Machine Learning for Optimization

Ellen Vitercik

Stanford University

Machine learning for optimization

Optimization underpins critical societal challenges

- E.g., decarbonizing energy, allocating scarce medical resources, …
- These are **massive**, **discrete**, **NP-hard** problems

Uncertainty compounds difficulty (e.g., demand in power grids)

Research goals:

- ML to exploit underlying structure of optimization problems
  - Leverage ML to learn from historical patterns & problem stability
- Optimization algorithms that adapt to uncertainty

Learning from structured problems

Why machine learning?

- Real-world optimization problems evolve over time
- But they rarely change arbitrarily; **underlying structure persists**

Example (dynamic yet structured domains):

- Shipping company re-optimizes routes as demand, traffic shift
- Yet the road network remains fixed

Historical problems contain **rich structural information**

- If learned effectively, structure can accelerate & improve future decisions

Why now?

Decision-aware learning for optimization at a critical stage

- Advances in computational **infrastructure** and **pretrained models**
- Now becoming feasible to embed ML within **large-scale solvers**

Example: **integer programming**

- Tons of applications across planning, scheduling, routing, …
- Many different ways to incorporate learning into solving
- Solving is very difficult, so ML can make a **huge difference**

ML for (NP-hard) optimization: Key challenges

This talk: isolate and address key challenges across two case studies

ML approaches to (NP-hard) optimization **rarely work out of the box**:
must be **aligned** to the algorithmic task at hand
[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq

E.g., Vinyals et al., NeurIPS'15 [introduced **pointer networks** motivated by TSP]; Bello et al., '17; …

ML for (NP-hard) optimization: Key challenges

This talk: isolate and address key challenges across two case studies

ML approaches to (NP-hard) optimization **rarely work out of the box**:
must be **aligned** to the algorithmic task at hand
[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq, GNNs

E.g., survey by Cappart et al., JMLR'23

ML for (NP-hard) optimization: Key challenges

This talk: isolate and address key challenges across two case studies

ML approaches to (NP-hard) optimization **rarely work out of the box**:
must be **aligned** to the algorithmic task at hand
[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq, GNNs, transformers, …

E.g., Garg et al., NeurIPS'22; Akyürek et al., ICLR'23; …

ML for (NP-hard) optimization: Key challenges

This talk: isolate and address key challenges across two case studies

ML approaches to (NP-hard) optimization **rarely work out of the box**:
must be **aligned** to the algorithmic task at hand
[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq, GNNs, transformers, …
- **Integration:** integrate with existing method or train end-to-end method?

E.g., **integer programming solvers**: Khalil et al., AAAI'16; Gasse et al., NeurIPS'19; …

ML for (NP-hard) optimization: Key challenges

This talk: isolate and address key challenges across two case studies

ML approaches to (NP-hard) optimization **rarely work out of the box**:
must be **aligned** to the algorithmic task at hand
[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq, GNNs, transformers, …
- **Integration:** integrate with existing method or train end-to-end method?

E.g., **neural algorithmic reasoning**: Veličković et al., ICLR'20; Xu et al., ICLR'21; …

ML for (NP-hard) optimization: Key challenges

This talk: isolate and address key challenges across two case studies

ML approaches to (NP-hard) optimization **rarely work out of the box**:
must be **aligned** to the algorithmic task at hand
[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq, GNNs, transformers, …
- **Integration:** integrate with existing method or train end-to-end method?
- **Supervision:** how to supervise on (NP-hard) algorithmic problems?
- **Robustness:** can we provide any guarantees?
  [e.g., book chapters by Balcan, '20; Mitzenmacher, Vassilvitskii, '20; …]

Outline

1. Introduction
2. **Mathematical programming** [Lawless, Li, Wikum, Udel, Vitercik, CPAIOR'25]
    i. Background:
        a. Linear and integer programming
        b. ML for integer programming
    ii. LLMs for mathematical optimization
3. Uncertainty quantification in scheduling [Shen, Wikum, Vitercik, ICML'25]
4. Conclusions

Outline

1. Introduction
2. Mathematical programming [Lawless, Li, Wikum, Udel, Vitercik, CPAIOR'25]
    i. **Background:**
        a. **Linear and integer programming**
        b. ML for integer programming
    ii. LLMs for mathematical optimization
3. Uncertainty quantification in scheduling [Shen, Wikum, Vitercik, ICML'25]
4. Conclusions

Linear programming

Linear programming (LP) is a central topic in optimization

Provides a powerful tool for modeling many applications

Tons of attention over past two decades due to:

- **Applicability:** Many real-world applications can be modeled via LPs
- **Solvability:** Efficient techniques for solving large-scale problems

Basic components of an LP

Each optimization problem consists of 3 elements:

- **Decision variables**: describe our choices that are under our control;
- **Objective function**: Criterion that we wish to minimize (e.g., cost)
  or maximize (e.g., profit)
- **Constraints**: Limitations restricting our choices for decision variables

"Linear programming" refers to an optimization problem where:

- The **objective function** is linear
- Each **constraint** is a linear inequality or equality

Example linear program

Machine scheduling

maximize $28x + 30y$

//Firm obtains $28 per unit of item P they produce and $30 per unit of item Q

subject to $50x + 24y \leq 2400$

//Each unit of P (resp., Q) requires 50 min (resp., 24 min) on machine A, which is available for 2400 min

$30x + 33y \leq 2100$

//Each unit of P (resp., Q) requires 30 min (resp., 33 min) on machine B, which is available for 2100 min

