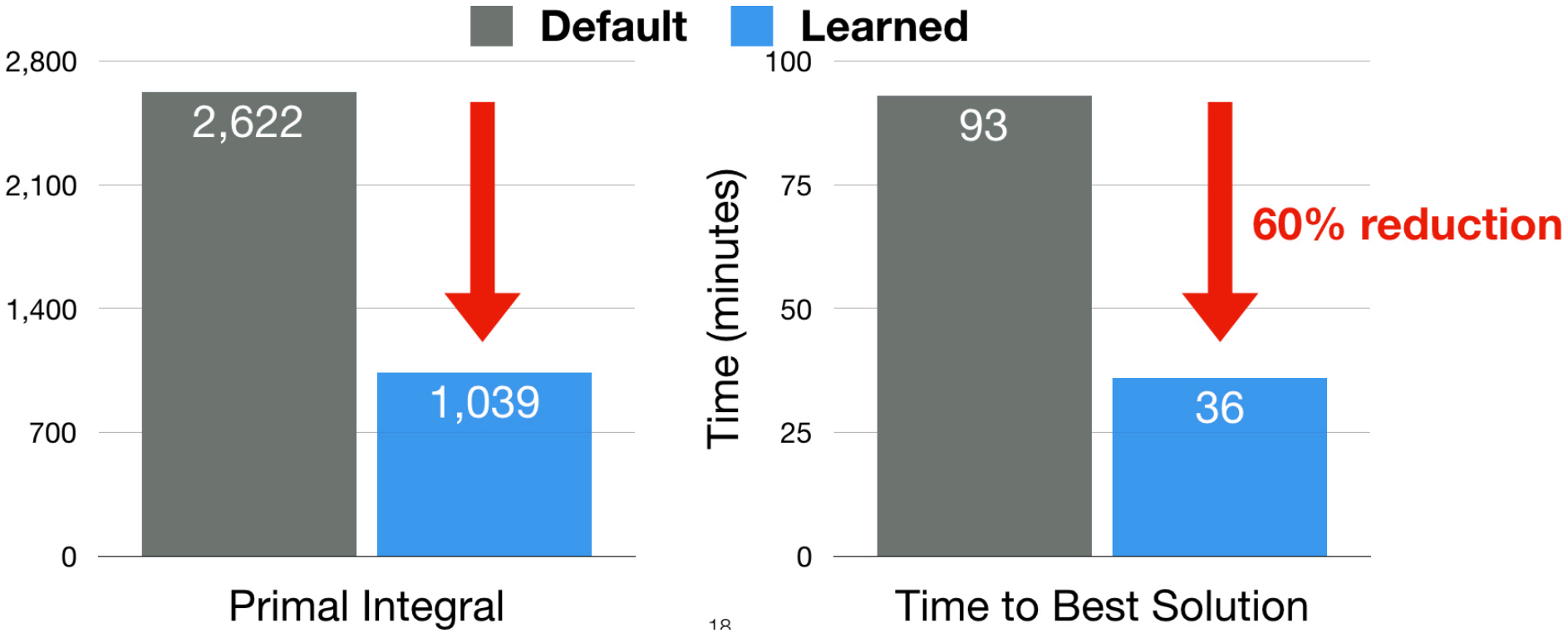
Learn: a classifier of heuristic success



