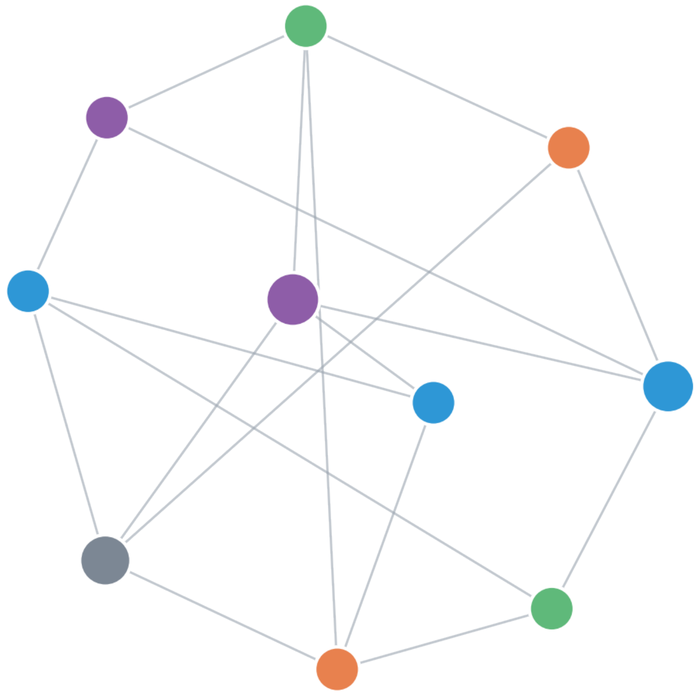
Machine Learning for Combinatorial Optimization

Ellen Vitercik · Stanford University



ML for combinatorial optimization

Combinatorial opt has long captured ML researchers' imaginations

- E.g., traveling salesman problem (TSP): iconic for theory and ML
  - 40 years from Hopfield networks [e.g., Hopfield, Tank, '86] to LLMs

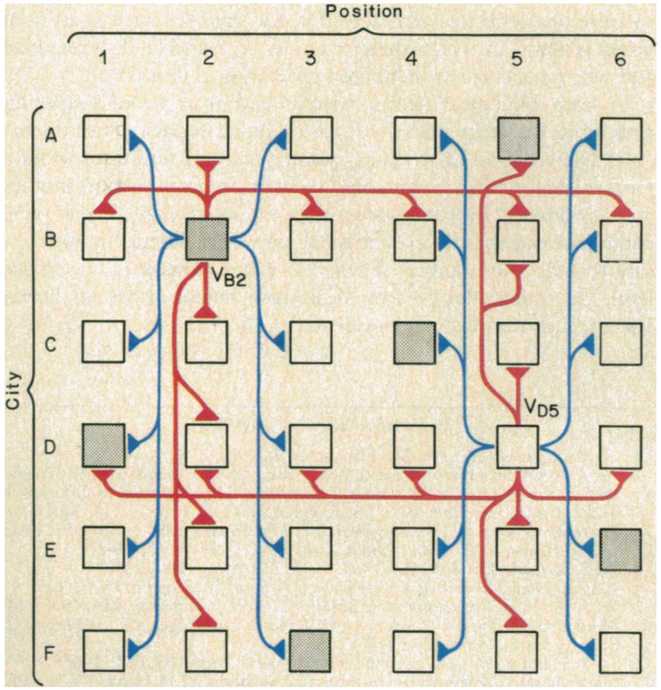


Fig. 5. A stylized picture of the syntax and connections of the TSP neural circuit. Each neuron is symbolically indicated by a square. The neurons are arranged in an n by n array. Each city is associated with n neurons in a row, and each position in the final tour is associated with n neurons in a column. A given neuron ($V_{X,j}$) represents the hypothesis that city X is in position j in the solution. The patterns of synaptic connection for two different neurons are indicated.

8 AUGUST 1986

ML for combinatorial optimization

Combinatorial opt has long captured ML researchers' imaginations

- E.g., traveling salesman problem (TSP): iconic for theory and ML
  - 40 years from Hopfield networks [e.g., Hopfield, Tank, '86] to LLMs
- Hard problems recur on **structured instance distributions**
  - Data reveals patterns not leveraged by worst-case analysis
- ML can **guide decisions** at multiple interfaces
  - Policies, solver controls, heuristics, configurations
- Inputs are often **uncertain** or **partially observed**
  - Predictions help, but errors can amplify

ML for combinatorial optimization

Many different forms

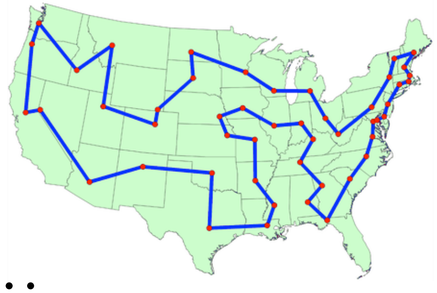
- **Goal 1:** Improve algorithms on structured instance distributions
  - Neural combinatorial optimization
  - ML for exact combinatorial optimization
  - Data-driven algorithm design
  - Automated heuristic discovery
- **Goal 2:** Learn uncertainty before optimizing
  - Data-driven robust optimization
  - Algorithms with predictions
  - Decision-focused learning
  - Neural algorithmic reasoning
- Discover new algorithms with provable guarantees?

Outline

1. **Survey of different approaches**
    i. **Goal 1: Improve algorithms on structured instance distributions**
    ii. Goal 2: Learn uncertainty before optimizing
2. Algorithms with calibrated machine learning predictions [ICML'25]
3. Probe LLM reasoning through the lens of optimization [ICML'26]
4. Conclusions and future directions

Neural combinatorial optimization

Hopfield, Tank, ’85; Durbin, Willshaw, ’87; Vinyals et al., ’15; Bello et al., ’17; Dai et al., ’17; Nazari et al., ’18; …



- Learn **end-to-end policies** for discrete optimization
  - Output solutions directly, without hand-designed heuristics
- Canonical problems: TSP, VRP, vertex cover, independent set, …
- **Training:** imitation learning or reinforcement learning
  - **Decoding:** greedy, sampling, beam search
- **Core challenges:** feasibility, generalization, improvement over solvers
- **Theory angle:** what explains an architecture’s strong performance?

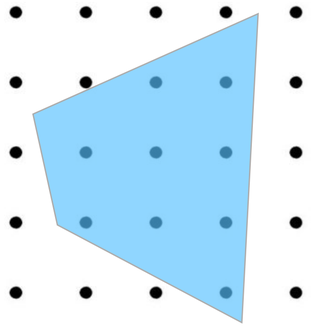
  Hayderi, Saberi, **V**, Wikum, ICML’25

ML for exact combinatorial solvers

Gomes, Selman, ’01; Nudelman et al., ’04; Kadioglu et al., ’11; Sandholm, ’13; He et al., ’14; Khalil et al., ’16; …

- **Keep solver**; learn the expensive choices
  - Branching, cutting, restarts, presolve, primal heuristics
- **Domains:** integer programming, SAT, CP
- **Objective:** reduce search tree nodes, wall-clock time
- ML guides; solvers certify solutions and proofs
- What’s the right place to integrate LLMs?