$x \geq 0$ and $y \geq 0$

//Must produce a non-negative number of units

Example linear program

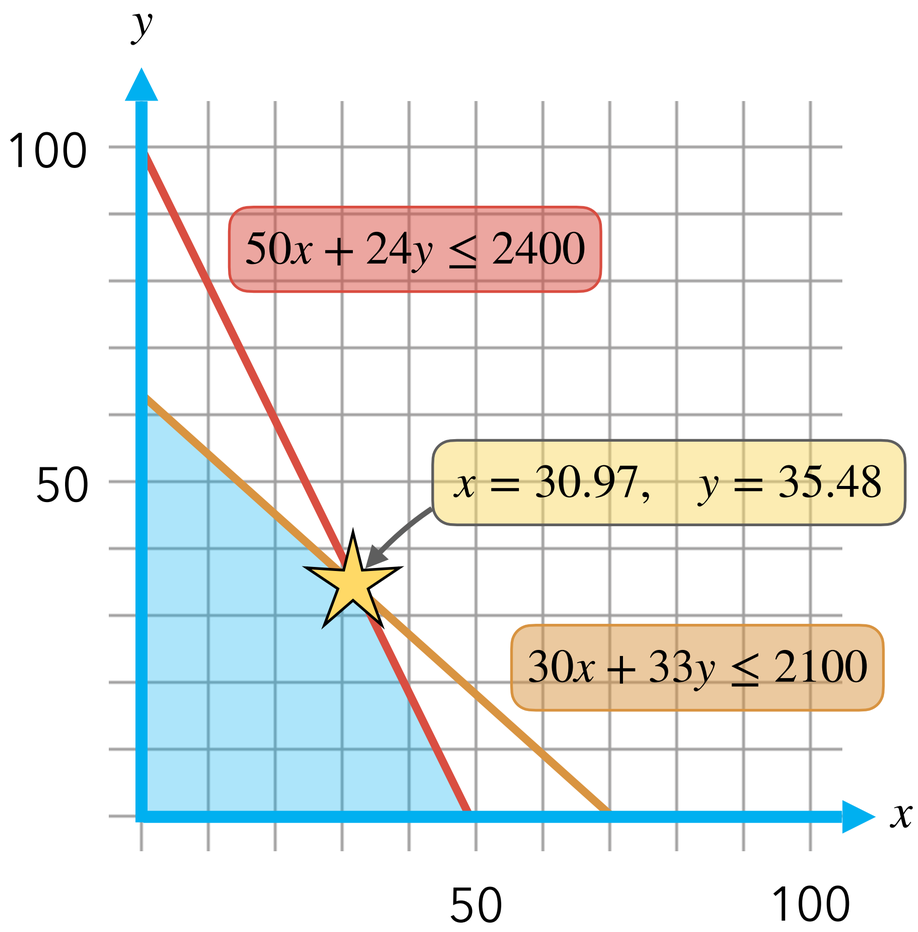
Machine scheduling

maximize $28x + 30y$

subject to $50x + 24y \leq 2400$

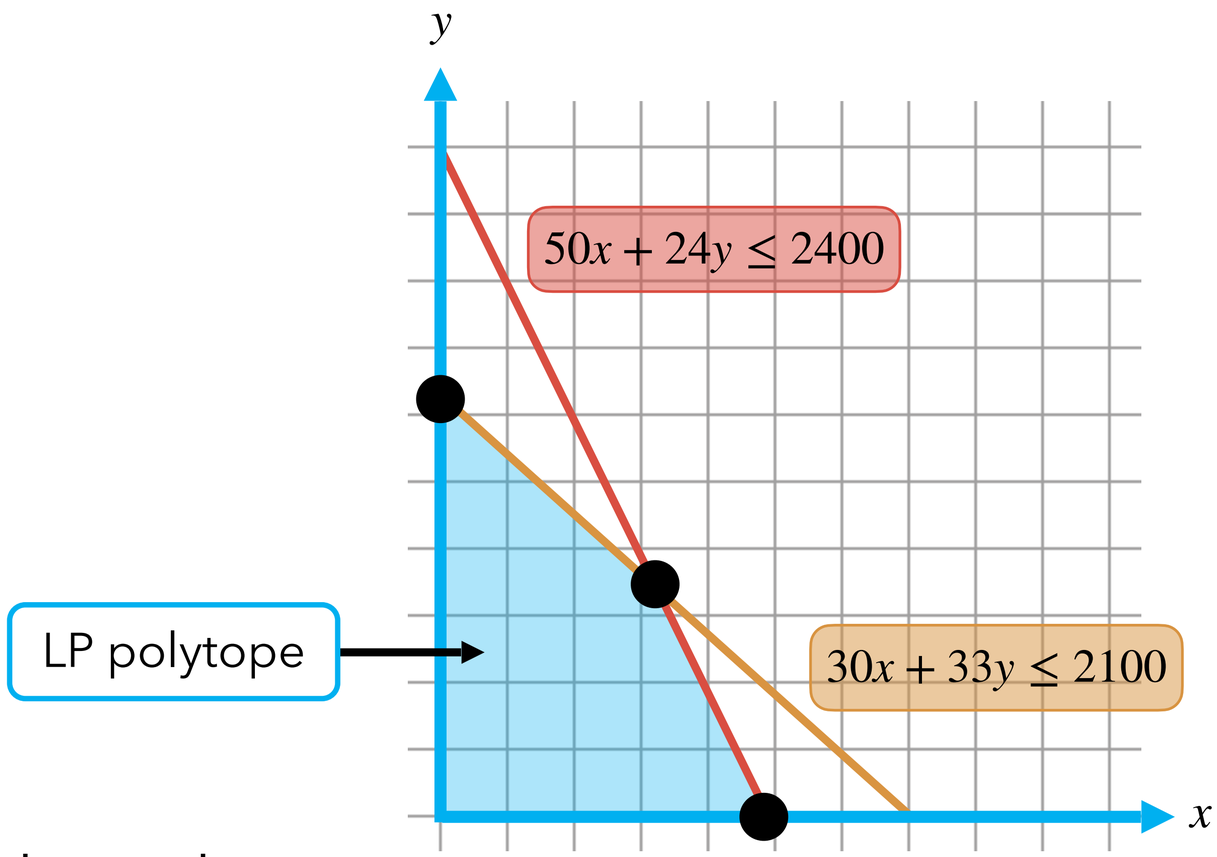
$30x + 33y \leq 2100$

$x \geq 0$ and $y \geq 0$



Example linear program

Machine scheduling



Fact: LP optimal solution is always at a vertex

Integer programming (IP)

What if the decision variables must be integral?

maximize $28x + 30y$

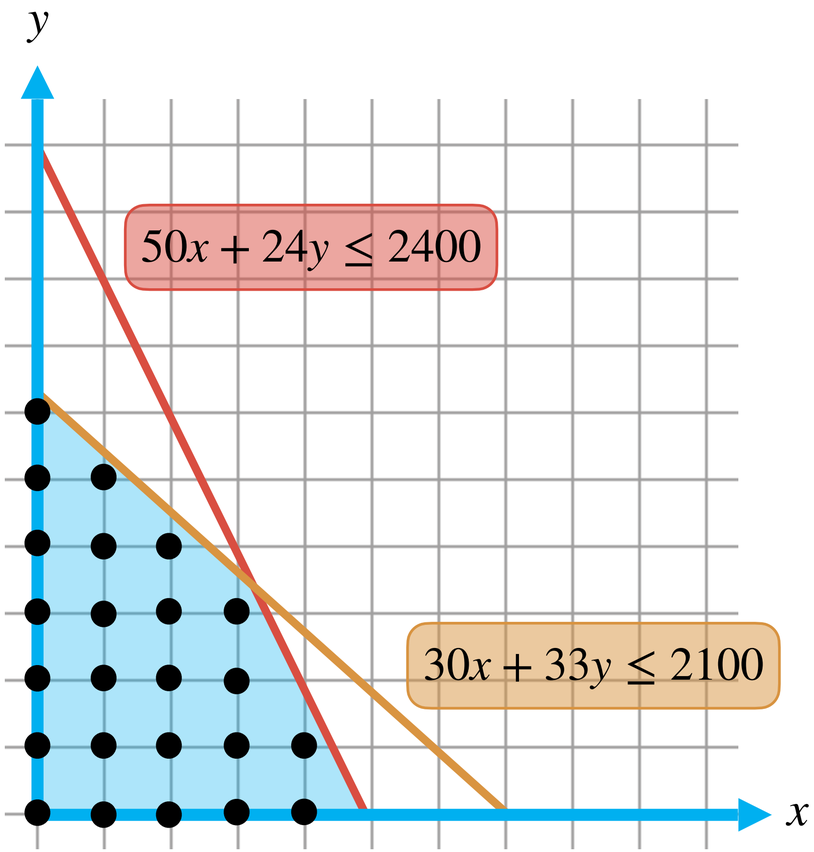
subject to $50x + 24y \leq 2400$

$30x + 33y \leq 2100$

$x \geq 0$ and $y \geq 0$

$x, y \in \mathbb{Z}$

Integer programming is NP-complete



LP relaxations

Integer program

maximize $\boldsymbol{c}^T\boldsymbol{x}$

subject to $A\boldsymbol{x} \leq \boldsymbol{b}$

$x_i \in \mathbb{Z}$ for all $i \in I$

NP-hard

Linear programming (LP) relaxation

maximize $\boldsymbol{c}^T\boldsymbol{x}$

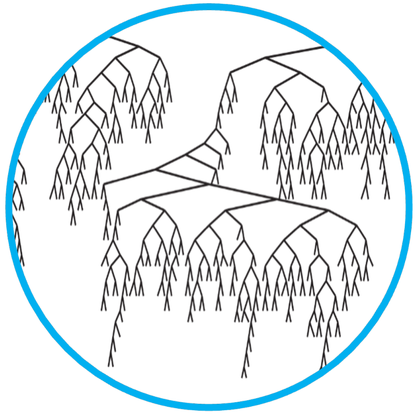
subject to $A\boldsymbol{x} \leq \boldsymbol{b}$