Learning to Run Heuristics **in Practice**

Forest Harvesting: Generalized Independent Set

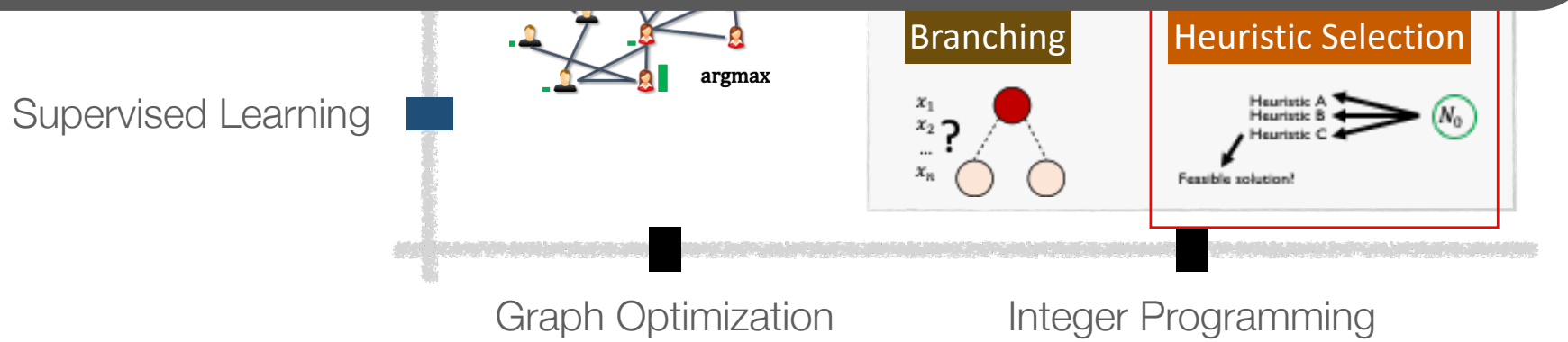




Learning-Driven Algorithm Design

Takeaways

- First ML framework for heuristic selection in B&B
- Dynamic, node-dependent decision-making
- Forest Harvesting: 60% reduction in Primal Integral
- MIPLIB2010 : Even on the heterogeneous benchmark 6% reduction in Primal Integral



Learning-driven Algorithm Design

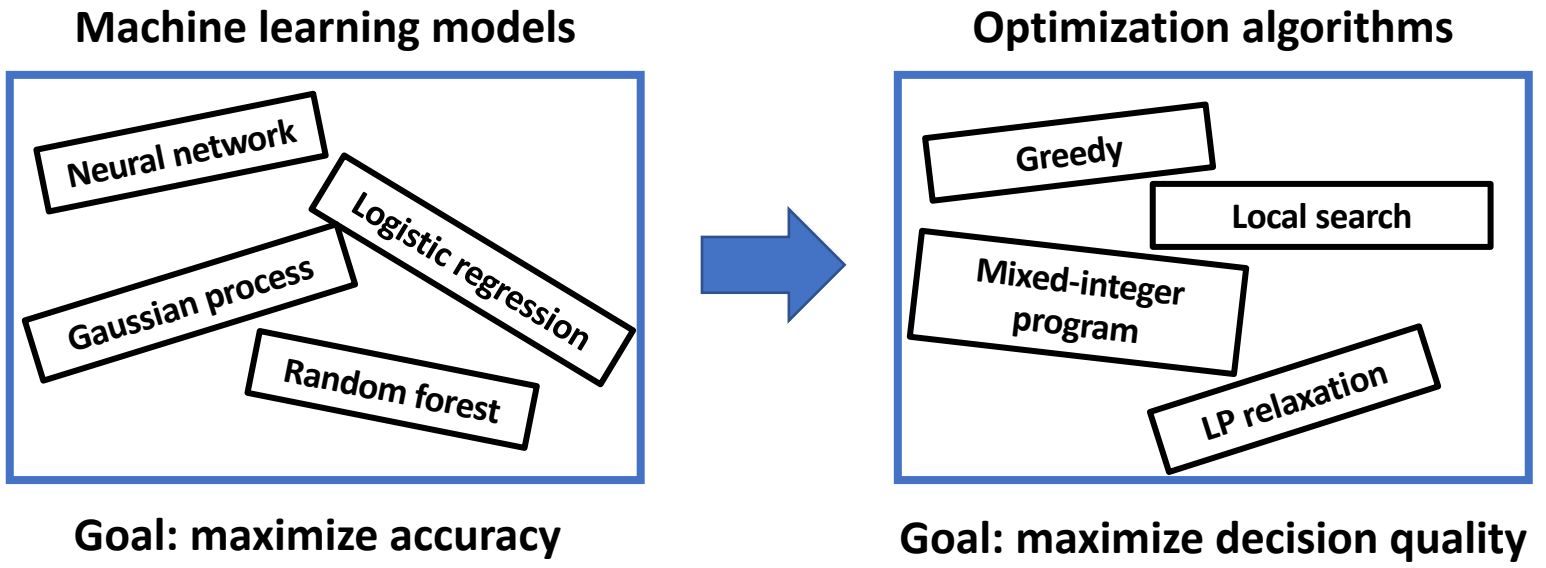
- Exciting and growing research area
- Many Open Directions:
  - Local Search, Nonlinear Optimization, Constraint Programming, Decision Diagrams
  - Theoretical Foundations
  - Lifelong Learning