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

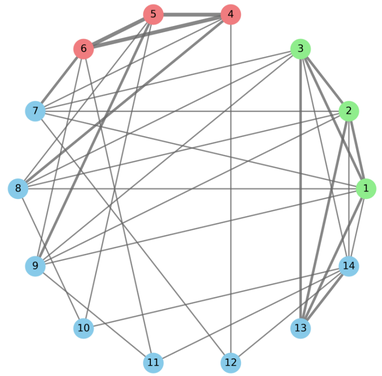


LLMs for automated heuristic discovery

Liu et al., ’23; Romera-Paredes et al., ’24; Liu et al., ’24; Ye et al., ’24; Zhang et al., ’24; Novikov et al., ’25; …

- Use LLMs to automate the human loop of **heuristic design**
- Search over programs, rules, priorities
- Heuristic generation/refinement via LLM-guided **evolutionary search**
  - Fitness: solution quality, runtime
- Examples: Heuristics for
  - Canonical problems like TSP, bin packing, …
  - Discovering lower bound constructions

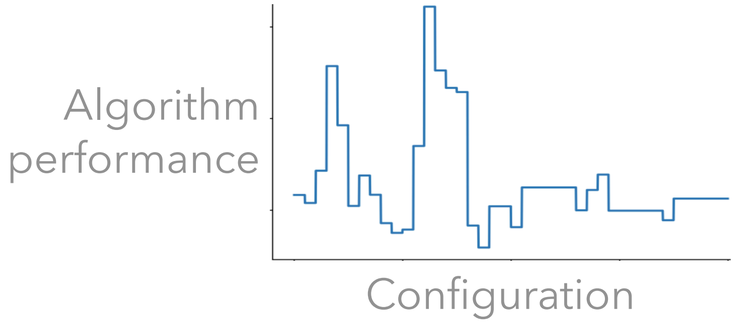
    E.g., Nagda et al., ’25



Data-driven algorithm design

Gupta, Roughgarden '16; Balcan, Nagarajan, **V**, White, '17; Kleinberg et al., '17; Balcan, Dick, **V**, '18; Garg, Kalai, '18; …

- What **theory guarantees** can we give for learning algorithm configurations?
  - Examples: clustering, tree search, dynamic programming, …
- Training data represents **historical** problem instances
  - Goal: strong expected performance – proxy for **future** performance
- Analyze generalization beyond sampled training set: **sample complexity**
- **Key challenge:** algorithm performance is volatile
  - Tweak parameters ⇒ large changes in behavior



Goal 1: Improve algs on structured problems

Summary

Method	Where ML enters	What is learned/produced	Major focus/challenge
Neural CO	Solution construction	Neural policy	Feasibility, generalization
ML for exact solvers	Solver control	Branching/cutting/… rules	Distribution shift
LLM heuristic discovery	Heuristic design	Executable heuristic code	Benchmark overfitting
Data-driven algorithm design	Algorithm configuration	Tuned algorithm family	Sample complexity

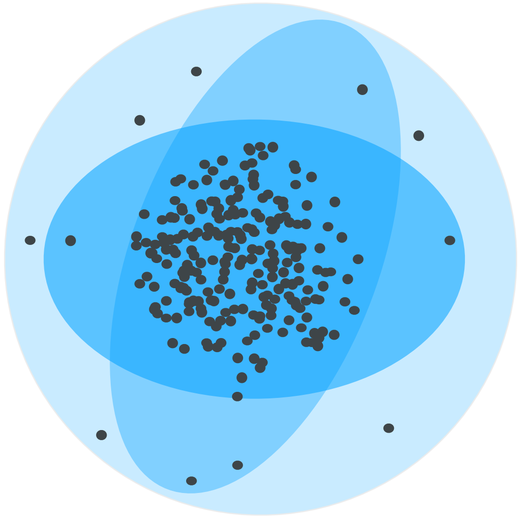
Outline

1. Survey of different approaches
    i. Goal 1: Improve algorithms on structured instance distributions
    ii. Goal 2: Learn uncertainty before optimizing
2. Algorithms with calibrated machine learning predictions [ICML'25]
3. Probe LLM reasoning through the lens of optimization [ICML'26]
4. Conclusions and future directions

Data-driven robust optimization

Scarf, ’58; Soyster, ’73; El Ghaoui, Lebret, ’97; Ben-Tal, Nemirovski, ’98; Bertsimas, Sim, ’04; Calafiore, Campi, ’06; …

- Learn an **uncertainty model** from historical data
  - Demands, costs, travel times
- Uncertainty enters the **optimization problem** via confidence/ambiguity sets, calibrated intervals
- Solver chooses decisions safe across plausible futures
- **Goal:** protect against sampling error and distribution shift
- **Tradeoff:** robustness versus conservatism



Algorithms with predictions

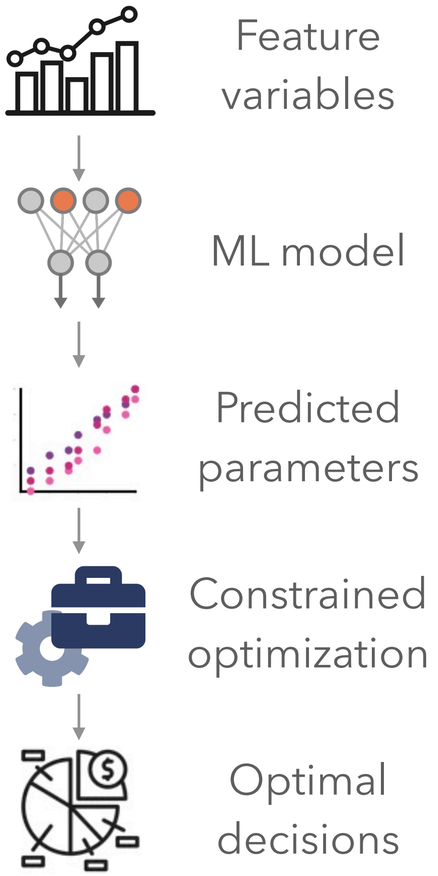
Mahdian, et al., ’07; Devanur, Hayes, ’09; Emek et al., ’11; Boyar et al., ’17; Lykouris, Vassilvitskii, ’18; …

- Use point predictions (typically) while preserving **worst-case guarantees**
- Add predictions to **classical algorithms** (future arrivals, sizes, ranks, optima)
- Predictions guide decisions (e.g., thresholds, evictions, matchings)
- **Goal:** exploit ML advice without over-trusting it
- **Robustness:** Retain worst-case guarantees when predictions fail
- **Consistency:** Guarantees scale with prediction error

Decision-focused learning

Bengio, ’97; Amos, Kolter, ’17; Donti et al., ’17; Elmachtoub, Grigas, ’17; Wilder et al., ’19; Agrawal et al., ’19; …

- Train predictors for **downstream optimization quality**
- Loss measures **final decision cost**, not prediction error
- **Optimization layer** maps predictions to decisions
  - Differentiable solvers or gradient surrogates
- Useful when prediction accuracy misaligns with value
- **Tradeoff:** decision quality versus training complexity