~~$x_i \in \mathbb{Z}$ for all $i \in I$~~

Poly-time solvable

Integer programming solvers

Uses guidance from LP relaxations to guide search



- Explore **search tree** of restricted MILP subproblems
  - Each node adds bounds on integer variables
- Solve **LP relaxation** to upper bound subproblem's integer-feasible solution
  - If integer-feasible: incumbent solution
- **Branch** by choosing a variable and splitting its domain ← **Variable selection policy**
- **Prune** nodes whose bound can't beat incumbent
- Terminate when all nodes **pruned** or **proven optimal**

Integer programming solvers

Uses guidance from LP relaxations to guide search

Integer program

maximize $\boldsymbol{c}^T\boldsymbol{x}$

subject to $A\boldsymbol{x} \leq \boldsymbol{b}$

$x_i \in \mathbb{Z}$ for all $i \in I$

Linear programming (LP) relaxation

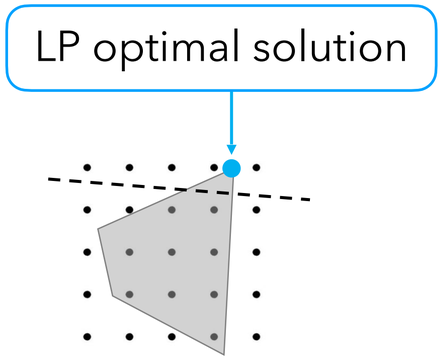
maximize $\boldsymbol{c}^T\boldsymbol{x}$

subject to $A\boldsymbol{x} \leq \boldsymbol{b}$

~~$x_i \in \mathbb{Z}$ for all $i \in I$~~

Cutting planes (CPs) are additional constraints that:

- Separate LP optimal solution
- Don't separate any integer point



Many different families of CPs; which to use when?

Outline

1. Introduction
2. Mathematical programming [Lawless, Li, Wikum, Udel, Vitercik, CPAIOR'25]
   i. **Background:**
      a. Linear and integer programming
      b. ML for integer programming
   ii. LLMs for mathematical optimization
3. Uncertainty quantification in scheduling [Shen, Wikum, Vitercik, ICML'25]
4. Conclusions

ML to speed up MILP solvers

- Modern MILP solvers expose **hundreds** of parameters
- Defaults are rarely optimal: instance families vary widely in structure
- Small changes can yield **large runtime** differences
- **Many** ways to integrate ML into solvers
  - Potential for ML to provide **significant** speedups over defaults

Background: General setup

ML for MILP solvers

- Define a parameterized solver $A(\theta)$. E.g.:
  - θ are parameters exposed by Gurobi
  - θ are parameters of a neural network embedded in solver
- Specify distribution D over MILPs z (models **application domain**)
- Choose a **performance metric** $c(z, \theta)$; e.g., runtime
- **Ultimate goal:** minimize $\mathbb{E}_{z \sim D}\left[c(z, \theta)\right]$ (proxy of *future* cost on unseen MILPs)
  - Can learn *offline* θ or *instance-aware* $\theta(z)$ configuration

Background: Blackbox algorithm configuration

- Early work: treat solver (largely) as a black-box; learn from evaluations
- Small subset of examples:
  - **ParamILS**: iterated local search over parameter settings
    Hutter, Hoos, Leyton-Brown, Stützle, JAIR'09
  - **SMAC**: model-based search with surrogate predictions
    Hutter, Hoos, Leyton-Brown, LION'11
  - **Portfolio-based algorithm selection**
    Lobjois, Lemaître, AAAI'98; Gomes, Selman, AI'01; Xu, Hoos, Leyton-Brown, AAAI'10; Kadioglu et al., ECAI'10, Sandholm, Handbook of Market Design'13

Background: Learning solver components

Next gen: don’t treat solver as blackbox; adapt to solver components. E.g.:

Cut selection	Variable selection	Node selection
Tang et al., ICML’20	Khalil et al., AAAI’16	He et al., NeurIPS’14
Balcan, Prasad, Sandholm, **Vitercik**, NeurIPS’21, NeurIPS’22	Alvarez et al., INFORMS JoC’17	Labassi et al., NeurIPS’22
Paulus et al., ICML’22	Balcan, Dick, Sandholm, **Vitercik**, ICML’18	Zhang et al., ICLR’25
Wang et al., ICLR’23	Gasse et al., NeurIPS’19	…
Li et al., NeurIPS’23	Gupta et al., NeurIPS’20	
Deza, Khalil, IJCAI’23	Zarpellon et al., AAAI’21	
Ling et al., AAAI’24	Scavuzzo et al., NeurIPS’22	
Cheng, Basu, NeurIPS’24	…	
Cheng et al., NeurIPS’24		
…		

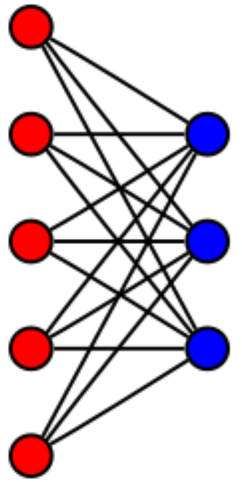
Background: Learning solver components

Next gen: don't treat solver as blackbox; adapt to solver components. E.g.:

Cut selection	Variable selection	Node selection
Tang et al., ICML'20	Khalil et al., AAAI'16	He et al., NeurIPS'14
Balcan, Prasad, Sandholm, **Vitercik**,	Alvarez et al., INFORMS JoC'17	Labassi et al., NeurIPS'22
NeurIPS'21, NeurIPS'22	Balcan, Dick, Sandholm, **Vitercik**, ICML'18	Zhang et al., ICLR'25
Paulus et al., ICML'22	**Gasse et al., NeurIPS'19**	…
Wang et al., ICLR'23	Gupta et al., NeurIPS'20	
Li et al., NeurIPS'23	Zarpellon et al., AAAI'21	
Deza, Khalil, IJCAI'23	Scavuzzo et al., NeurIPS'22	
Ling et al., AAAI'24	…	
Cheng, Basu, NeurIPS'24		
Cheng et al., NeurIPS'24		
…		

Graph neural networks for variable selection

- **Key idea:** encode B&B state as variable-constraint **bipartite graph**
  - Use bipartite **graph neural network** as a variable selection policy
- Training: behavioral cloning of strong branching (expensive **gold standard**)
- Integrated in SCIP; four NP-hard benchmarks
- Results:
  - Best imitation accuracy among ML baselines
  - Generally faster than SCIP default; good size generalization



Gasse, Chételat, Ferroni, Charlin, Lodi, NeurIPS'19

Outline

1. Introduction
2. Mathematical programming [Lawless, Li, Wikum, Udel, Vitercik, CPAIOR'25]
    i. Background
    ii. LLMs for mathematical optimization
3. Uncertainty quantification in scheduling [Shen, Wikum, Vitercik, ICML'25]
4. Conclusions

Key challenge

Conventional data-driven approaches require a lot of compute

Conventional data-driven configuration pipeline:

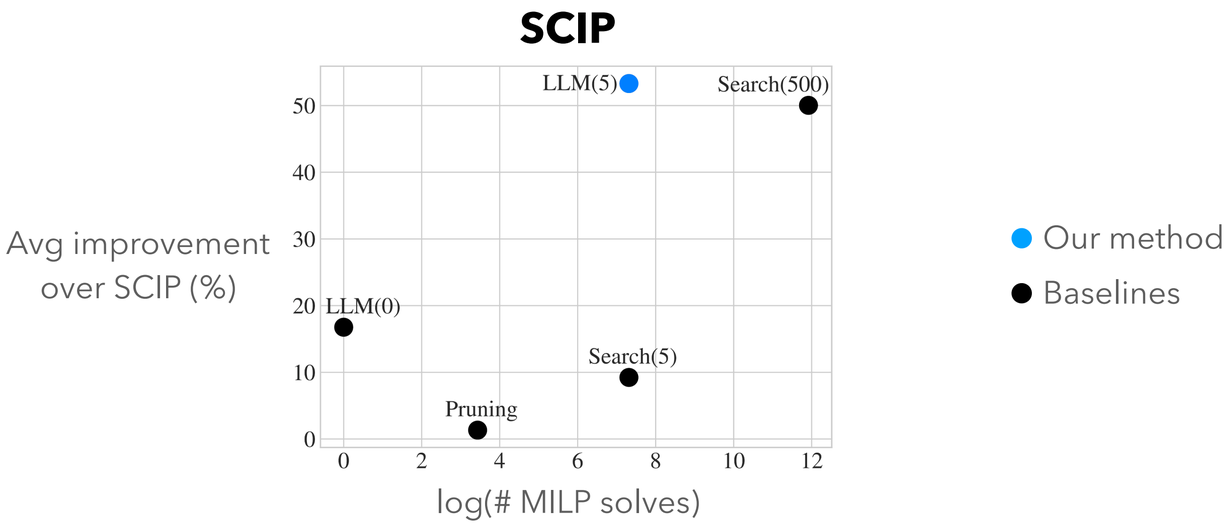
1. Gather a **training set** of MILPs
2. Find **configuration(s) with good average runtime** over training set
3. Hope for a good runtime from the same application (or distribution)

Key challenge: Evaluating one configuration's average runtime requires **solving** every MILP in the training set using that configuration

Key question: Can we generate problem-specific cutting plane configurations with **little to no historical data and compute**?

Our contributions

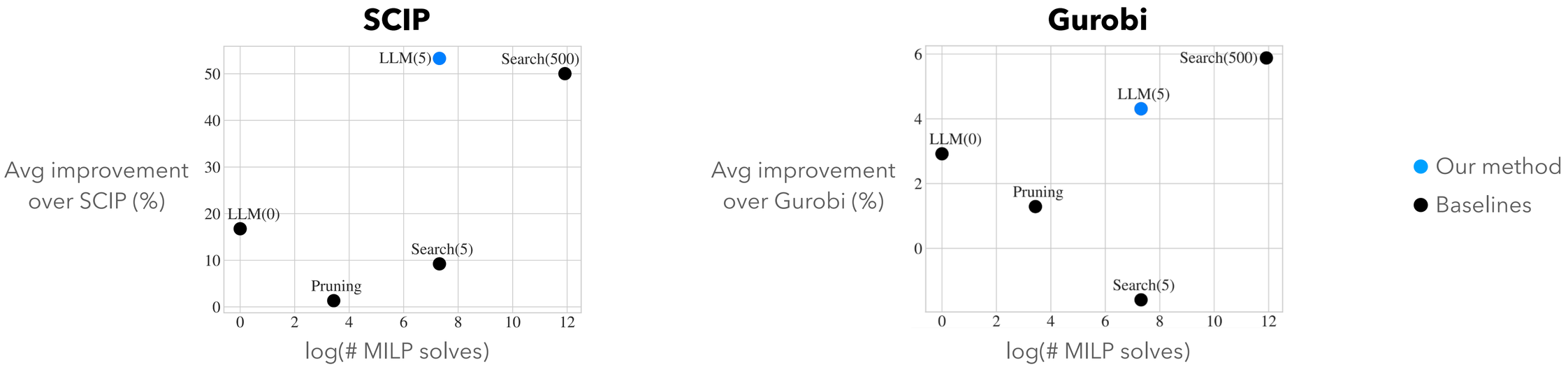
- **First LLM-based framework** to configure MILP solvers
- **Consistent improvement** over solver default (SCIP and Gurobi)
  - Pareto-optimal compared to baseline methods



Lawless, Li, Wikum, Udell, **V**, CPAIOR'25

Our contributions

- **First LLM-based framework** to configure MILP solvers
- **Consistent improvement** over solver default (SCIP and Gurobi)
  - Pareto-optimal compared to baseline methods



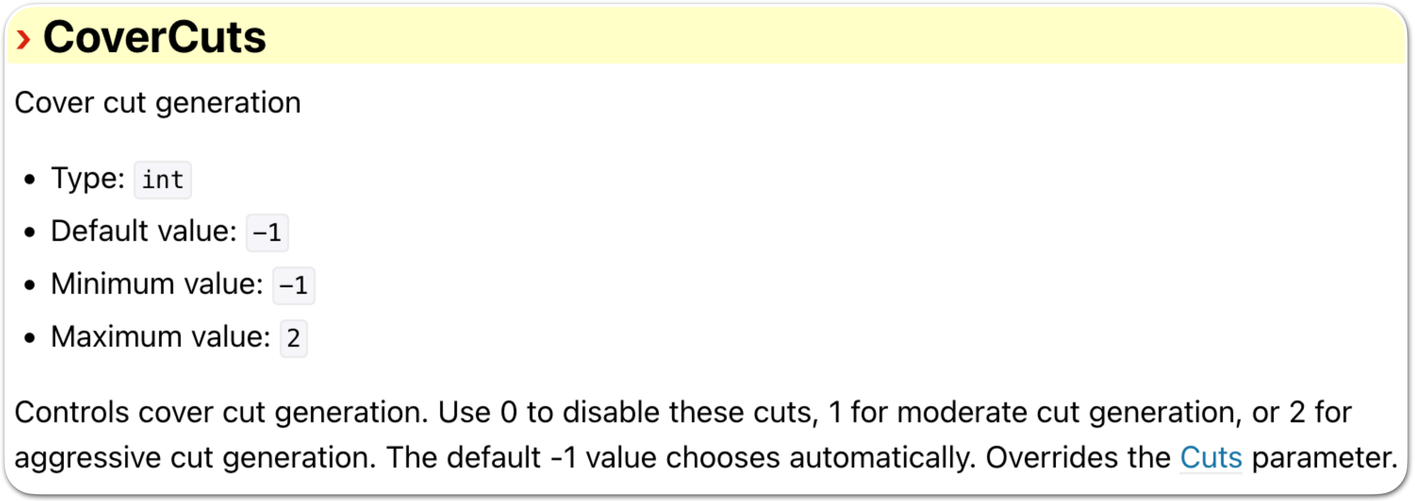
Lawless, Li, Wikum, Udell, **V**, CPAIOR'25

Why LLMs

- LLMs are powerful, but they can’t do everything
- They are good at **information retrieval**
- There’s a **rich literature** on cutting planes

Challenges to using LLMs

- LLM output can be highly **unstable**
- Cutting plane separators are **solver-specific**
  - Details of solver separators are not always available