The data-decisions pipeline

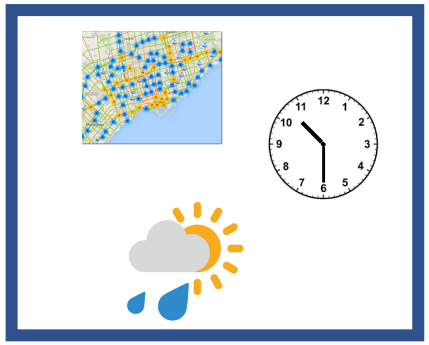
Many real-world applications of AI involve a common template:

[Horvitz and Mitchell 2010; Horvitz 2010]

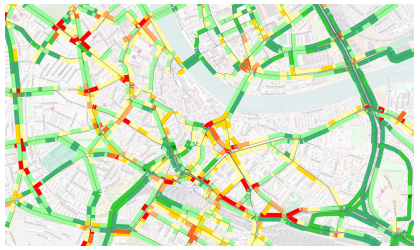
Typical two-stage approach



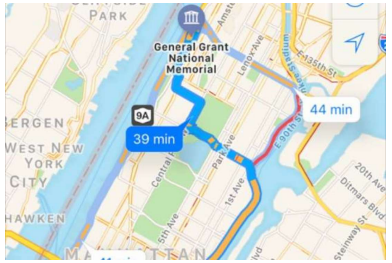
Google maps



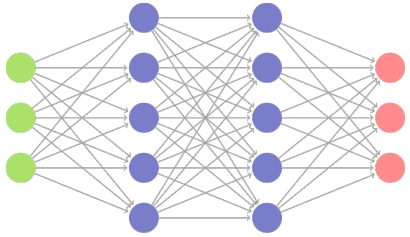
Data



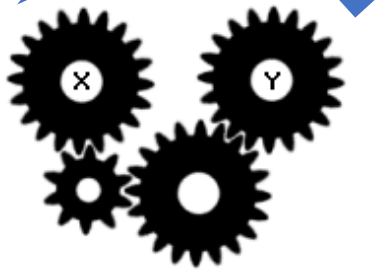
Predicted travel times



Shortest path



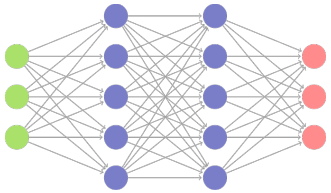
Predictive model



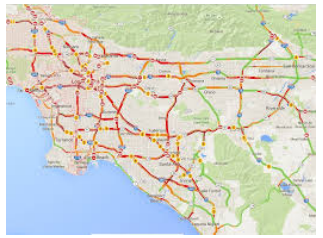
Routing algorithm

Two-stage training

Data

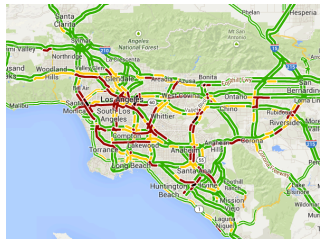


Predictive model



Predicted delays

vs



Actual delays

Update model to make predictions closer to actual delays

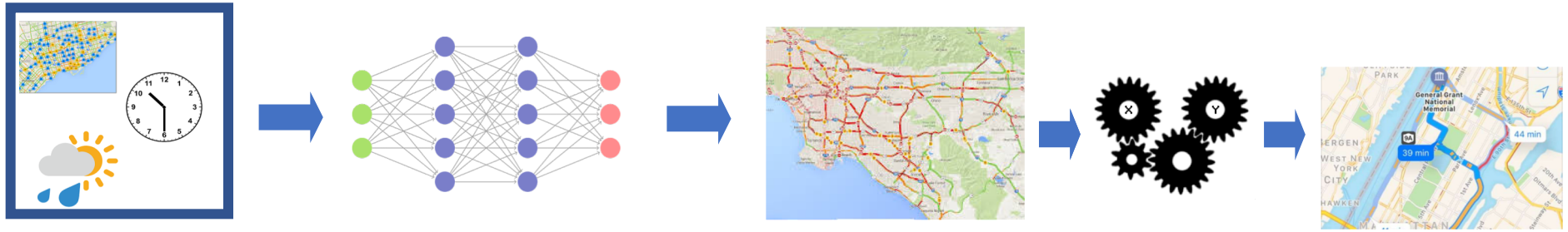
Challenge

- Maximizing accuracy ≠ maximizing decision quality
- “All models are wrong, some are useful”
- Two-stage training doesn’t align with end goal

Key idea:

Decision-focused learning

Automatically shape the ML model's loss by incorporating the optimization problem into the training loop



Ferber et al (2019), Wilder et al. (2019), Amos and Kolter (2017)

Decision-focused learning

Objective function $\boldsymbol{f}(x, \theta)$