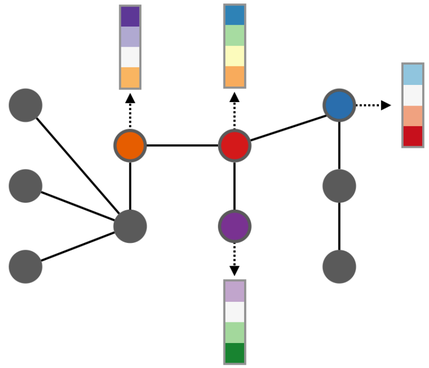
Mandi et al., ’24

Neural algorithmic reasoning

Zaremba, Sutskever, ’14; Graves et al., ’14; Reed, de Freitas, ’16; Veličković et al., ’20; Veličković, Blundell, ’21; …

- Learn **algorithmic procedures internally** (no external solver call)
- Neuralize algorithmic **process** (e.g., dynamic programming, search)
- Encode algorithmic priors as **inductive biases**
- Desired wins: extrapolation (e.g., size generalization), robustness
- **Example:** GNNs can exactly simulate primal-dual algorithm

  He, **V**, ICML’25



Goal 2: Learn uncertainty before optimizing

Summary

Method	What ML provides	Where ML enters	Output
Data-driven robust optimization	Uncertainty model	Optimization formulation	Robust decision
Algorithms with predictions	Point advice	Algorithm decision rule	Guided classical algorithm
Decision-focused learning	Decision-aware predictor	Training objective	Decision-aware predictor
Neural algorithmic reasoning	Learned computation	Neural architecture/process	Internal algorithm executor

Outline

1. Survey of different approaches
    i. Goal 1: Improve algorithms on structured instance distributions
    ii. Goal 2: Learn uncertainty before optimizing
2. **Algorithms with calibrated machine learning predictions** [ICML'25]
3. Probe LLM reasoning through the lens of optimization [ICML'26]
4. Conclusions and future directions

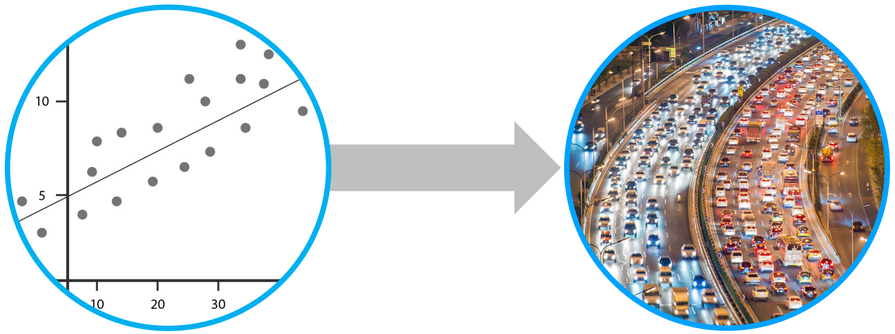
Judy Hanwen Shen

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

Shen, **Vitercik**, Wikum, ICML’25

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

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

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

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

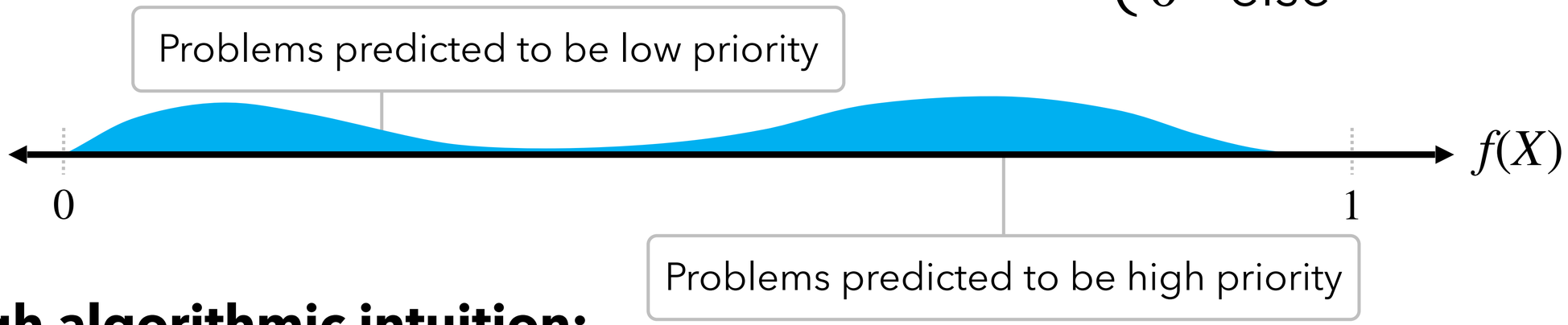
Cost per unit delay, with $\omega_1 > \omega_0 > 0$

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:

$f(X)$

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

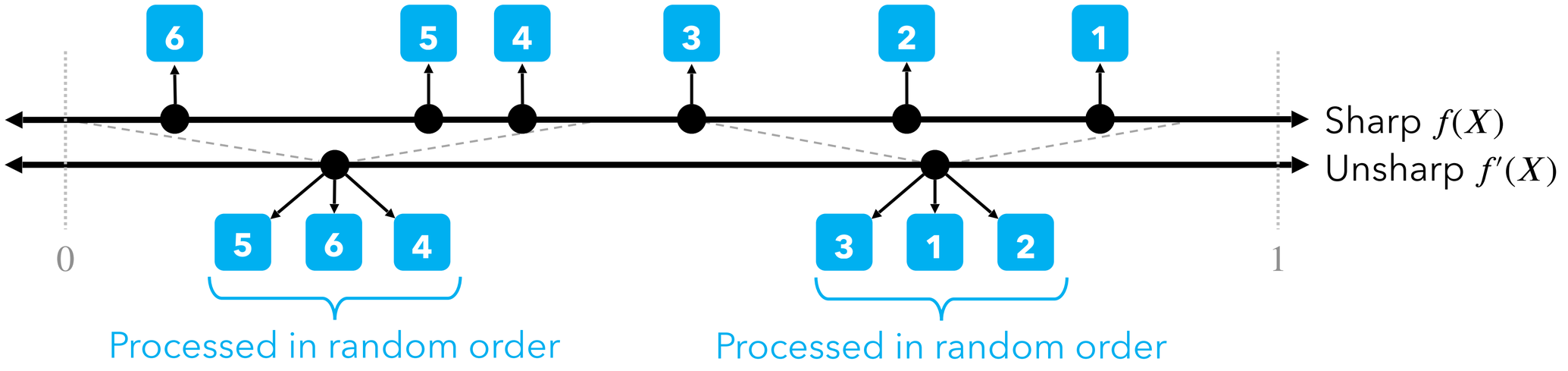
Importance of predictor sharpness

- 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

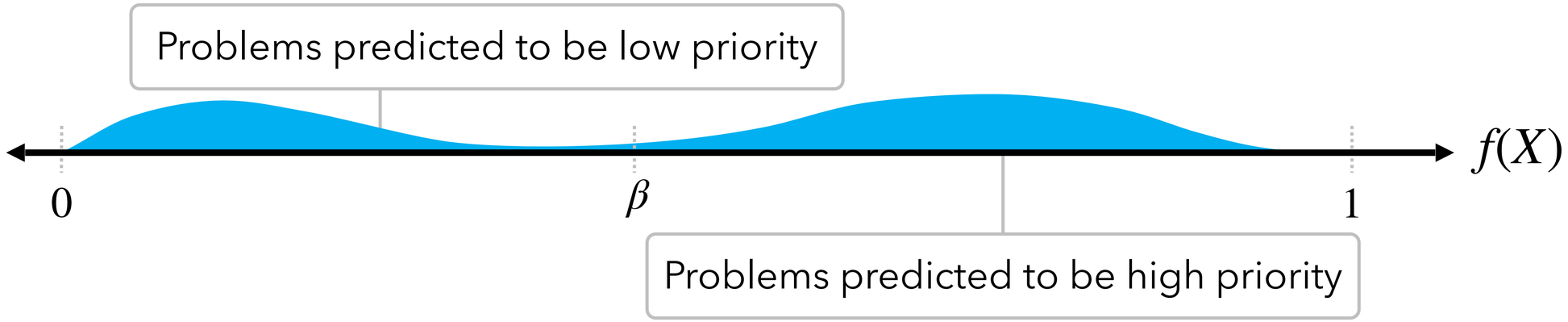
- **Sharp predictors** lead to **fewer interchanges**



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

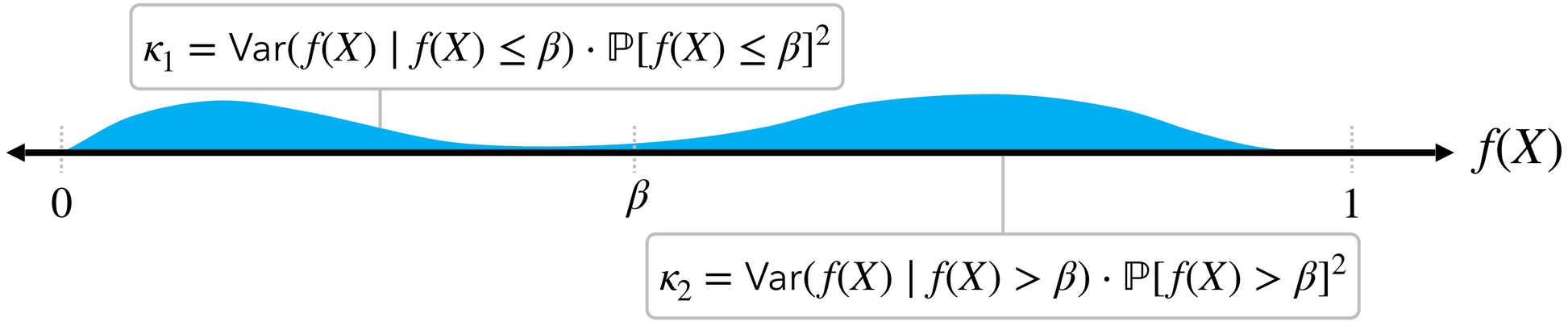
Importance of predictor sharpness

More formally



Importance of predictor sharpness

More formally



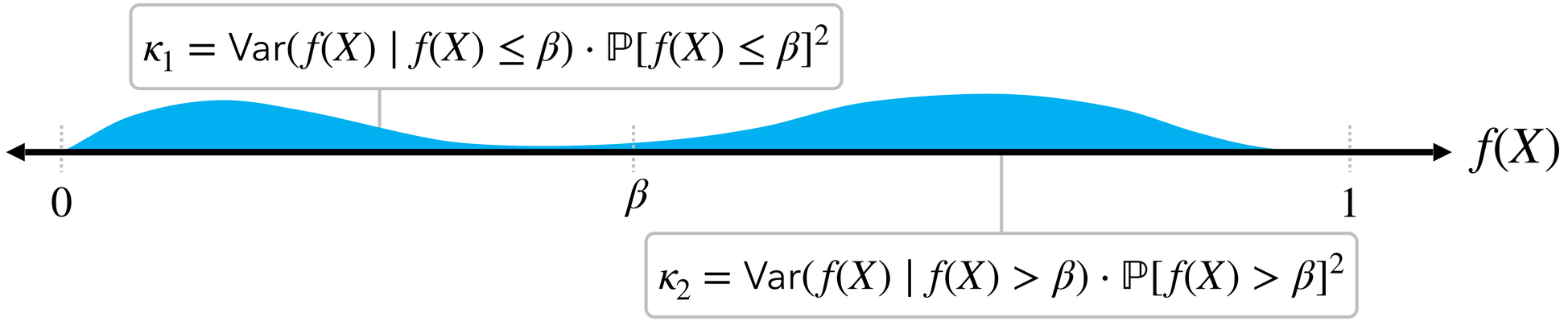
Thm: Let $\mathbb{P}[f(X) > \beta \mid Y = 0] = \epsilon_0, \quad \mathbb{P}[f(X) \leq \beta \mid Y = 1] = \epsilon_1$

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

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

Importance of predictor sharpness

More formally



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

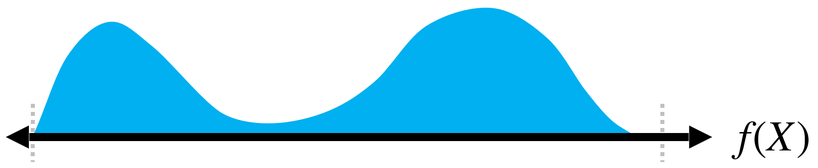
$$\mathbb{E}[\text{fraction of interchanged jobs}] \leq \text{Var}(Y)(\epsilon_0 + \epsilon_1) - (\kappa_1 + \kappa_2)$$

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

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

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



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

Outline

1. Survey of different approaches
2. Algorithms with calibrated machine learning predictions [ICML'25]
3. **Probe LLM reasoning through the lens of optimization** [ICML'26]
4. Conclusions and future directions

