

AI for trustworthy optimization

AI for trustworthy optimization

How can we use ML/AI to improve optimization and decision-making while retaining the guarantees that make traditional algorithms trustworthy?

- Optimization underpins many of society's most pressing challenges
 - Decarbonizing energy systems, allocating scarce medical resources, ...
- These problems are often NP-hard, and thus difficult to solve at scale
- **Overall research goals are to:**
 1. Leverage data/ML to improve optimization, decision-making, algorithms
 2. Understand the algorithmic capabilities of LLMs

Learning from structured problems



Why machine learning?

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

Example:

- 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**
- Many new ways to leverage LLMs for **automated algorithm design**

Example: **mathematical optimization**

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



Key challenges: AI for trustworthy optimization

Designing hybrid method that satisfy the essential obligations of trust

- **Fidelity:** solve the problem decision-makers actually intended
 - Problem formulation remains challenging for non-expert users
- **Feasibility:** every recommended action can be executed
 - No hallucinations or relaxations
- **Optimality:** quantify remaining room for improvement
 - Guarantee a *substantially* better solution doesn't exist
 - Especially important in high-stakes domains

Outline

1. Introduction
- 2. Formal certification for AI-driven optimization** [NeurIPS'26]
3. Uncertainty quantification for downstream decision-making [ICML'25]
4. Conclusions



Henry Robbins



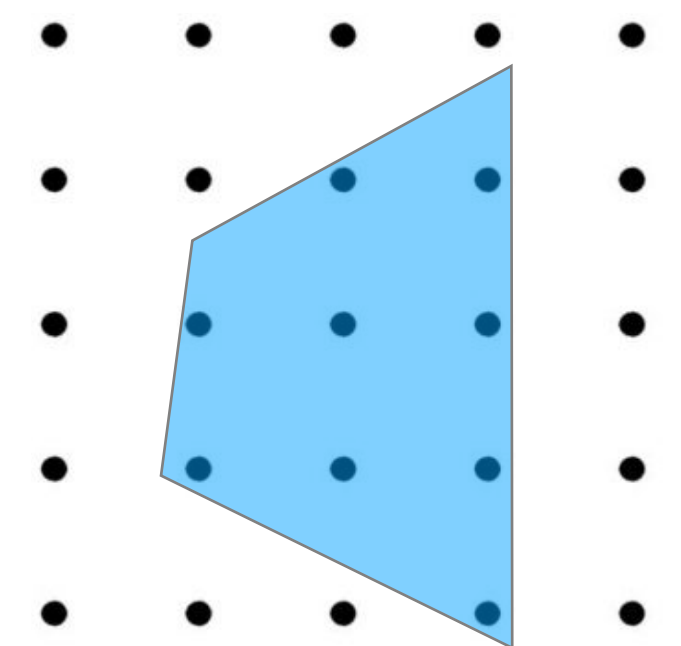
Conor Lawless



Madeleine Udell

Key facets of an optimization problem

- **Objective:** criterion we wish to minimize (e.g., cost) or maximize (e.g., profit)
 - Minimize cost, emissions, delay, unmet demand, ...
- **Decision variables:** represent available actions
 - Purchase assets, assign resources, schedule operations, ...
- **Constraints:** define limitations restricting our choices for decision variables
 - Capacity, safety, timing, ...
- Optimization selects the best feasible actions
- $\min f(x)$ subject to $x \in F$



Example: power grid operation

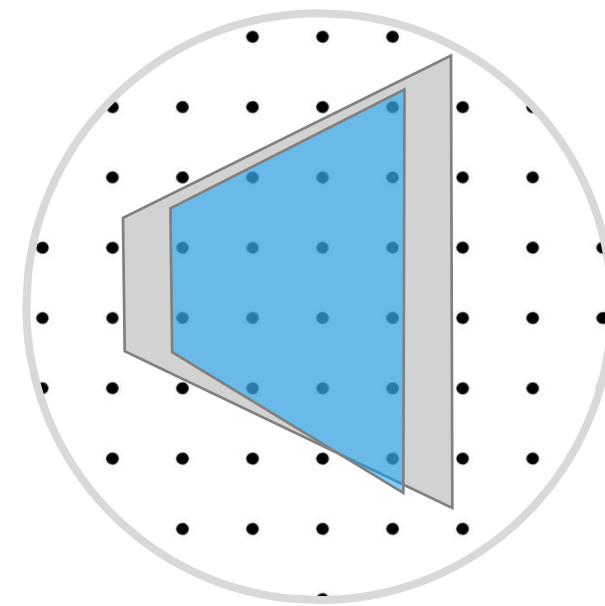
- **Objective:** Minimize operating costs
- **Decision variables:** Decide which generators to turn on and off when
- **Constraints:**
 - Meet electricity demand every hour
 - Maintain reliability and reserve requirements
 - Account for renewable generation uncertainty
- Tomorrow resembles previous days, but conditions change
 - Historical patterns can guide **faster optimization**



Can AI improve optimization models...

...without changing the problem?