$\boldsymbol{x} \in \{0,1\}^n$ are the **decision variables**

$\boldsymbol{\theta}$ are **unknown parameters** (i.e. the coefficients in the objective e.g., true travel times)

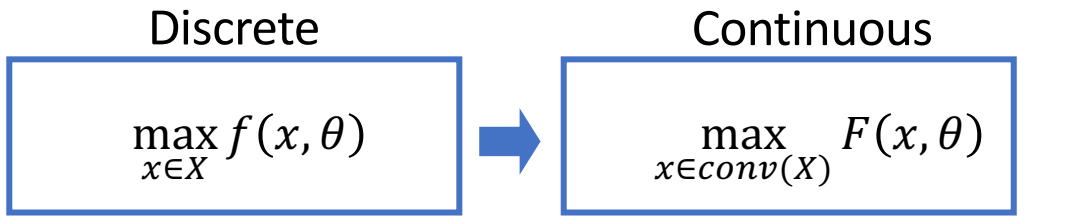
Idea: **Take derivative of decision objective w.r.t. ML model weights**, train model via gradient descent (e.g. *similar approach for convex opt. [Donti et al '17]*)

$$\frac{\partial \text{obj(decision)}}{\partial \text{weights}} = \frac{\partial \text{prediction}}{\partial \text{weights}} \frac{\partial \text{decision}}{\partial \text{prediction}} \frac{\partial \text{obj(decision)}}{\partial \text{decision}}$$

Approach

[Wilder, Dilkina, Tambe, AAAI 2019]

- **Challenge:** the optimization problem is discrete!
- **Solution: relax to continuous problem, differentiate, round**



- How to compute $\frac{dx^*}{d\theta}$?
- Idea: (locally) optimal continuous solution must satisfy KKT conditions (which are sufficient for convex problems)

- The KKT conditions define a system of linear equations based on the gradients of the objective and constraints around the optimal point.
- Differentiate those equations at optimum (e.g. convex opt. [Donti et al '17])