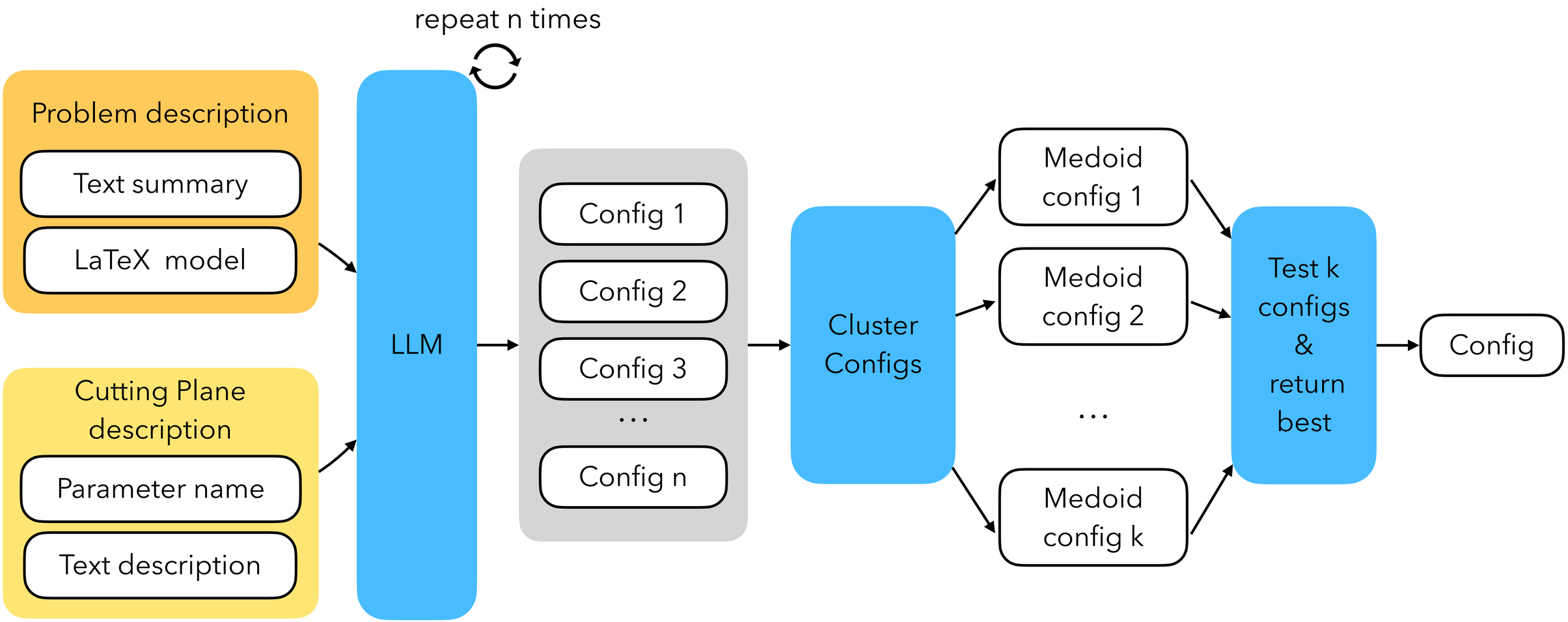
› CoverCuts

Cover cut generation

- Type: `int`
- Default value: `-1`
- Minimum value: `-1`
- Maximum value: `2`

Controls cover cut generation. Use 0 to disable these cuts, 1 for moderate cut generation, or 2 for aggressive cut generation. The default -1 value chooses automatically. Overrides the Cuts parameter.

LLM for cold-start configuration pipeline



Experimental set-up

Baselines and our method

- **Pruning**: turns off all CPs not used while solving validation set
  - Use the default setting for other CPs
- **Search**(d): sample d candidate configurations uniformly at random
  - Use the one with best median performance on validation set
- **Zero-shot**: use medoid of the largest cluster
- **Cold-start**(k):
  1. Run k medoids clustering
  2. Select the best performing medoid on the validation set

Experimental set-up

Datasets, model

	Dataset	# vars	# constrs
Classic MILP families	Binary packing	300	300
	Capacitated facility location	100	100
	Combinatorial auction	100	500
	Maximum independent set	500	1088
	Max cut	54	134
	Packing	60	60
	Set cover	500	250
Complex real-world MILP families	Load balancing	64340	61000
	Middle-mile consolidation network design (MM)	569	248

Model: GPT-4o

30 validation instances
100 evaluation instances

Evaluation metric:
% improvement over default solve time

Empirical results

Cold-start(5) yields 6–71% faster runtimes than SCIP's default

Problem	Pruning	Search(5)	Search(500)	Zero-shot	Cold-start(5)
Bin. pack.	1.33	9.23	39.3	16.76	38.35
Cap. fac.	-0.64	9.57	2.72	7.61	26.12
Comb. auc.	1.96	58.1	64.01	21.06	63.59
Ind. set	2.07	26.95	67.01	21.6	71.95
Max. cut	-2.18	17.72	69.63	71.43	71.01
Pack.	15.87	-13.81	24.49	15.09	25.51
Set cov.	6.62	-10.04	61.08	61.72	61.74
Load bal.	0.08	-150.01	-50.02	0.0	6.37
MM	-0.12	-8.83	50.03	-6.52	53.3

Empirical results

Cold-start(5) yields 6–71% faster runtimes than SCIP's default

Problem	Pruning	Search(5)	Search(500)	Zero-shot	Cold-start(5)
Bin. pack.	1.33	9.23	39.3	16.76	38.35
Cap. fac.	-0.64	9.57	2.72	7.61	26.12
Comb. auc.	1.96	58.1	64.01	21.06	63.59
Ind. set	2.07	26.95	67.01	21.6	71.95
Max. cut	-2.18	17.72	69.63	71.43	71.01
Pack.	15.87	-13.81	24.49	15.09	25.51
Set cov.	6.62	-10.04	61.08	61.72	61.74
Load bal.	0.08	-150.01	-50.02	0.0	6.37
MM	-0.12	-8.83	50.03	-6.52	53.3

By testing only 5 configs, we match/beat Search(500) on **all instances**

Empirical results

Cold-start(5) yields 6–71% faster runtimes than SCIP's default

Problem	Pruning	Search(5)	Search(500)	Zero-shot	Cold-start(5)
Bin. pack.	1.33	9.23	39.3	16.76	38.35
Cap. fac.	-0.64	9.57	2.72	7.61	26.12
Comb. auc.	1.96	58.1	64.01	21.06	63.59
Ind. set	2.07	26.95	67.01	21.6	71.95
Max. cut	-2.18	17.72	69.63	71.43	71.01
Pack.	15.87	-13.81	24.49	15.09	25.51
Set cov.	6.62	-10.04	61.08	61.72	61.74
Load bal.	0.08	-150.01	-50.02	0.0	6.37
MM	-0.12	-8.83	50.03	-6.52	53.3

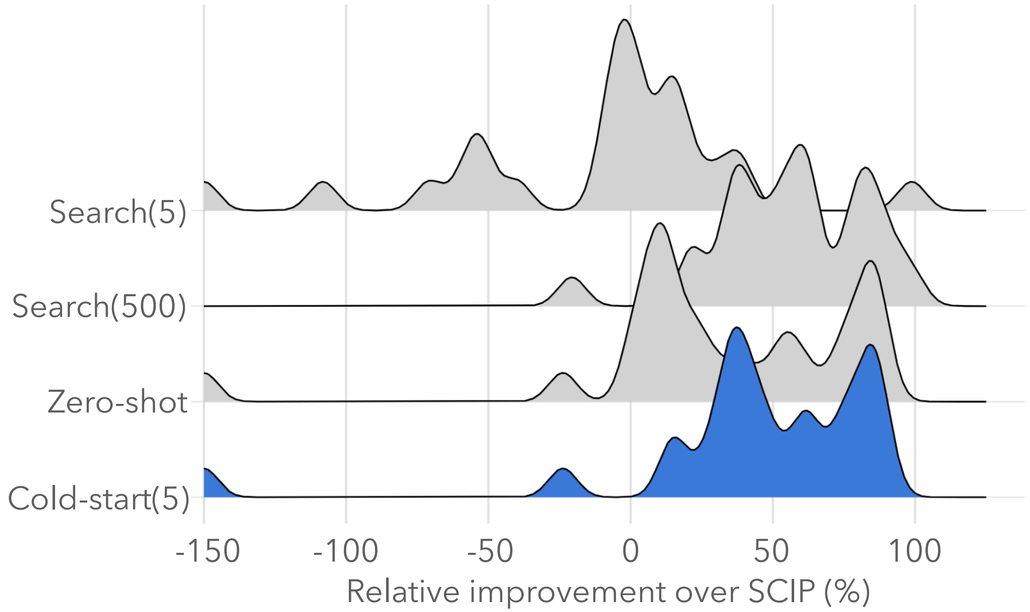
By testing only 5 configs, we match/beat Search(500) on **all instances**