Yu He

Yingxi Li

Colin White

We need an algorithmic understanding of GenAI

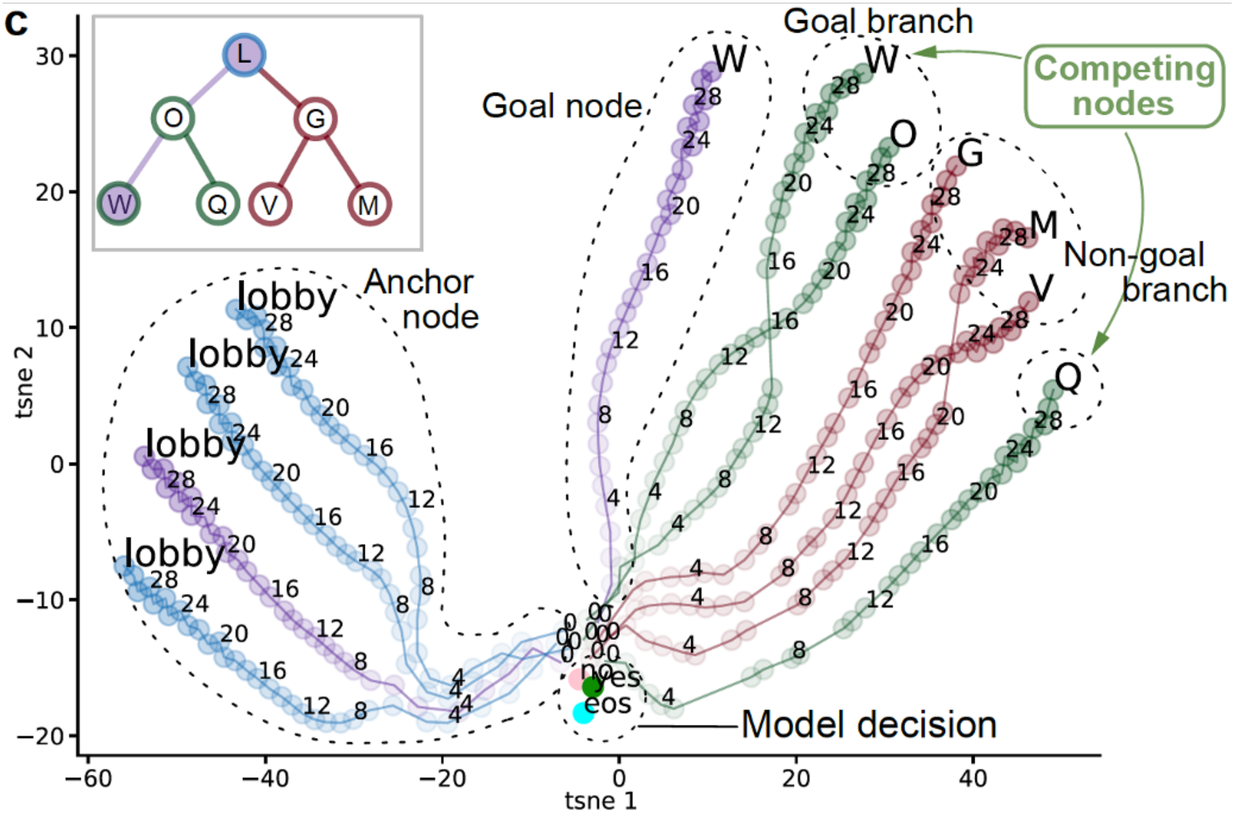
- LLMs increasingly handle **complex, multi-step** real-world decision tasks
  - Planning, analysis, and structured problem-solving
- These tasks require more than language generation:
  - Models must **execute operations, update state, compose steps**
- **Central question:** How do LLMs reason?
  - Determine *how* models compute, not just *what* they predict
- Why now?
  - Scaling is hitting limits: diminishing returns on larger models
  - To get around this, key to understand/improve reasoning mechanisms

[See, e.g., "Position: We Need An Algorithmic Understanding of Generative AI," Eberle et al., ICML'25]

We need an algorithmic understanding of GenAI

When Llama asked if W is reachable from L, how does it reason?

- t-SNE: room-token embeddings
- Does it run Dijsktra?
  - It doesn't seem like it…
- Embeddings separate over layers
- Goal, non-goal branches diverge
- How is the model reasoning?



"Position: We Need An Algorithmic Understanding of Generative AI," Eberle et al., ICML'25

Benchmarks don’t isolate structural reasoning

- Existing benchmarks are domain-specific, atomic skills may be **entangled**
  - E.g., math/STEM [Liu et al., NeurIPS’23, EMNLP’23; Hendrycks et al., NeurIPS’21]
- Coding benchmarks measure **generation**, not **inherent reasoning**
  - Interpreters offload computation; online code risks contamination
  - **Our focus: inherent** reasoning without code
    - Tool use can obscure the model’s inherent reasoning ability
    - In line with recent initiatives like Google & OpenAI performance in IMO competitions
      - Prohibit code/proof assistance, emphasizing end-to-end reasoning
- Algorithmic benchmarks still diagnose too **coarsely** [Markeeva et al., DMLR’26]
  - Complex responses obscure algorithmic primitives
  - Failures difficult to localize precisely

Data Structure Reasoning (DSR) Benchmark

TCS lens for evaluation: use primitives to study generalization, reliability

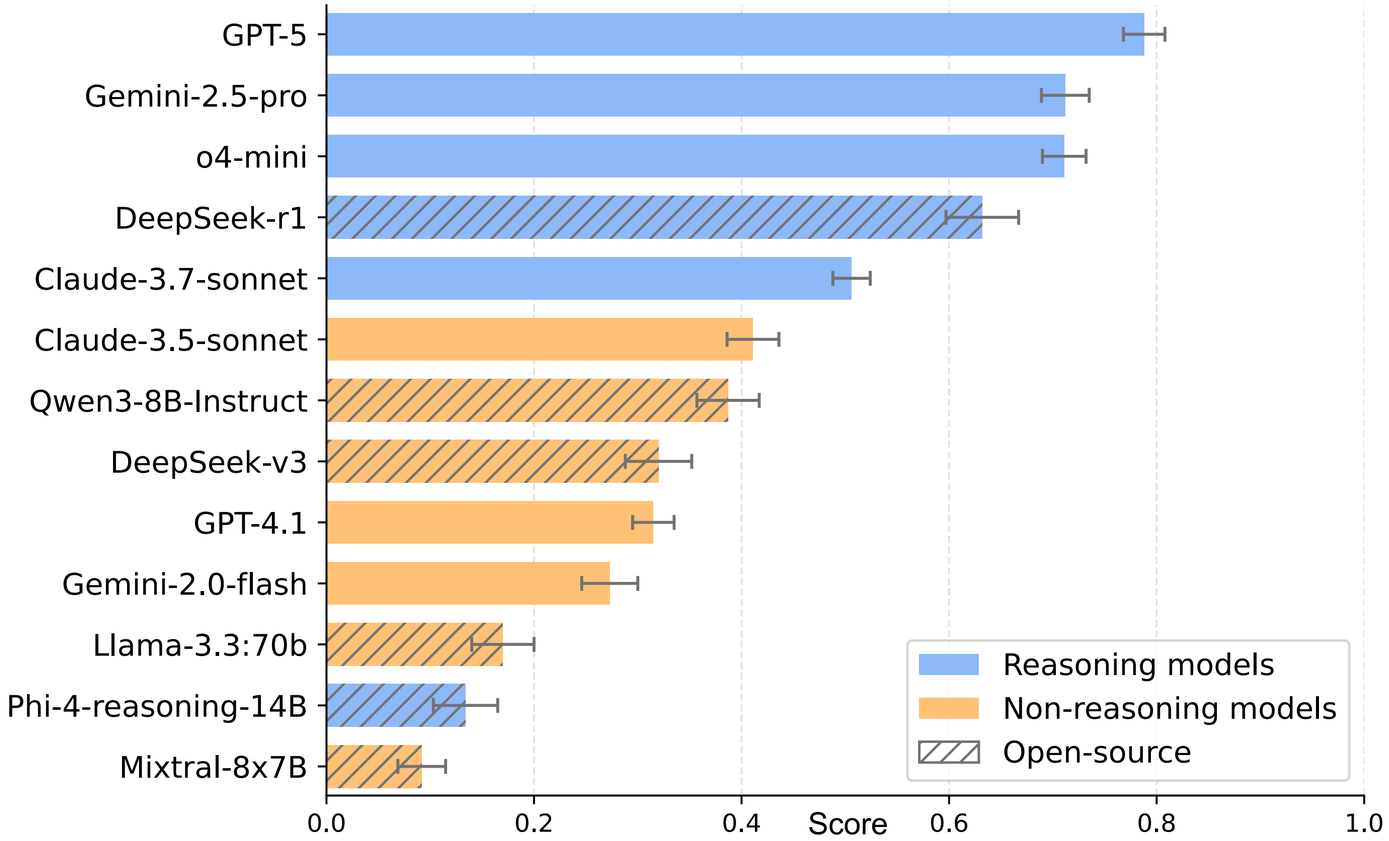
- We introduce **DSR-Bench** for structural reasoning evaluation
  - 20 data structures; 35 operations; 4,140 instances
- Organizes tasks **hierarchically** by reasoning complexity
  - Simple tasks support complex tasks
- Uses **synthetic generation** and **deterministic scoring**
  - Minimizes contamination; avoids subjective judging
- Produces interpretable **failure-mode diagnostics**
  - Pinpoints where structural reasoning breaks down