- Same optimization problem can have many mathematical formulations
- Different formulations can have dramatically different solve times



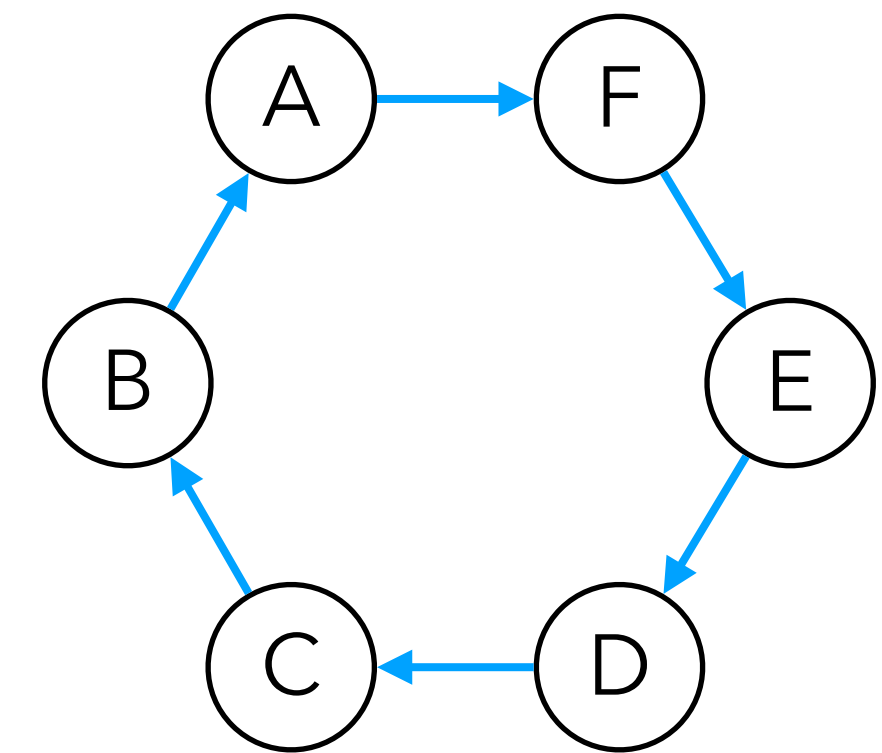
- Can we let AI propose improvements and certify they preserve the problem?



Example: traveling salesman problem

Find shortest tour through every city

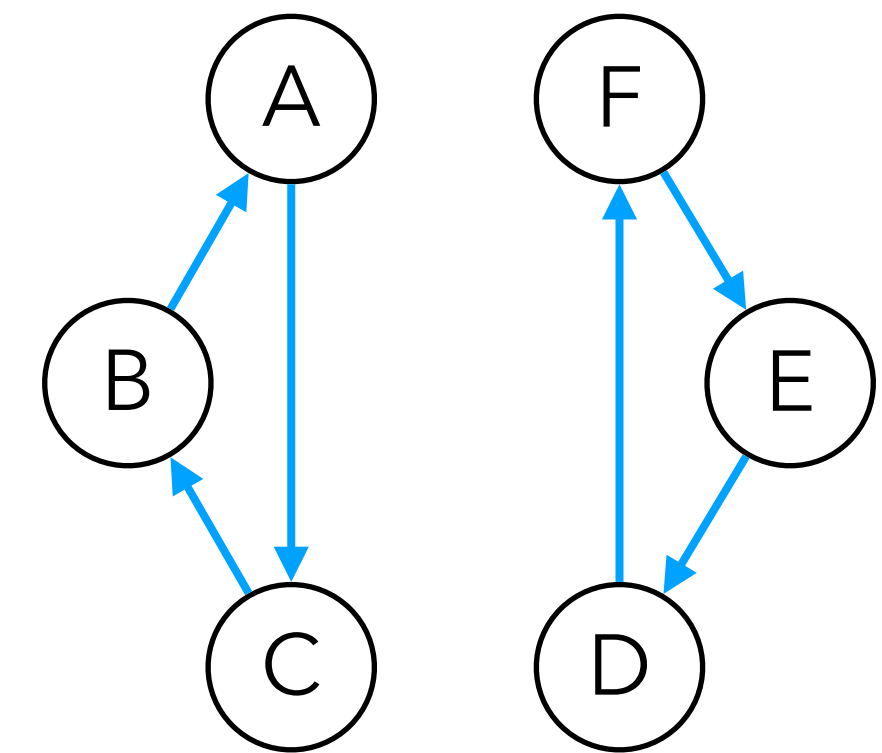
- **DFJ formulation** [Dantzig et al., '54]:
 - Represent tours using selected edges