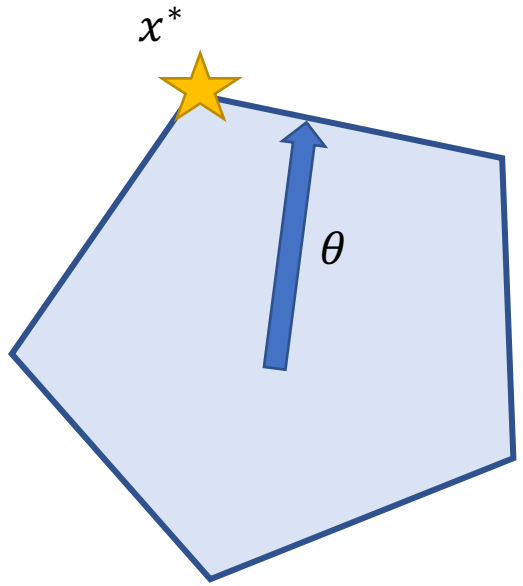
Linear programs

Model exactly combinatorial problems like bipartite matching, shortest path, mincut, etc.

Or correspond to a relaxation of other combinatorial problems

Standard form:

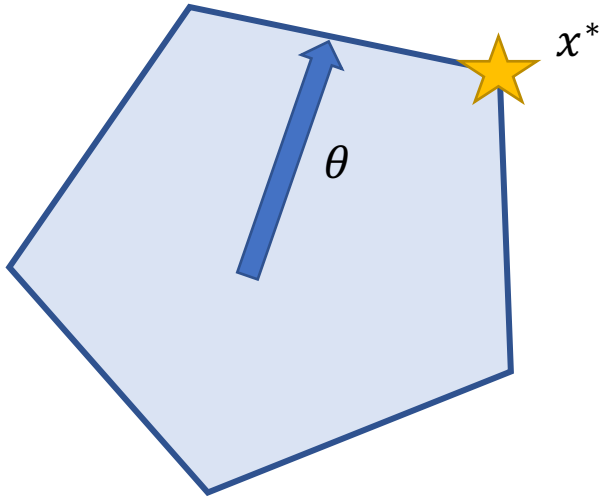
$$\max_x \theta^T x$$
$$Ax \leq b$$



- $\frac{dx^*}{d\theta}$ doesn't exist!

- Solution: add a regularizer to smooth things out

$$\max_x \theta^T x \ - \gamma \|x\|_2^2$$
$$Ax \leq b$$



- Now, Hessian is $\nabla_x^2 f(x, \theta) = -2\gamma I \prec 0$
- Provably (a) differentiable and (b) close to original LP

Results

USC

- Combinatorial problems: encoded as LP, e.g. **bipartite maximum matching**
- Combinatorial problems: submodular maximization, e.g. **influence maximization**

- Decision-focused has consistently better solution quality
  - **15-70% improvement in solution over 2-Stage**, across three domains

Table 1: Solution quality of each method for the full data-decisions pipeline.

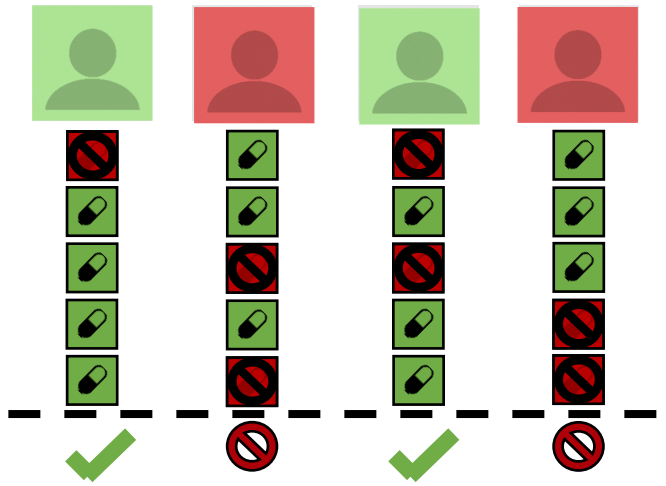
	Budget allocation			Matching	Diverse recommendation		
$k =$	5	10	20	–	5	10	20
NN1-Decision	**49.18 ± 0.24**	**72.62 ± 0.33**	**98.95 ± 0.46**	2.50 ± 0.56	**15.81 ± 0.50**	**29.81 ± 0.85**	**52.43 ± 1.23**
NN2-Decision	44.35 ± 0.56	67.64 ± 0.62	93.59 ± 0.77	**6.15 ± 0.38**	13.34 ± 0.77	26.32 ± 1.38	47.79 ± 1.96
NN1-2Stage	32.13 ± 2.47	45.63 ± 3.76	61.88 ± 4.10	2.99 ± 0.76	4.08 ± 0.16	8.42 ± 0.29	19.16 ± 0.57
NN2-2Stage	9.69 ± 0.05	18.93 ± 0.10	36.16 ± 0.18	3.49 ± 0.32	11.63 ± 0.43	22.79 ± 0.66	42.37 ± 1.02