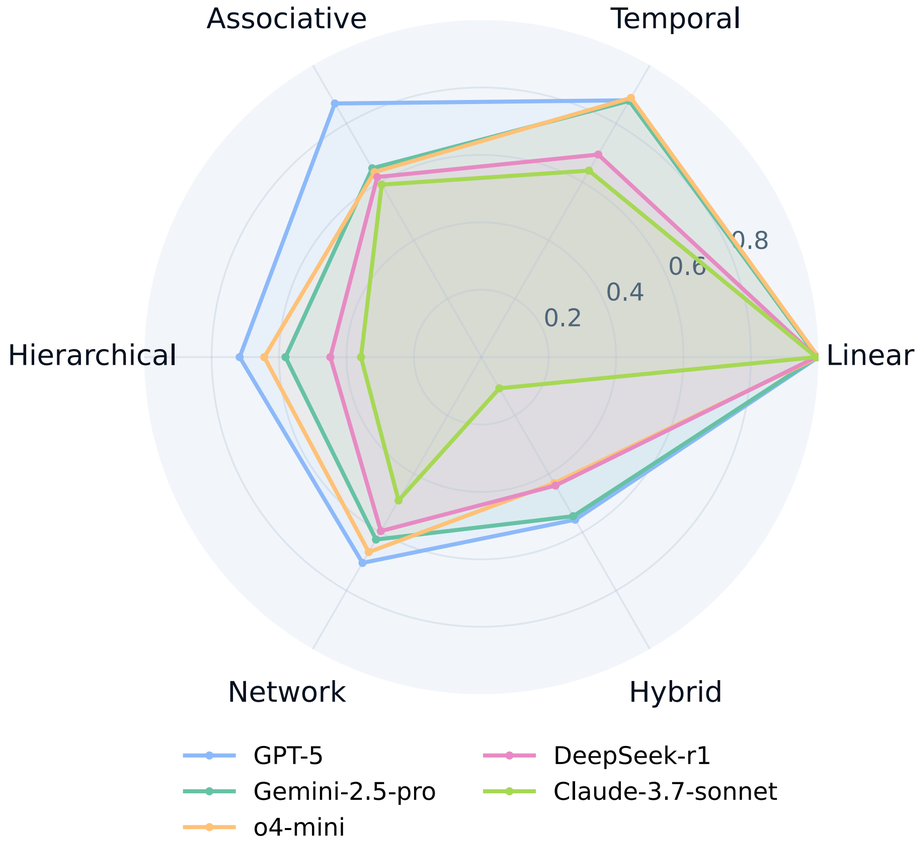
He, Li, White, **Vitercik**, ICML'26

Example prompt

- A queue maintains a first-in, first-out (FIFO) order
  - Items are added at one end and removed from the other
- There are two operations:
  - (`enqueue, k`) adds `k` to the back
  - (`dequeue`) removes the front
- You are given an empty queue initially
- What is the final queue after: (`enqueue, 49`), (`enqueue, 85`), (`dequeue`), …?

Aggregated scores





He, Li, White, **Vitercik**, ICML'26

Where composition breaks

Multi-attribute and multi-hop failure analysis

- **Composition** fails when state has multiple attributes
  - E.g., priority queues add priorities and insertion order
- **Multi-hop** structures stress memory across levels
  - E.g., red-black trees require ancestral balance updates

Structure	GPT-5	Gemini-2.5-Pro	o4-mini	DeepSeek-R1	Claude-3.7-Sonnet	Take-away
Queue	1.00	1.00	1.00	0.98	1.00	FIFO state is mostly solved
Priority queue	0.52	0.51	0.55	0.65	0.28	Add priority/tie-breaking: accuracy drops
BST	1.00	0.97	0.86	0.73	0.64	Basic tree structure is easier
RB tree	0.76	0.49	0.65	0.62	0.30	Balance constraints are harder

He, Li, White, **Vitercik**, ICML'26

Length generalization is still weak

Take-away: Many models degrade sharply with input length

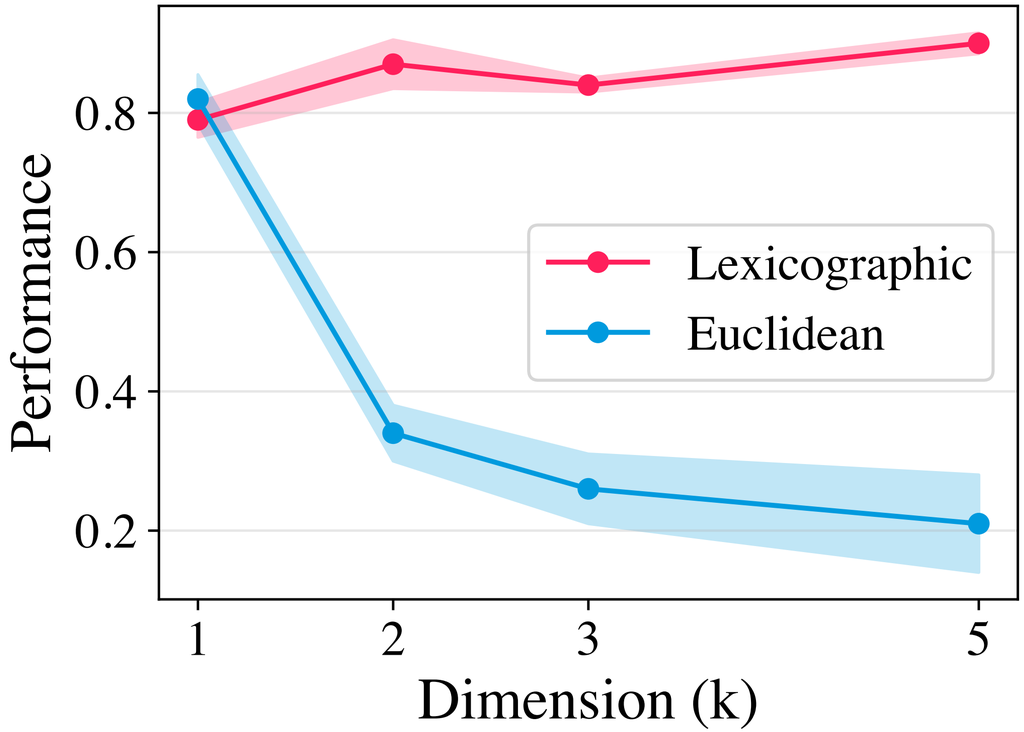
Short = 5–10; Medium = 11–20; Long = 21–30 inputs/operations