Example: traveling salesman problem

Find shortest tour through every city

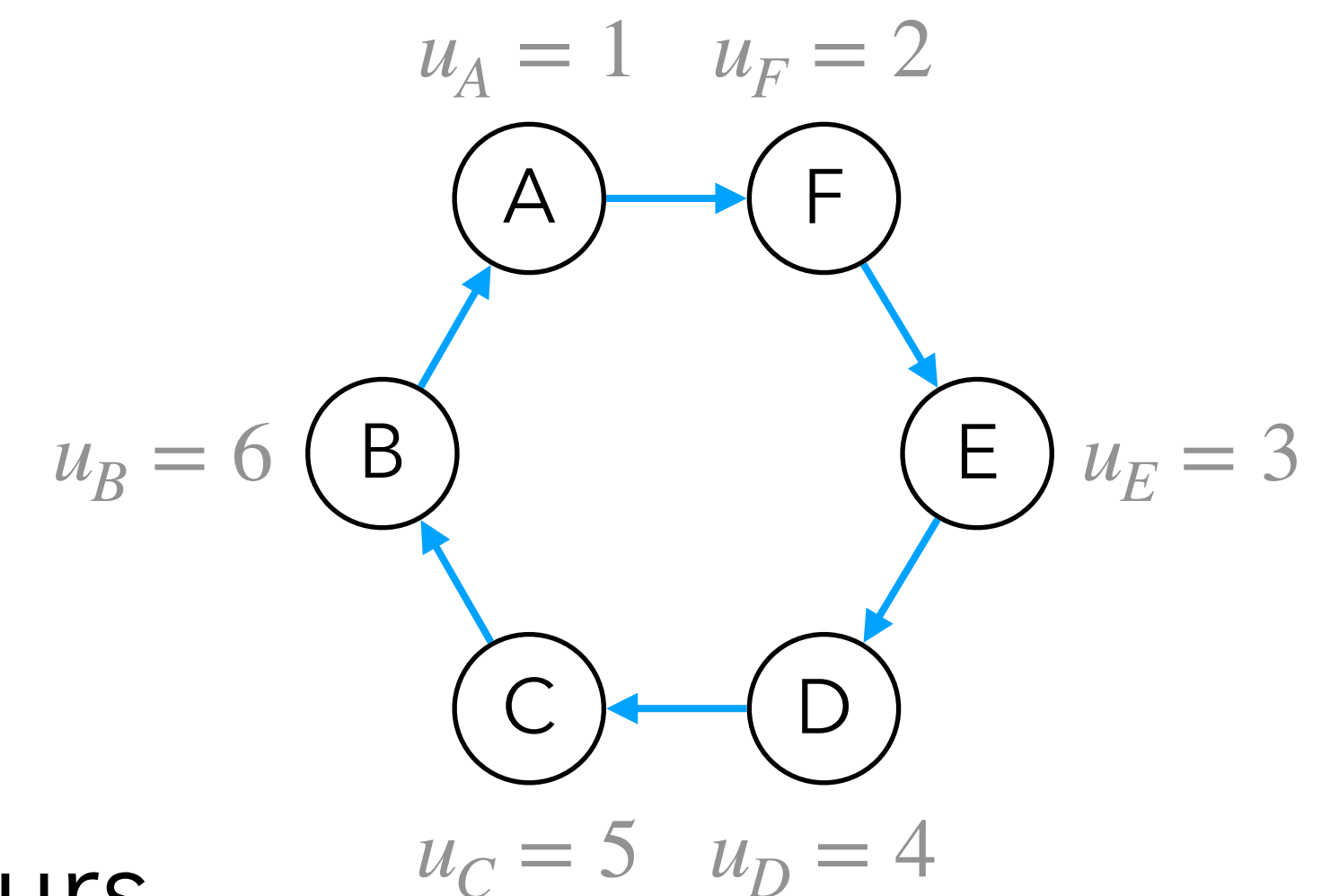
- **DFJ formulation** [Dantzig et al., '54]:
 - Represent tours using selected edges
 - Exponentially many constraints rule out subtours



Example: traveling salesman problem

Find shortest tour through every city

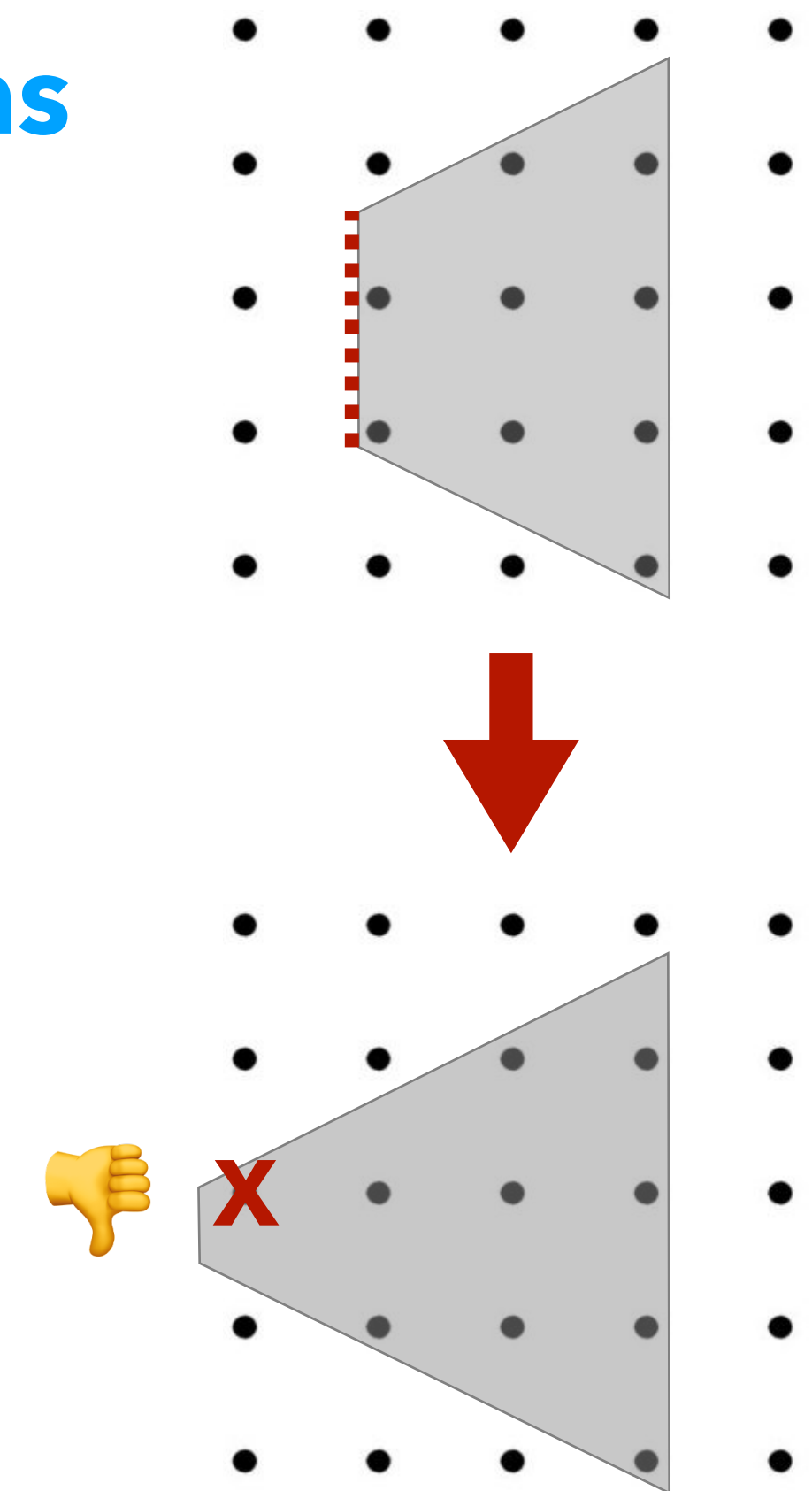
- **DFJ formulation** [Dantzig et al., '54]:
 - Represent tours using selected edges
 - Exponentially many constraints rule out subtours
- **MTZ formulation** [Miller et al., '60]:
 - Add variables representing cities' tour positions
 - Ordering constraints prevent disconnected subtours
 - Polynomially many constraints suffice
- Different variables and constraints, but same optimization problem