- But typically much less accurate (wrt AUC, MSE etc.)

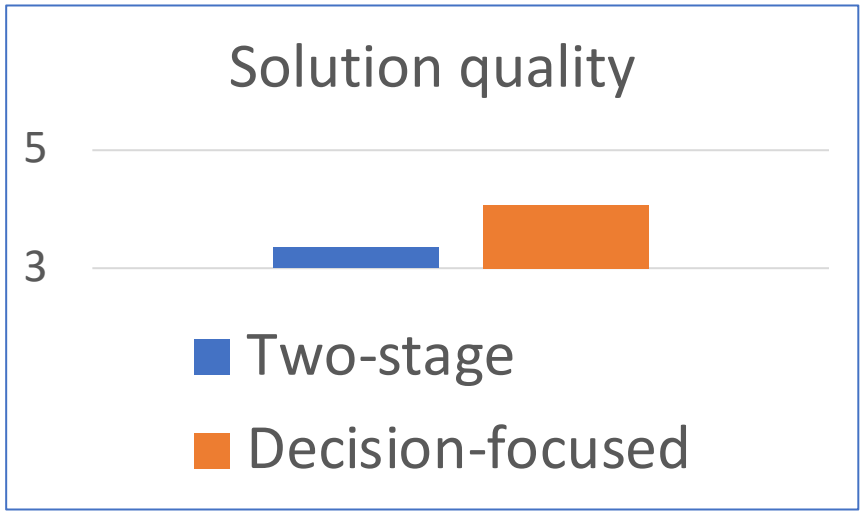
Application: Tuberculosis treatment

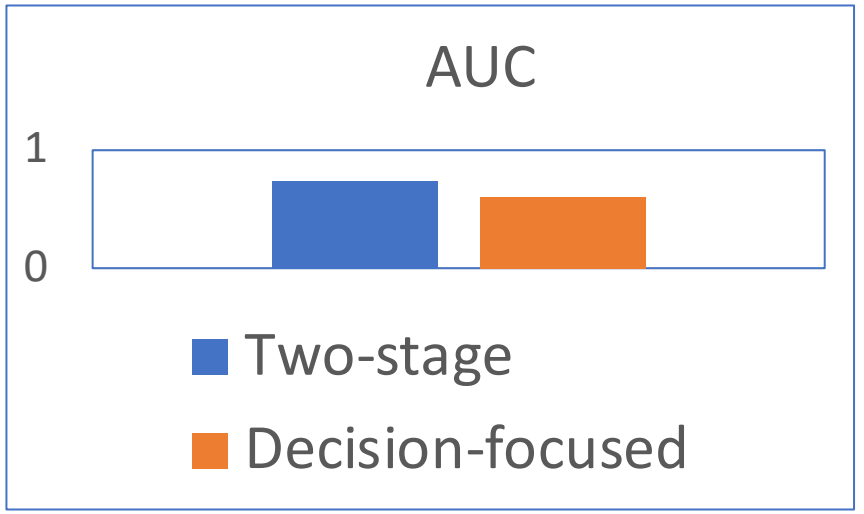
- Follow-on work improving treatment in Indian TB system
- In collaboration with Everwell (NGO)
- **Predict** if patients will miss daily dose
- **Optimize** health worker visits subject to knapsack constraints (LP)