Without solving **any** MILPs, we match/beat Search(500) on **4/9 instances**

On out-of-distribution instances

25 families of problem from MILP-Evolve dataset [Li et al. ICLR'25]

- New dataset, "evolving" pipeline for generating new MILP families
- Designed to be highly diverse to mimic real-world optimization scenarios



Ablations

Our design choices are robust

Setting	Ind. set	Max cut	Bin. pack.	MM
Ours	71.95	71.01	38.35	53.3
Disable cutting planes	-14.96	71.25	30.43	-150
No CP text descr.	72.27	71.49	16.85	9.29
Ensembling strategies				
Average configuration	20.65	71.24	17.52	-11.08
Mode configuration	21.08	71.44	18.11	-12.63
Smallest configuration	20.83	70.91	17.42	-4.74

Disabling CPs can reduce performance

Our **CP descriptions** boost performance

k-medoids outperforms simpler heuristics

Lawless, Li, Wikum, Udell, **V**, CPAIOR'25

Recap

Can we use LLMs to configure MILP solvers with minimal training data?

- New **LLM-based framework** to configure cutting plane separators
- Finds high-performing configuration by **solving only a few MILPs**
- **Ensembling strategy** to build portfolio of high-performing configurations
- Requires **no custom solver interface**
- Competitive with existing configuration approaches
  but only requires a **fraction of the training data and computation time**

Outline

1. Introduction
2. Mathematical programming [Lawless, Li, Wikum, Udel, Vitercik, CPAIOR'25]
3. **Uncertainty quantification in scheduling** [Shen, Wikum, Vitercik, ICML'25]
4. Conclusions

Decision-making under uncertainty

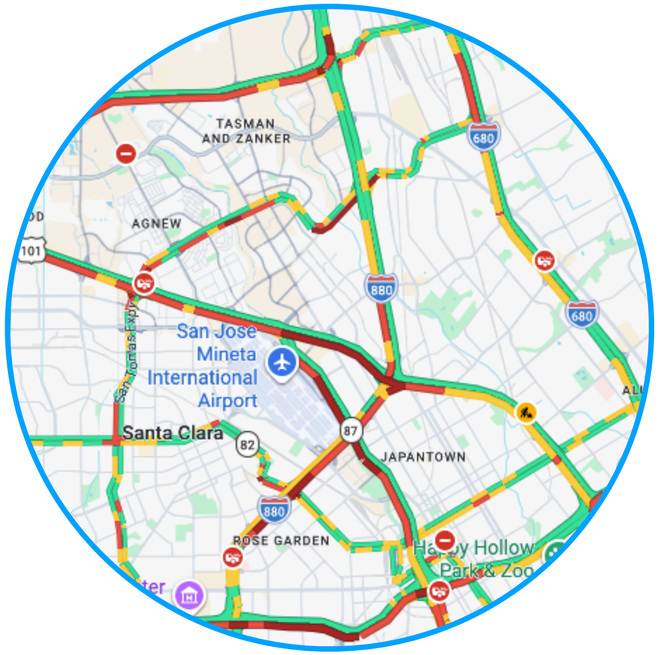
In practice, many aspects of inputs are **unknown** a priori

- E.g., future traffic or demand in routing

However, we often have **rich historical data**

- ML can help predict unknown aspects of inputs
- Research area: **Algorithms with predictions**

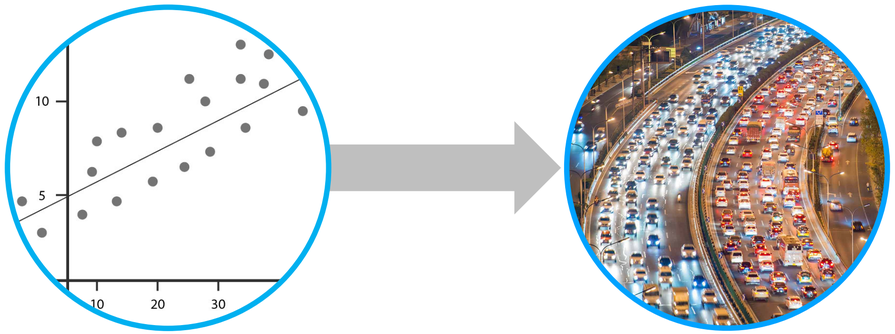
  [e.g., book chapter by Mitzenmacher, Vassilvitskii, ‘20]



Algorithms and prediction uncertainty

Challenge: prediction **errors can amplify** in decision-making

Don’t blindly trust predictions



Insight: ML models can estimate uncertainty **automatically**

- Examples: calibration and conformal predictions [Sun et al. ’24]
- Well-defined, statistical notion of whether a prediction can be trusted

Our contributions

Demonstrate calibration's utility for discrete optimization tasks

Two case-studies:

Online rent-or-buy problems

- E.g., cloud vs equipment purchase, subscription vs lifetime licenses, …
- Algorithm with guarantees that improve with accuracy and calibration error

Online job scheduling

- E.g., processing radiology diagnostic images
- Calibrated predictions yield better schedules than prior work [Cho et al., '22]

Validate methods on real-world datasets

Calibration (binary target)

- Random variables (X, Y) with support $\mathcal{X} \times \{0,1\}$
- $f : \mathcal{X} \to [0,1]$ is **calibrated** if $\mathbb{P}[Y = 1 \mid f(x) = p] = p$
- E.g., rain prediction: Weather is rainy
  - 50% of days where $f(X) = 0.5$
  - 100% of days where $f(X) = 1$

- Let $T(X) = \mathbb{P}[Y = 1 \mid f(X)]$ *(equals $f(X)$ if perfectly calibrated)*

$$\underbrace{\mathbb{E}\left[\left(Y - f(X)\right)^2\right]}_{\ell_2 \text{ error}} = \underbrace{\text{Var}(Y)}_{\text{Uncertainty}} - \underbrace{\text{Var}(T(X))}_{\text{Sharpness}} + \underbrace{\mathbb{E}\left[\left(T(X) - f(X)\right)^2\right]}_{\text{Calibration error}}$$

- Calibrated, **unsharp**: $f(X) = \mathbb{P}[Y = 1]$ for all X
- Calibrated, **sharp**: $f(X) = \mathbb{P}[Y = 1 \mid X]$ for all X

Case study: Online job scheduling

Motivation [Cho et al. ’22]**: radiologist’s scheduling problem**

- One radiologist, backlog of imaging cases
- True case **urgency** initially **unknown**
  - Partial review reveals true urgency
- ML model provides **noisy** urgency probabilities
- Must choose processing order under uncertainty
- Goal: develop a simple, **robust scheduling policy**