Failure mode: fast solution to wrong problem

Learned reformulations can accidentally change the original problem

- Missing constraints (for example) may admit **impossible plans**
 - Example: reserve requirements disappear from grid scheduling
- Solvers certify only the **reformulated model**
 - Their guarantees may not apply operationally
- Faster runtimes can't compensate for lost fidelity



Testing is not verification

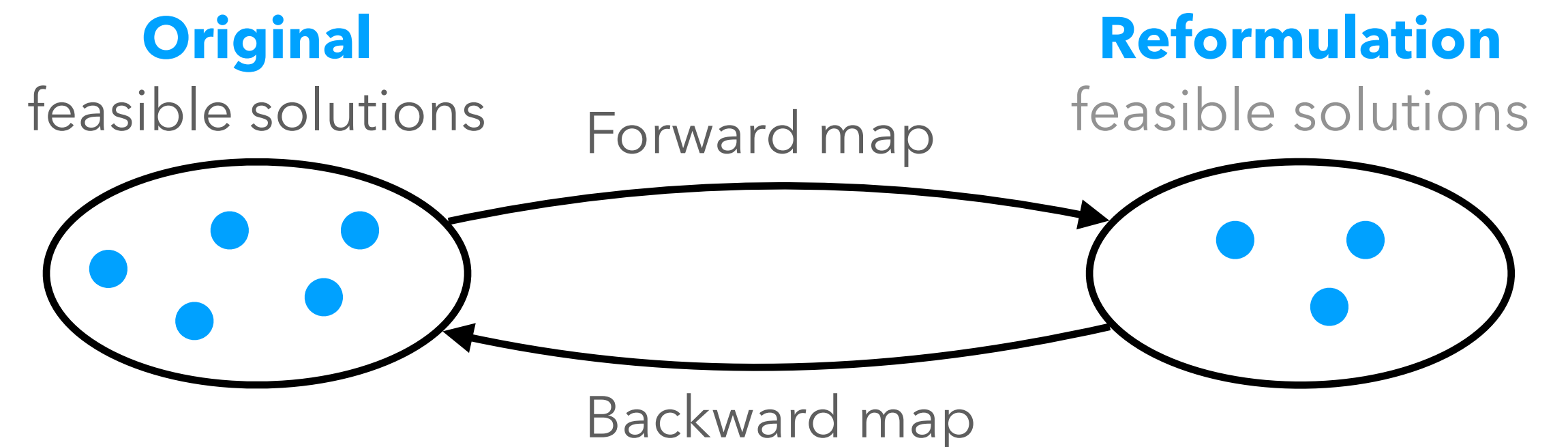
AI-generated reformulations must be formally verified

- AI-generated formulations must **faithfully represent the original problems**
 - Existing approaches rely heavily on heuristics
 - E.g., comparing **optimal objective values** on small set of instances
 - AhmadiTeshnizi, ICML'24; Huang et al., OR'25; ...
- Testing checks **only the instances tested**
 - Doesn't establish correctness for every instance
- We need **formulation-level verification**, not just instance-level testing
 - Guarantee correctness for every instance (e.g., every TSP instance)

What's it mean for a **reformulation** to be valid?