- More in our paper [Killian et al, KDD 2019]



Application: Tuberculosis treatment

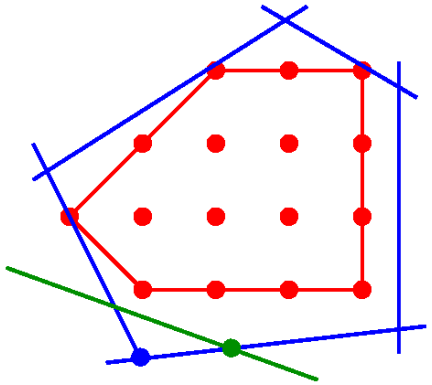




Less “accurate”, but +15% successful interventions!

Decision Focused Learning for Mixed Integer Programming (MIP) problems

- Many combinatorial problems do not have a nice relaxation-based algorithm
- But we know how to differentiate through LP optimization,



- Idea: cutting planes for MIP results in LP with added cuts
- Differentiate through Cutting-plane-generated LP for training

- At test time, obtain predictions and solve MIP with Branch-and-Bound

[Ferber, Wilder, Dilkina, Tambe, ArXiv, 2019]

Domains

USC

- **Portfolio Optimization**
  - Predict monthly rate of return (% return)
  - Optimize monthly return for portfolio
  - Limiting risk, sector exposure, transactions...
  - Data: SP500 (USA), DAX (Germany)
- **Diverse Bipartite Matching**
  - Predict match success probability
  - Optimize total number of successful matches
  - Matching constraints: each node matched at most once
  - Ensure min % of proposed matches are different/same type
  - Data: CORA citation network, nodes = papers, edges = citations
- **Energy Production Knapsack**
  - Predict energy prices
  - Optimize total revenue
  - Limit on number of time periods we can generate energy
  - Data: ICON Energy Scheduling Challenge

Results: decision quality at test time

Objective: monthly % increase for portfolio optimization (SP500 and DAX), number of pairs successfully matched for Matching (CORA), and value of items for Knapsack (Energy).

	SP500	DAX	Matching	Knapsack
MIPaaL	**2.79 ± 0.17**	**5.70 ± 0.68**	**4.80 ± 0.71**	**507.70 ± 0.471**
MIPaaL-Warm	1.09 ± 0.18	0.68 ± 1.01	2.14 ± 0.51	499.60 ± 0.566
MIPaaL-Hybrid	1.08 ± 0.15	0.74 ± 1.10	3.21 ± 0.73	503.36 ± 0.578
MIPaaL-1000	2.60 ± 0.16	4.39 ± 0.66	3.45 ± 0.71	506.34 ± 0.662
MIPaaL-100	1.25 ± 0.14	0.35 ± 0.63	2.57 ± 0.54	505.99 ± 0.621
RootLP (Wilder et al. 2019)	1.97 ± 0.17	-1.97 ± 0.69	3.17 ± 0.60	501.58 ± 0.662
TwoStage	1.19 ± 0.15	0.70 ± 1.46	3.42 ± 0.78	501.49 ± 0.523

- MIPaaL gives **2x monthly returns on SP500 and 8x on DAX**
- MIPaaL **improves the objective by 40.3% and 1.2% for Matching and Knapsack respectively**.
- MIPaaL outperforms all other variants considered.