Case study: Online job scheduling

- 1 machine to process n unit-length jobs
- Each job i has **unknown** high ($y_i = 1$) or low ($y_i = 0$) **priority**
  - Processing a θ-fraction of a job reveals its priority
- Jobs can be stored after partial processing
- Objective: Minimize weighted sum of completion times

$$\sum C_i \cdot \omega_{y_i}$$

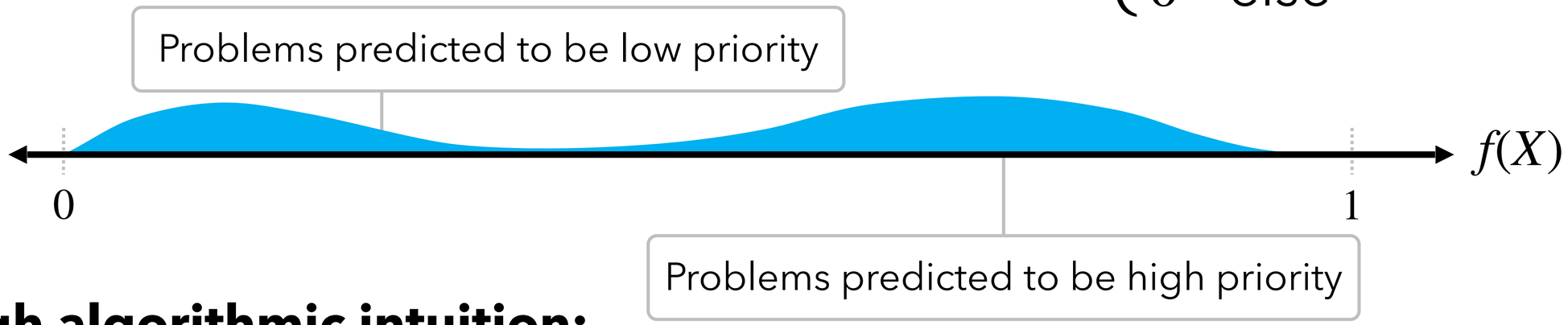
Cost per unit delay, with $\omega_1 > \omega_0 > 0$

Completion time of job i

Shen, **Vitercik**, Wikum, ICML'25

Algorithmic intuition

Given job features X, predictor $f(X) \in [0,1]$ of $Y = \begin{cases} 1 & \text{if job is high priority} \\ 0 & \text{else} \end{cases}$



Rough algorithmic intuition:

Run jobs predicted to be high priority first, then low priority later, preemptively

Start new job if discover current is low-priority

Shen, **Vitercik**, Wikum, ICML'25

Importance of predictor sharpness

Key insight: **interchanges** are the primary source of **regret**

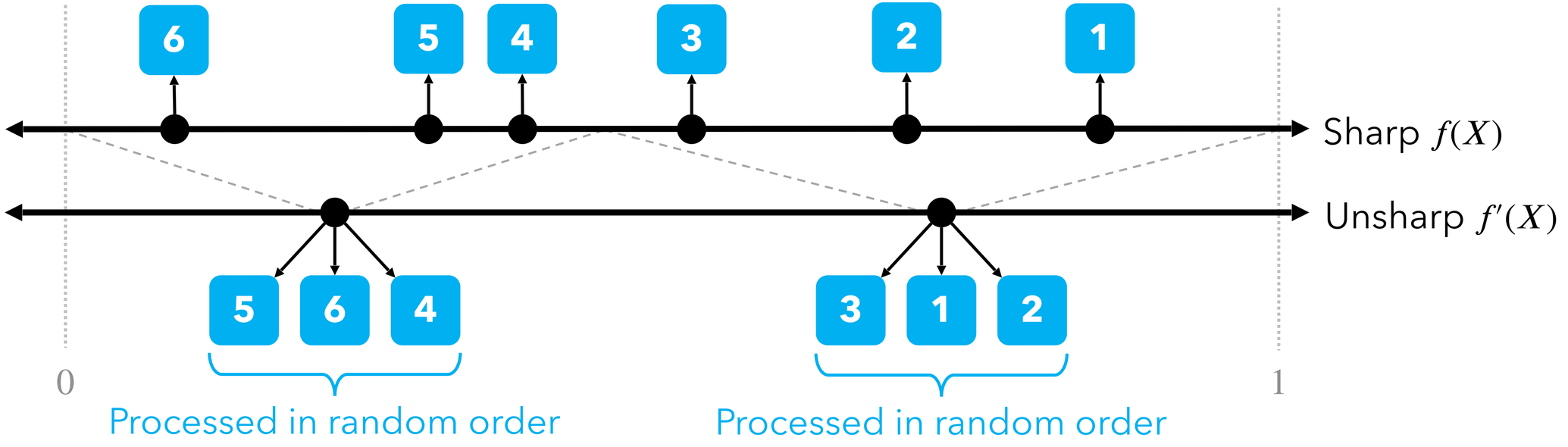
Weighted sum of completion times compared to optimal in hindsight

Shen, **Vitercik**, Wikum, ICML'25

Importance of predictor sharpness

Key insight: **interchanges** are the primary source of **regret**

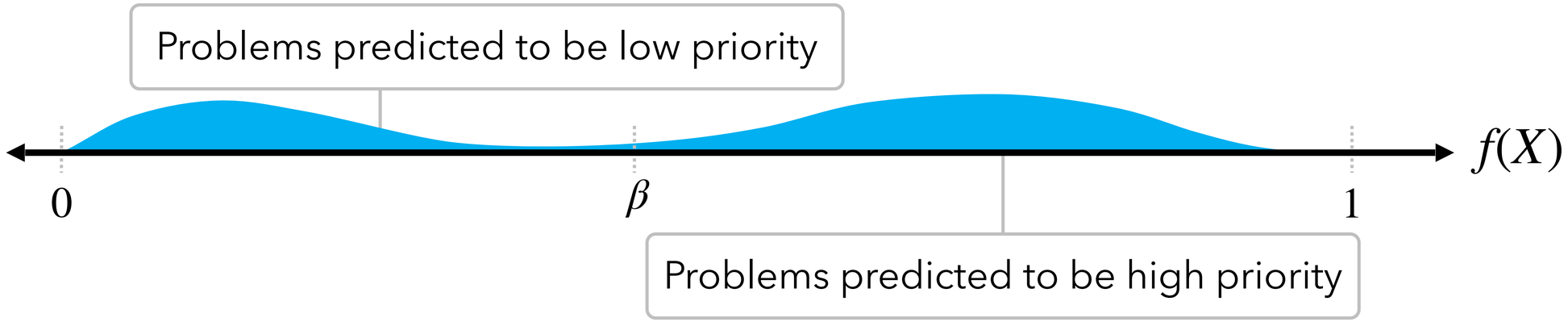
- Low priority job (partially) processed before high priority job
- **Sharp predictors** lead to **fewer interchanges**