Sol'n to reformulation should directly imply a sol'n to original problem

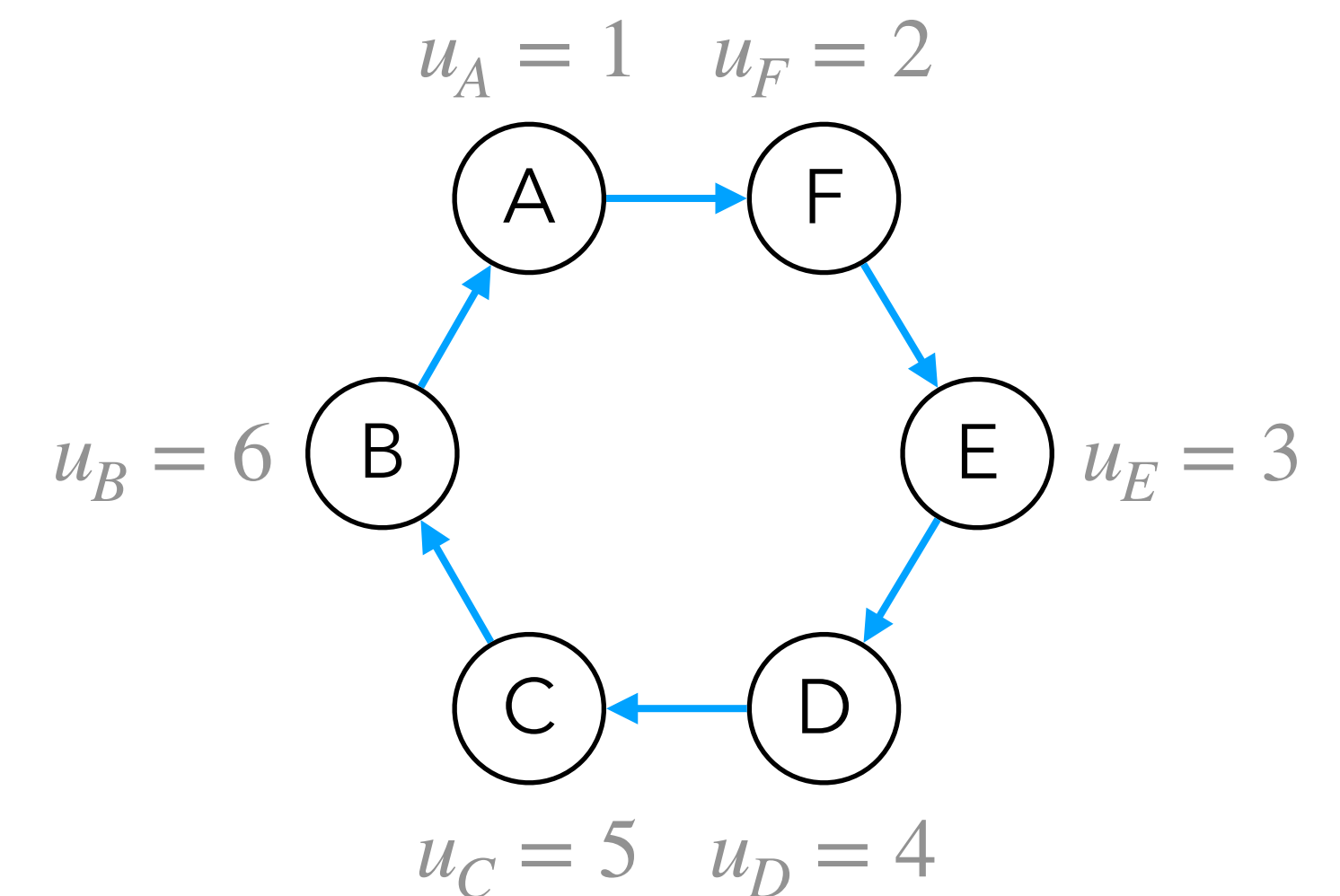
- To prove a reformulation is valid (high-level idea; see paper for details):
 1. Maps that **translate feasible solutions** between the formulations
 - Every feasible original sol'n maps to a feasible reformulated sol'n, and vice versa
 2. **Relate objective values** through a strictly increasing transformation
 - Better solutions stay better, never becoming tied
- Conditions ensure
optimal sol'ns $\mapsto$ optimal sol'ns



What's it mean for a **reformulation** to be valid?

Example: traveling salesman problem

- **DFJ $\rightarrow$ MTZ:** Add variables denoting tour positions
 - Start with any feasible DFJ tour
 - Number cities in order they're visited
 - These positions satisfy MTZ's ordering constraints
- **MTZ $\rightarrow$ DFJ:** Drop the position variables
 - Keep the selected edges
 - MTZ's ordering constraints already rule out subtours
- Both mappings **preserve tour length**
 - Selected edges and total cost remain unchanged
 - So optimal solutions $\mapsto$ optimal solutions



FLARE

FLARE builds a checkable certificate for a reformulation claim

1. **Formalize:** Translate both formulations into Lean
2. **Prove:** LLM (Claude Opus 5*) constructs solution mappings and proofs
 - Show both mappings preserve feasibility
 - Show objective values satisfy the required relationships
3. **Check:** Lean verifies the proposed mathematical proof
 - The LLM revises proofs using Lean's feedback

Accept only after Lean checks a complete proof

*See paper for model ablations

What does the certificate guarantee?

A guarantee about the **formalized reformulation statement**

- Valid for every instance in the problem family
- Not just instances evaluated during testing

A condition we still **need to check**

- Lean definitions faithfully represent the intended formulations

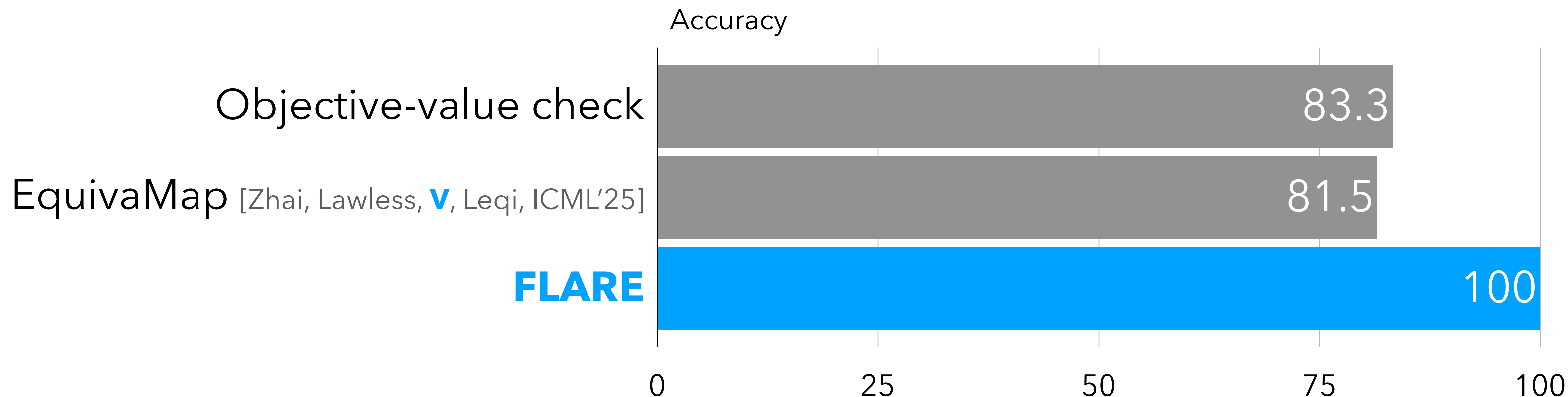
A failed proof attempt is **inconclusive**