Structure / operation	GPT-5	Gemini-2.5-Pro	DeepSeek-R1	Claude-3.7-Sonnet
Queue compound	1.00→1.00→1.00	1.00→1.00→1.00	1.00→1.00→0.97	1.00→0.93→0.98
Priority Queue compound	0.84→0.44→0.28	0.89→0.41→0.23	0.92→0.54→0.48	0.70→0.11→0.04
RB Tree compound	0.91→0.67→0.49	0.91→0.41→0.13	0.91→0.37→0.03	0.57→0.03→0.00
Graph DFS	0.93→1.00→0.96	1.00→0.81→0.14	0.80→0.58→0.22	0.50→0.11→0.00

Specifications can lose to learned priors

Take-away: Explicit constraints don’t override learned defaults

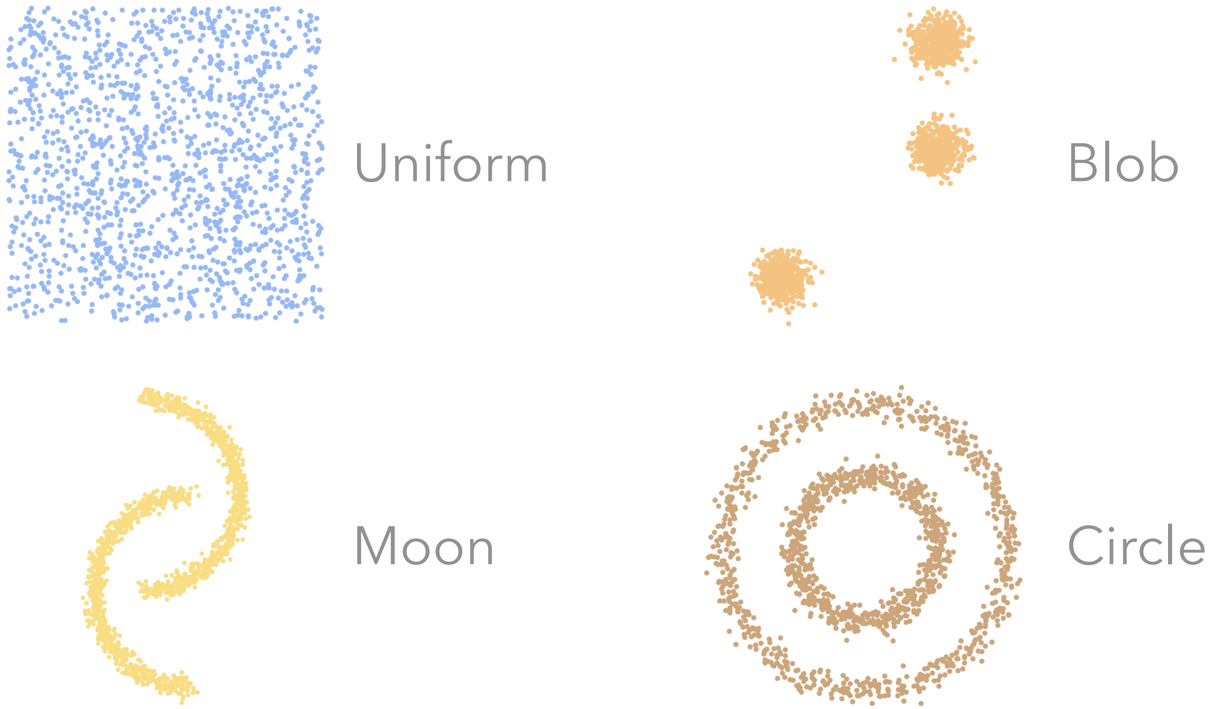
o4-mini on K-D Heap with different tie-breaking rules

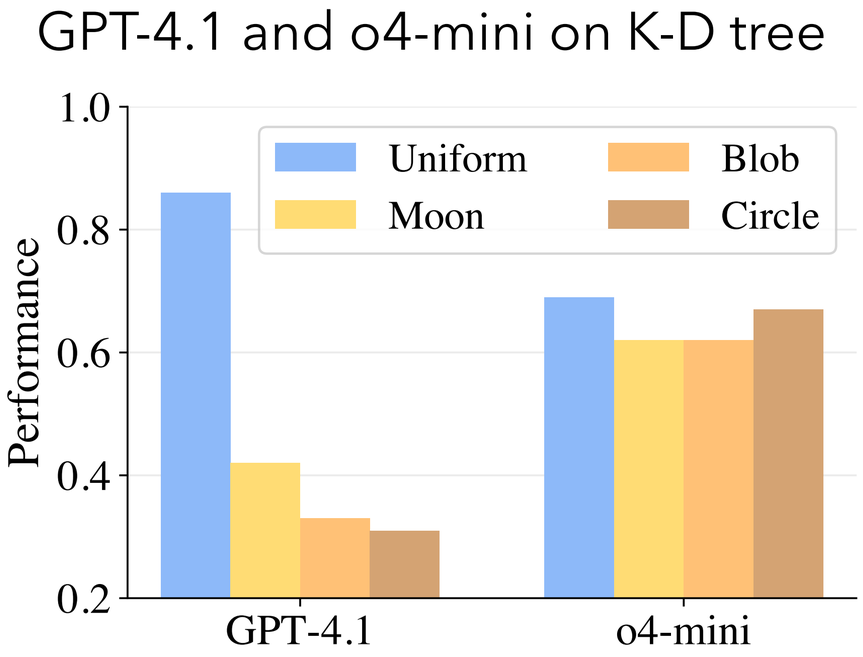


He, Li, White, **Vitercik**, ICML’26

Distribution shift ➡ non-algorithmic behavior

Auxiliary spatial probe varies dimensionality and input distribution





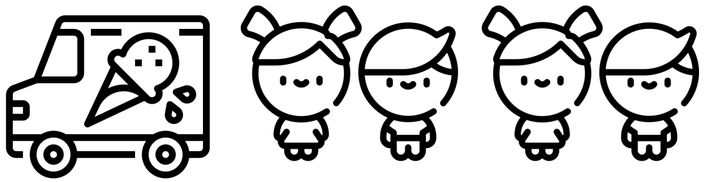
Take-away: Robust decisions shouldn’t depend on shortcuts, familiar patterns