Shen, **Vitercik**, Wikum, ICML'25

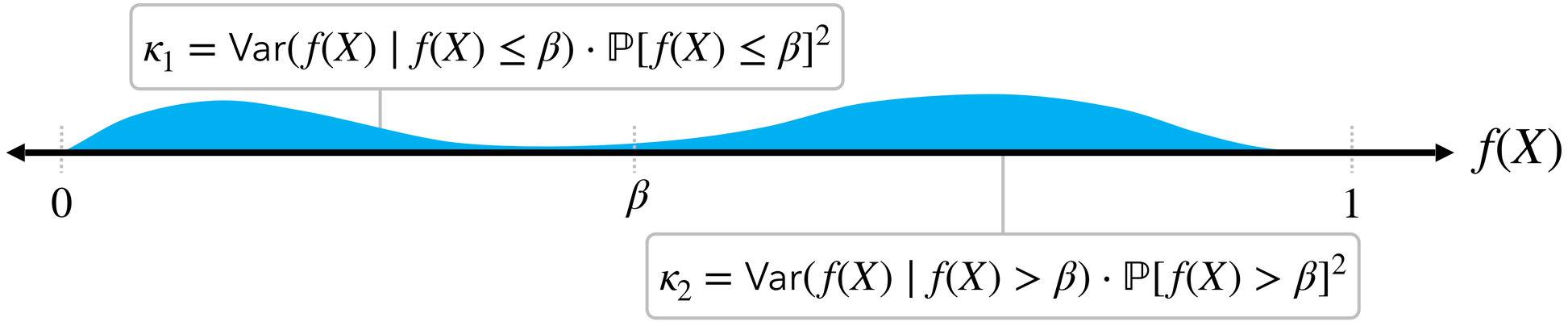
Importance of predictor sharpness

More formally



Importance of predictor sharpness

More formally



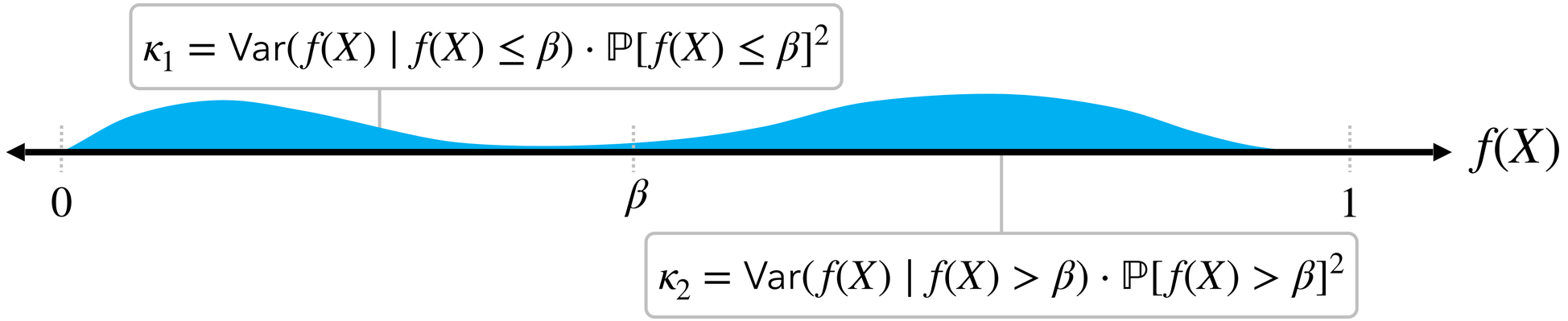
Thm: Let $\mathbb{P}[f(X) > \beta \mid Y = 0] = \epsilon_0, \quad \mathbb{P}[f(X) \leq \beta \mid Y = 1] = \epsilon_1$

$$\mathbb{E}[\text{fraction of interchanged jobs}] \leq \underbrace{\text{Var}(Y)}_{\text{Inherent uncertainty}}\underbrace{(\epsilon_0 + \epsilon_1)}_{\text{False positive/negative}} - \underbrace{(\kappa_1 + \kappa_2)}_{\text{Sharpness}}$$

Shen, **Vitercik**, Wikum, ICML'25

Importance of predictor sharpness

More formally



Thm: Let $\mathbb{P}[f(X) > \beta \mid Y = 0] = \epsilon_0, \quad \mathbb{P}[f(X) \leq \beta \mid Y = 1] = \epsilon_1$

$$\mathbb{E}[\text{fraction of interchanged jobs}] \leq \text{Var}(Y)(\epsilon_0 + \epsilon_1) - (\kappa_1 + \kappa_2)$$

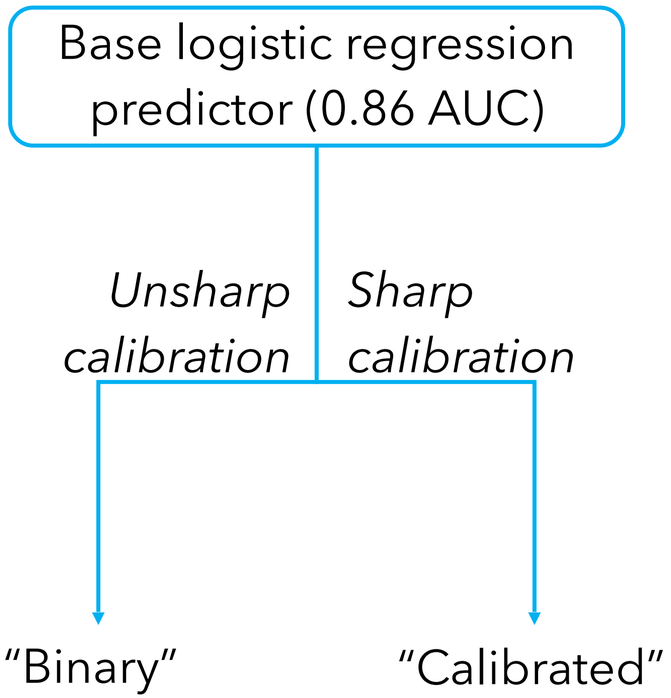
(Eventual) corollary: bound on algorithm's regret (see paper)

Shen, **Vitercik**, Wikum, ICML'25

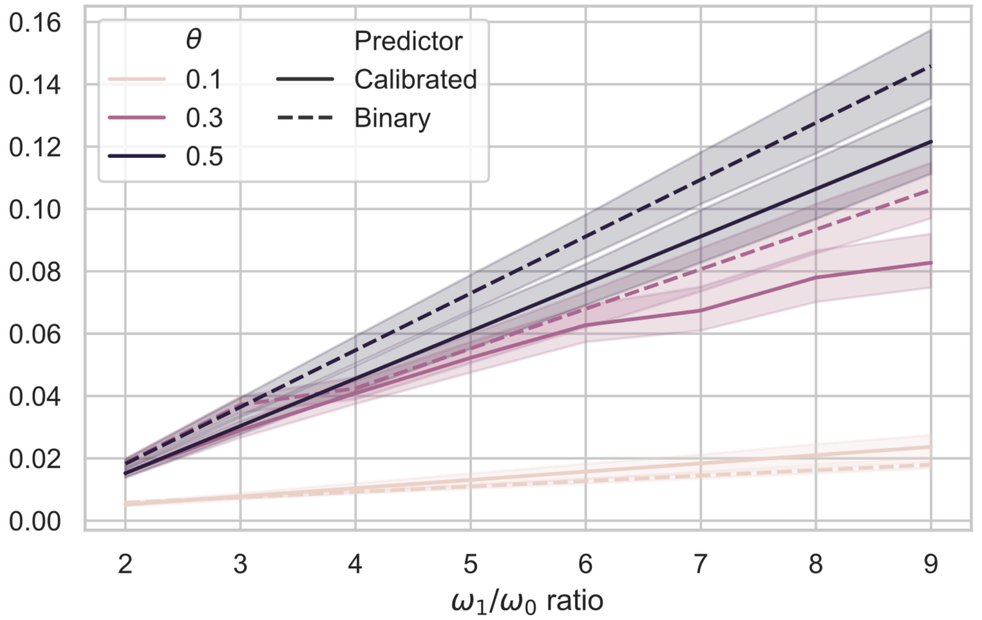
Experiments: Sepsis triage

ML for predicting **sepsis onset** to improve early detection

Dataset of 110,204 hospital admissions



Expected regret per job*



*Scheduling $n = 100$ patient reviews

Shen, **Vitercik**, Wikum, ICML'25

Recap

Uncertainty quantification for robust online optimization

- Prediction errors can amplify in downstream decisions
  - Don’t blindly trust point predictions
- Calibration gives a principled notion of when a prediction can be trusted
- Algorithms w/ guarantees that improve w/ accuracy *and* calibration quality
- In scheduling, **calibrated + sharp** predictions reduce costly **interchanges**

Processing low-priority before high-priority

  - ⇒ Low regret

Shen, **Vitercik**, Wikum, ICML’25

Outline

1. Introduction
2. Mathematical programming [Lawless, Li, Wikum, Udel, Vitercik, CPAIOR'25]
3. Uncertainty quantification in scheduling [Shen, Wikum, Vitercik, ICML'25]
4. **Conclusions**

Conclusions

- Real-world optimization is **dynamic but structured**:
  - Historical instances are a reusable source of signal
- **Case study 1:** LLMs to *cold-start* MILP configuration
- **Case study 2:** More *robust* policies than "trust-the-predictor," via calibration
- **Future directions:**
  - Evaluation pipelines that are more representative of real-world problems
  - Zeroing in on solver "knobs" matter most in practice
    - E.g., cuts, branching/heuristics, warm starts, re-solving cadence

Machine Learning for Optimization

Ellen Vitercik

Stanford University