- No certificate doesn't mean an invalid reformulation

FormulationBench

20 problems; 109 formulations

Main evaluation: 54 formulation pairs across 16 NP-hard problems



42/42 valid pairs certified; **none** of the 12 invalid pairs certified

FLARE-NL (Natural Language)

Fast and cheap proxy that doesn't produce certificates

FLARE-NL uses the same definition and assumptions, but no proof checker

	FLARE	FLARE-NL
Benchmark accuracy	100%	100%
Checkable certificate	Yes	No
Average time	6.8 minutes	13.8 seconds
Average cost	\$1.18	\$0.05

Possible workflow: use FLARE-NL for rapid screening, then FLARE to certify

Take-aways

LLMs construct proofs that Lean checks to certify reformulation validity

- **Testing examples** can't establish correctness for all inputs
- We **prove correctness** for every instance in problem family
 - Verify solution mappings without solving individual optimization instances
- **LLMs** construct proofs; **Lean** checks them
- FLARE certified **all 42** valid pairs in the evaluation
 - None of the 12 invalid pairs received certificates
- **Goal:** AI-driven optimization improvements with rigorous guarantees

Outline

1. Introduction
2. Formal certification for AI-driven optimization [NeurIPS'26]
- 3. Uncertainty quantification for downstream decision-making** [ICML'25]
4. Conclusions



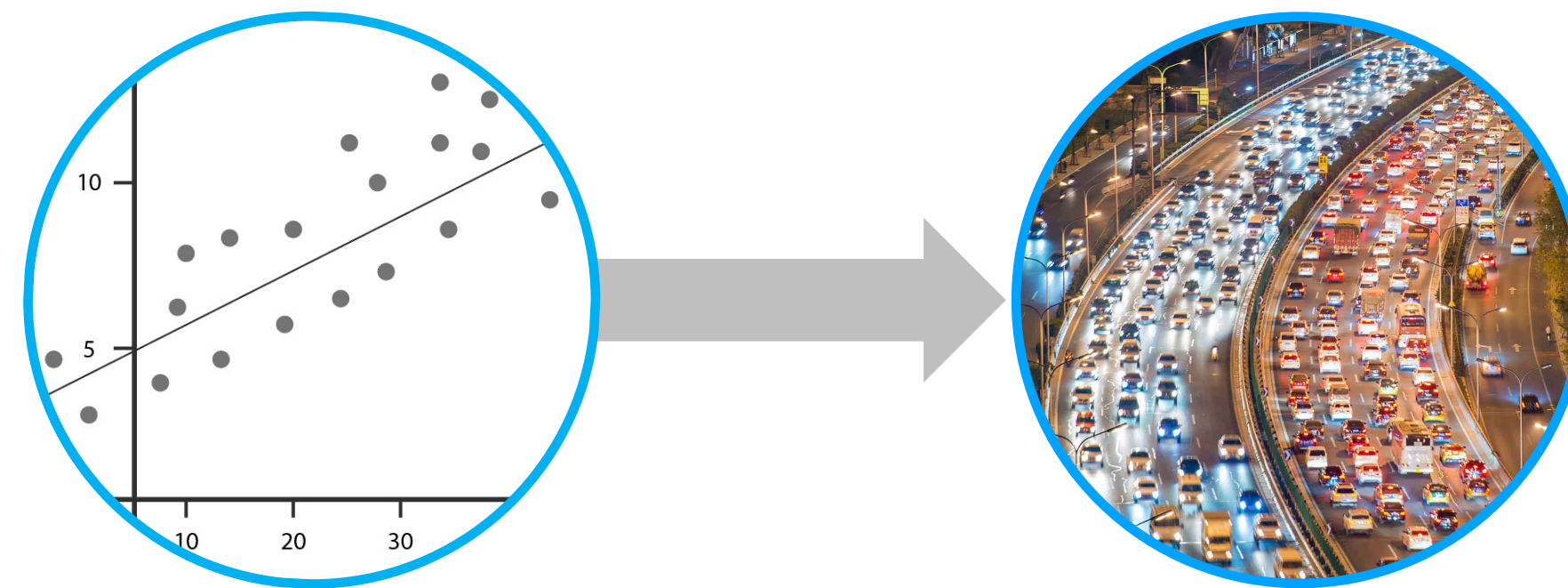
Judy Shen



Anders Wikum

Algorithms and prediction uncertainty

- Challenge: prediction **errors can amplify** in decision-making
 - Don't blindly trust predictions



- **Jumping off point:** ML models can estimate uncertainty **automatically**
 - Examples: calibration and conformal predictions
 - Well-defined, statistical notion of whether a prediction can be trusted

Our contributions

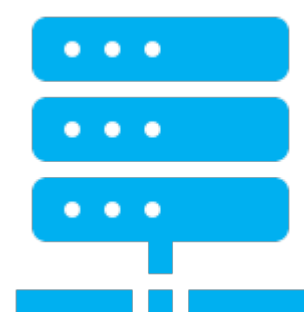
Demonstrate calibration's utility for online algorithms

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 $\mathcal{X} \times \{0,1\}$
- $f: \mathcal{X} \rightarrow [0,1]$ 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)