He, Li, White, **Vitercik**, ICML’26

Auxiliary probe: Natural language

Realistic probe wraps structures in narrative scenarios

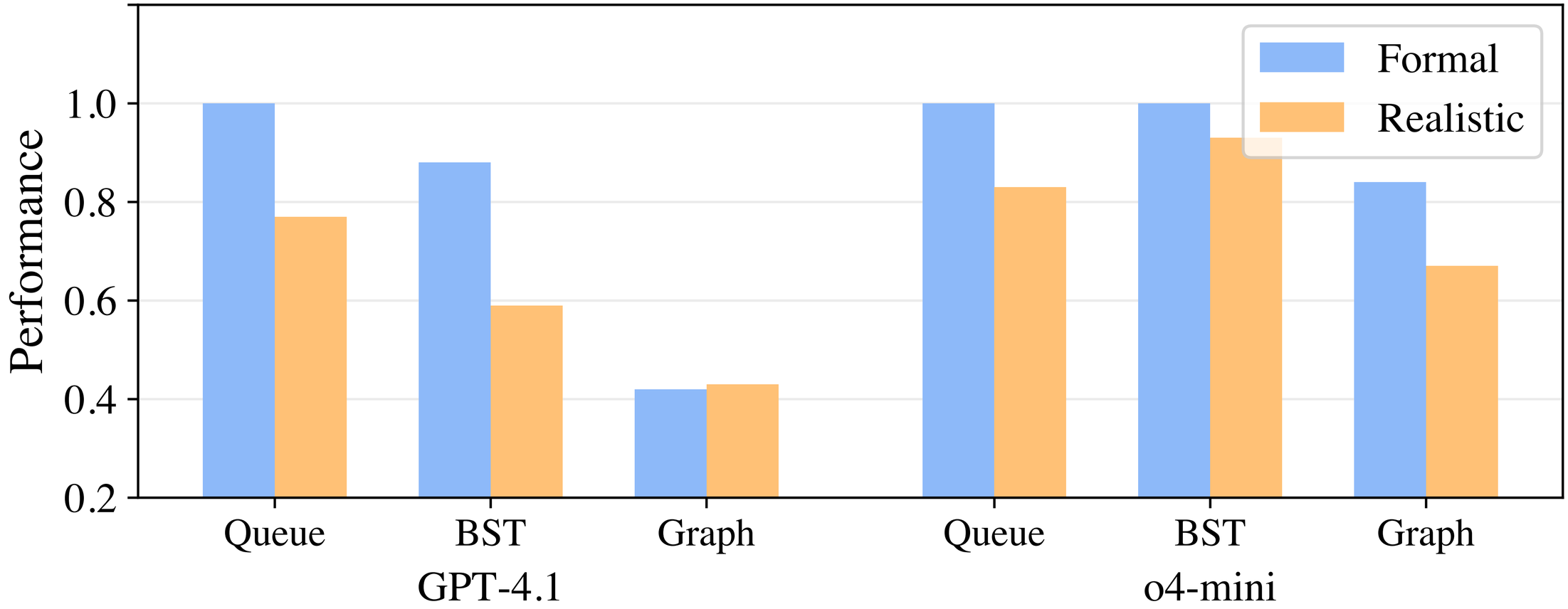
- On a sunny afternoon in the park, an ice cream truck rolled in…
- Children began to form a line, each newcomer taking their place at the end while the vendor served from the front...
  - Isabella Miller ran over and joined the ice cream line.
  - The next kid in line was served promptly.
- **Q:** What is the order of the remaining kids in line?



- Joining line means enqueue; served means dequeue

Auxiliary probe: Natural language

Take-away: Performance drops versus formal descriptors

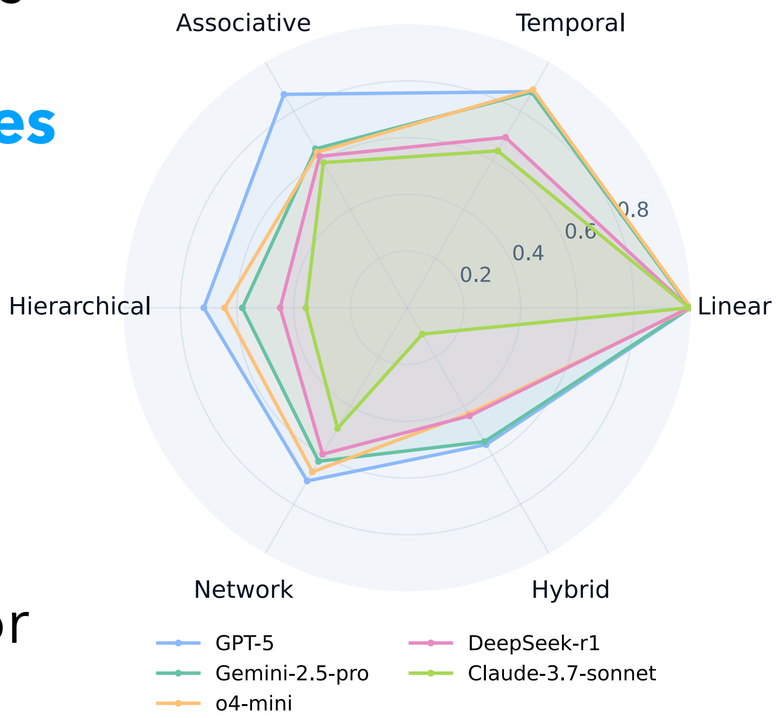


He, Li, White, **Vitercik**, ICML'26

Take-aways

LLM structural reasoning

- LLM reasoning requires more than fluent generation
  - Execute operations, update state, compose steps
- DSR-Bench isolates **structural reasoning primitives**
  - Diagnostic tasks reveal localized failures
- **Composition** and **length** remain brittle
- Specifications can lose to **learned priors**
- **Distribution shift** reveals non-algorithmic behavior



Conclusions

Machine learning for combinatorial optimization

- **Overall goal:** Exploit structure, but keep algorithmic safeguards
  - Certificates, robustness guarantees
- **Part 1:** Predictions help when uncertainty is explicit
  - Calibration makes trust algorithmically usable
  - Better (sharper) predictions can yield better guarantees
- LLMs broaden the algorithm-design interface
- **Part 2:** Structural reasoning is still a bottleneck for LLMs
  - Operations, state, composition, specification-following

Ongoing directions

- **Part 1:** generalize themes to a more abstract framework
  - Analyze how algorithm performance depend on **predictor properties**
  - MSE, calibration, sharpness, and inherent uncertainty
  - False positive/negative rate [see also, e.g., Anand et al. ’20]
  - **Ultimate goal:** Guide model choice and training for decision tasks

- **Part 2:** Use **verifiable tasks** for targeted RL
  - Reward exact intermediate states, not just answers
  - Apply **interpretability** methods to structural failure modes
    - Identify circuits for updates, memory, traversal

Machine Learning for Combinatorial Optimization

Ellen Vitercik · Stanford University