USC

Good ML Loss != Good Solutions

	SP500		DAX		Matching		Knapsack	
	MSE	Corr	MSE	Corr	CE	AUC	MSE	Corr
MIPaaL	0.22 ± 0.043	$\mathbf{0.15 \pm 0.015}$	0.13 ± 0.017	$\mathbf{0.25 \pm 0.032}$	0.66 ± 0.009	$\mathbf{0.535 \pm 0.004}$	2774 ± 97.664	0.567 ± 0.002
MIPaaL-Warm	$\mathbf{0.11 \pm 0.010}$	-0.01 ± 0.010	$\mathbf{0.09 \pm 0.067}$	0.07 ± 0.030	0.52 ± 0.003	0.509 ± 0.003	4660 ± 72.008	0.593 ± 0.003
MIPaaL-Hybrid	$\mathbf{0.09 \pm 0.030}$	$\mathbf{0.13 \pm 0.013}$	0.13 ± 0.099	$\mathbf{0.26 \pm 0.026}$	0.55 ± 0.002	0.502 ± 0.004	3824 ± 82.828	0.608 ± 0.006
MIPaaL-1000	$\mathbf{0.12 \pm 0.020}$	$\mathbf{0.13 \pm 0.013}$	0.35 ± 0.010	$\mathbf{0.27 \pm 0.035}$	0.61 ± 0.010	0.506 ± 0.007	5821 ± 154.793	0.590 ± 0.005
MIPaaL-100	0.98 ± 0.089	0.12 ± 0.013	0.99 ± 0.060	$\mathbf{0.26 \pm 0.037}$	0.54 ± 0.013	0.503 ± 0.004	5801 ± 145.331	0.553 ± 0.007
RootLP (Wilder et al. 2019)	0.71 ± 0.178	$\mathbf{0.15 \pm 0.013}$	1.06 ± 0.137	$\mathbf{0.28 \pm 0.032}$	0.49 ± 0.007	0.513 ± 0.001	6267 ± 212.063	0.574 ± 0.002
TwoStage	$\mathbf{0.09 \pm 0.017}$	0.06 ± 0.011	$\mathbf{0.02 \pm 0.066}$	0.13 ± 0.032	$\mathbf{0.39 \pm 0.004}$	0.514 ± 0.005	$\mathbf{684 \pm 15.568}$	$\mathbf{0.649 \pm 0.002}$

Transfer Learning

- Learn on one distribution of assets (30 SP assets) and test on another (30 other SP assets and 30 DAX assets), keeping the MIP size the same
- Learn on one size of MIPs (number of assets available, 30 SP) and test on larger MIPs (with more assets to choose from 50-500 SP)

		SP-30[b]	DAX	SP-50	SP-100	SP-200	SP500
Decision Quality	MIPaaL	**2.02 ± 0.48**	**2.77 ± 0.40**	**1.93 ± 0.13**	**2.27 ± 0.11**	**2.17 ± 0.48**	**2.26 ± 0.37**
	RootLP	1.81 ± 0.44	1.74 ± 0.43	1.50 ± 0.09	1.58 ± 0.08	1.82 ± 0.41	1.90 ± 0.29
	TwoStage	0.71 ± 0.04	0.82 ± 0.54	1.58 ± 0.13	1.22 ± 0.09	1.50 ± 0.58	1.11 ± 0.35
ML Loss	MIPaaL	4.81 ± 8.59	4.59 ± 8.80	5.42 ± 3.16	5.42 ± 2.37	5.25 ± 1.83	5.43 ± 1.67
	RootLP	5.14 ± 1.02	5.39 ± 1.04	4.73 ± 3.17	4.88 ± 2.58	4.81 ± 1.91	4.83 ± 1.56
	TwoStage	**0.08 ± 0.05**	**0.07 ± 0.03**	**0.08 ± 0.02**	**0.07 ± 0.01**	**0.08 ± 0.01**	**0.08 ± 0.01**

Decision-Focused Learning

- No need to silo out ML vs Optimization tasks
- When data is scarce, we want predictions to be accurate where it matters most for decision making
- Marrying predictive and prescriptive tasks in an **end-to-end system**

ML ⟺ Combinatorial Optimization

- Exciting and growing research area
- Design discrete optimization algorithms with learning components
- Learning methods that incorporate the combinatorial decision making they inform

USC

ExxonMobil

NSF

Microsoft AI for Earth

Georgia Tech

DEPARTMENT of the INTERIOR
SE CLIMATE SCIENCE CENTER

Thank you!