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

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

ℓ_2 error

Uncertainty

Sharpness

Calibration error

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**

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

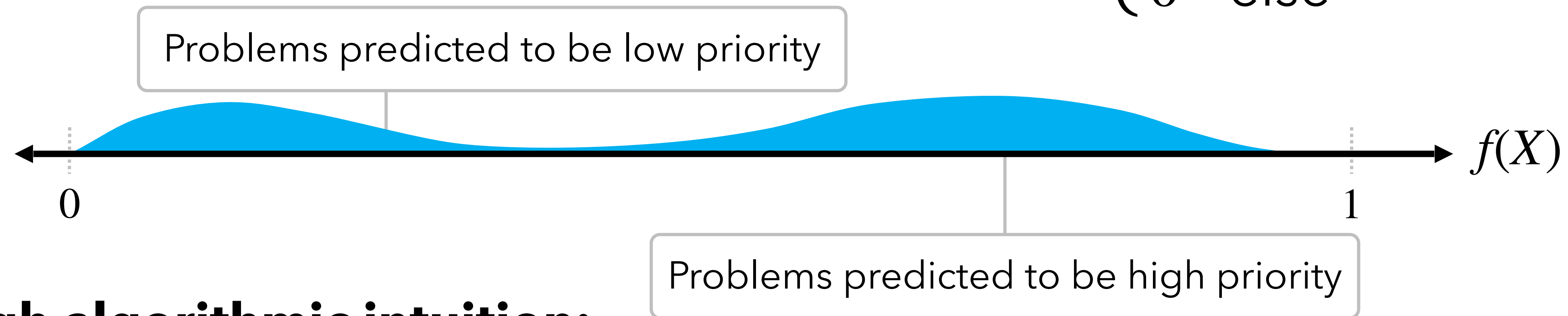
$$\sum C_i \cdot \omega_{y_i}$$

Completion time of job i

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

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

If discover current is low-priority, stop, move to end of queue, and start new job

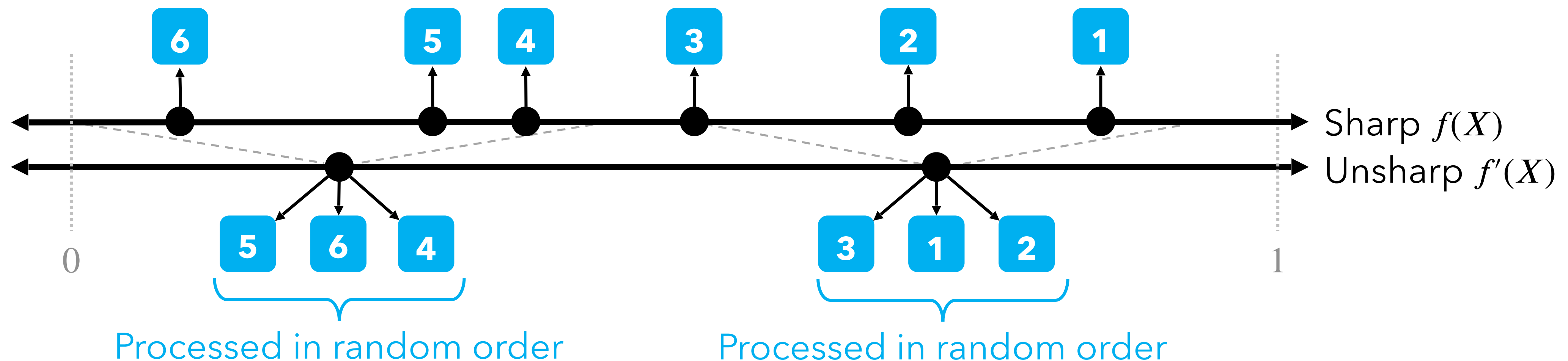
Importance of predictor sharpness

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

Low priority job processed before high priority job

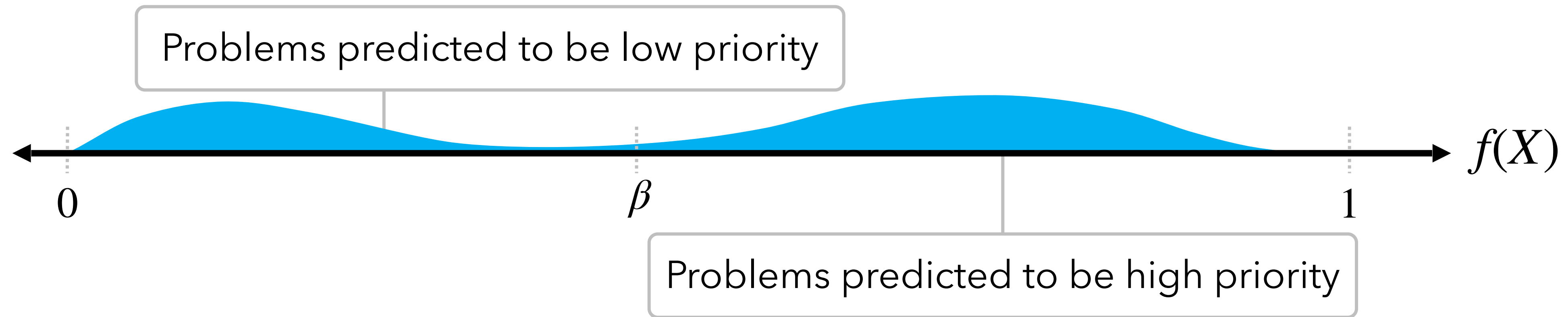
Weighted sum of completion times vs opt in hindsight

- Sharp predictors** lead to **fewer interchanges**



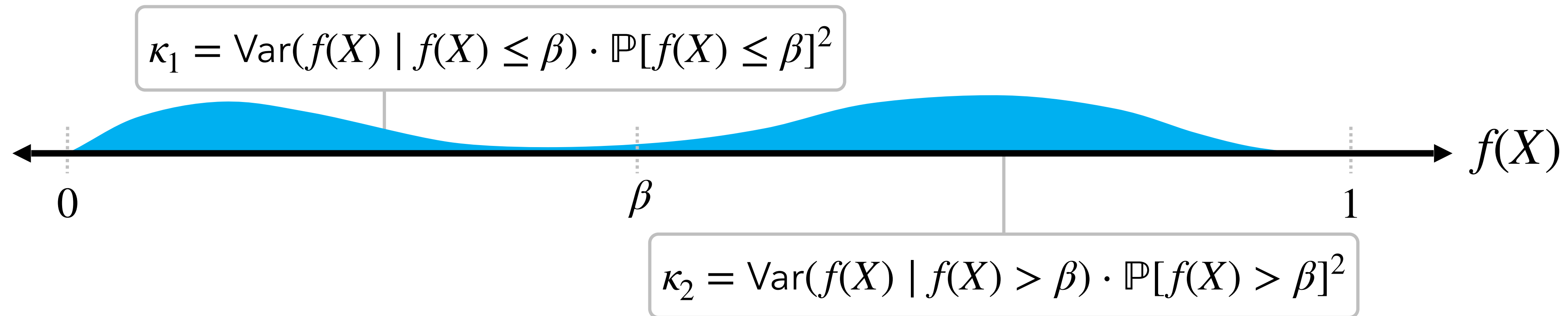
Importance of predictor sharpness

More formally



Importance of predictor sharpness

More formally



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

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

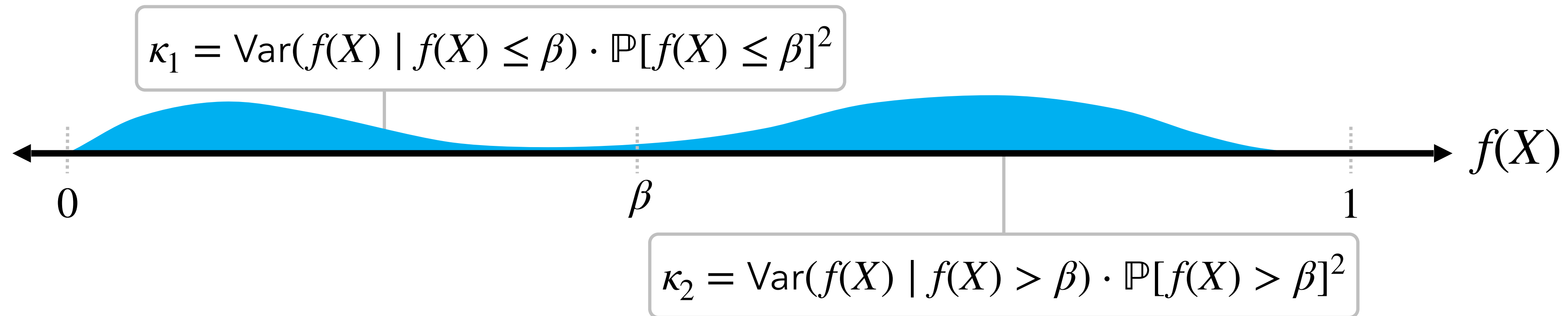
Inherent
uncertainty

False
positive/negative

Sharpness

Importance of predictor sharpness

More formally



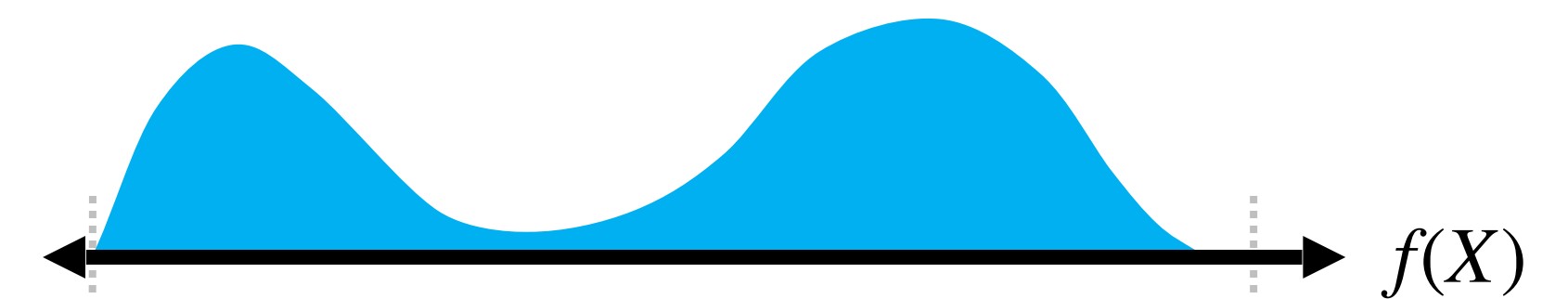
Thm: Let $\mathbb{P}[f(X) > \beta \mid Y = 0] = \epsilon_0$, $\mathbb{P}[f(X) \leq \beta \mid Y = 1] = \epsilon_1$

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

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

Take-aways

- Prediction errors can amplify in decision-making
 - Trust requires **uncertainty bounds**, not just accuracy
- **Calibration** gives statistical meaning to confidence scores
- We provide **online algorithms** that use calibrated uncertainty
- Regret tracks uncertainty, errors, and **sharpness**
- Case studies: rent-or-buy and online scheduling



Outline

1. Introduction
2. Formal certification for AI-driven optimization [NeurIPS'26]
3. Uncertainty quantification for downstream decision-making [ICML'25]
- 4. Conclusions**

AI for trustworthy optimization

How can we use ML/AI to improve optimization and decision-making while retaining the guarantees that make traditional algorithms trustworthy?

Additional examples:

- LLM reasoning [He et al., ICML'26, NeurIPS'26]
- Online decision-making [Sun et al., ICML'25; Li et al., ITCS'26]
- ML for algorithm selection [Chatziafratis et al., NeurIPS'25]
- Neural algorithmic reasoning [He et al., ICML'25]
- ML for solver configuration [Lawless et al., CPAIOR'25]
- Fact-checking [Selvam et al., ICML'26]

Thanks! Questions?



Yu He



Yingxi Li



Henry Robbins



Nikil Selvam



Anders Wikum



Xizhi Tan

PhD students

Postdoc