Machine Learning for Discrete Optimization: Theoretical Foundations

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 rideshare)

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

ML for discrete optimization

A rich history

ML for discrete optimization: Key challenges

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

ML/AI methods applied to discrete optimization **rarely work out of the box**:

structure of the algorithmic task should be accounted for

[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq

  E.g., Vinyals et al., NeurIPS'15; Bello et al., '17; …

ML for discrete optimization: Key challenges

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

ML/AI methods applied to discrete optimization **rarely work out of the box**:

structure of the algorithmic task should be accounted for

[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 discrete optimization: Key challenges

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

ML/AI methods applied to discrete optimization **rarely work out of the box**:

structure of the algorithmic task should be accounted for

[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 discrete optimization: Key challenges

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

ML/AI methods applied to discrete optimization **rarely work out of the box**:

structure of the algorithmic task should be accounted for

[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq, GNNs, transformers, LLMs, …
- **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;
Balcan, Dick, Sandholm, **V**, JACM'24; Lawless, Li, Wikum, Udell, **V**, CPAIOR'25; …

ML for discrete optimization: Key challenges

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

ML/AI methods applied to discrete optimization **rarely work out of the box**:

structure of the algorithmic task should be accounted for

[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq, GNNs, transformers, LLMs, …
- **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; He, **V**, ICML'25…

ML for discrete optimization: Key challenges

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

ML/AI methods applied to discrete optimization **rarely work out of the box**:

structure of the algorithmic task should be accounted for

[e.g., surveys by Bengio et al., '18; Cappart et al., '23; …]

Major challenges:

- **Architecture:** seq2seq, GNNs, transformers, LLMs, …
- **Integration:** integrate with existing method or train end-to-end method?
- **Training:** how to train 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. **GNNs for online matching** [Hayderi, Saberi, **V**, Wikum, ICML'24]
3. Calibration for online decision-making [Shen, **V**, Wikum, ICML'25]
4. Conclusion and future directions

Alexandre Hayderi

Amin Saberi

Anders Wikum

Online matching

Critical problem in many digital marketplaces

Advertising: website visitors matched to ads
[e.g., Mehta et al., FnT-TCS'13]

Crowdsourcing platforms: crowdworkers matched to tasks
[e.g., Tong et al., VLDB'20]

Rideshare: riders matched to drivers
[e.g., Zhao et al., AAAI'19]

Medicine: organ donors matched to patients
[e.g., Ezra et al., EC'20]

Primary challenge: **irrevocable** matching decisions must be made online **without** precise knowledge of how demand will evolve

🔭 Overview of contributions

Empirical: train a GNN for Online Bayesian Bipartite Matching (OBBM)

- Supervise using (exponential time) optimal online policy
  - Train on small graphs, test on large graphs
- Outperforms baselines on synthetic/semi-synthetic graphs

Theoretical: justify the suitability of GNNs for OBBM

Key challenge: sheer complexity of optimal online policy!

NP-hard to provide an α-factor approximation (for some constant α)

[Papadimitriou et al., EC'21]

Online Bayesian Bipartite Matching (OBBM)

Bipartite graph $G = (L, R, E)$ with offline nodes L, online nodes R

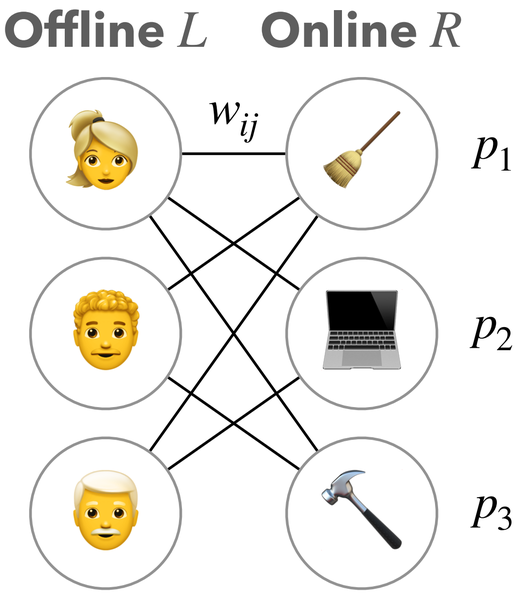
- Nodes represent **crowdworkers** and **tasks**
- Edge weights w_{ij} represent **workers' payoffs** for completing tasks

Online node arrival probability $p_t \in [0,1]$ for each $t \in R$

If node t appears, irrevocably decide:

- Match t with an unmatched offline neighbor, or
- Skip t and not match it to any node

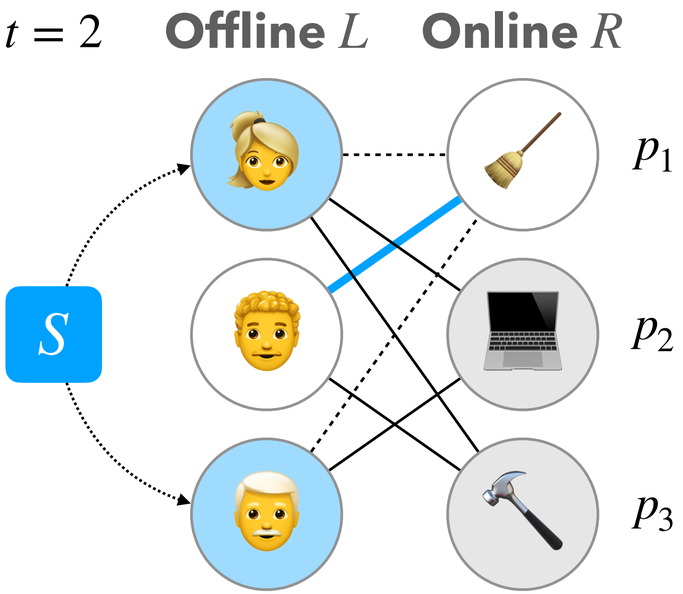
Goal: compute a high-weight matching



Online optimal algorithm and value-to-go

Let S be the set of offline nodes available on round t

OPT_{on} computes the **value-to-go (VTG)** function (or *Bellman equation*) $V_G(S, t)$



Online optimal algorithm and value-to-go

Let S be the set of offline nodes available on round t

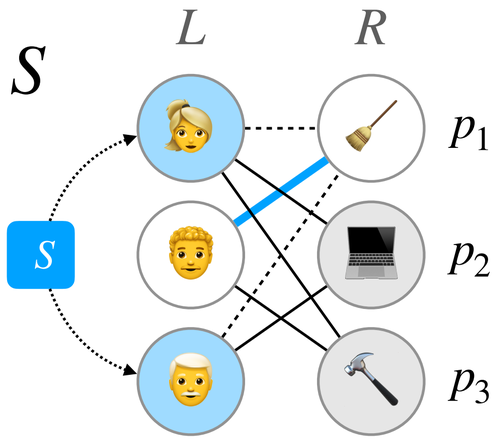
OPT_{on} computes the **value-to-go (VTG)** function (or *Bellman equation*) $V_G(S, t)$

$V_G(S, t)$ = expected value of the maximum weight matching achievable on G

- Expectation over arrivals $\{t, \ldots, |R|\}$
- Matchings restricted to the set of remaining offline nodes S

OPT_{on} selects the action that maximizes the VTG:

- Value of **matching to node** $u \in S$: $w_{tu} + V_G(S \backslash \{u\}, t+1)$
- Value of **skipping**: $V_G(S, t+1)$



Outline

1. Introduction
2. GNNs for online matching [Hayderi, Saberi, **V**, Wikum, ICML'24]
    i. **ML setup and experiments**
    ii. Theoretical guarantees
3. Calibration for online decision-making [Shen, **V**, Wikum, ICML'25]
4. Future directions

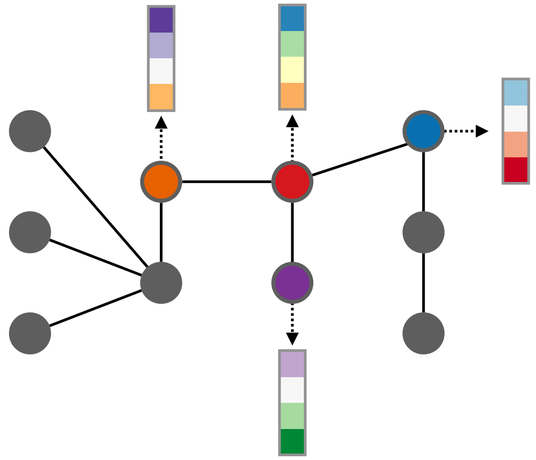
GNN architecture

Example

Each node v begins with initial embedding $\vec{h}_v^{(0)}$ (input features)

On iteration (or "layer") $k \in \{1,\ldots,L\}$, for each node v:

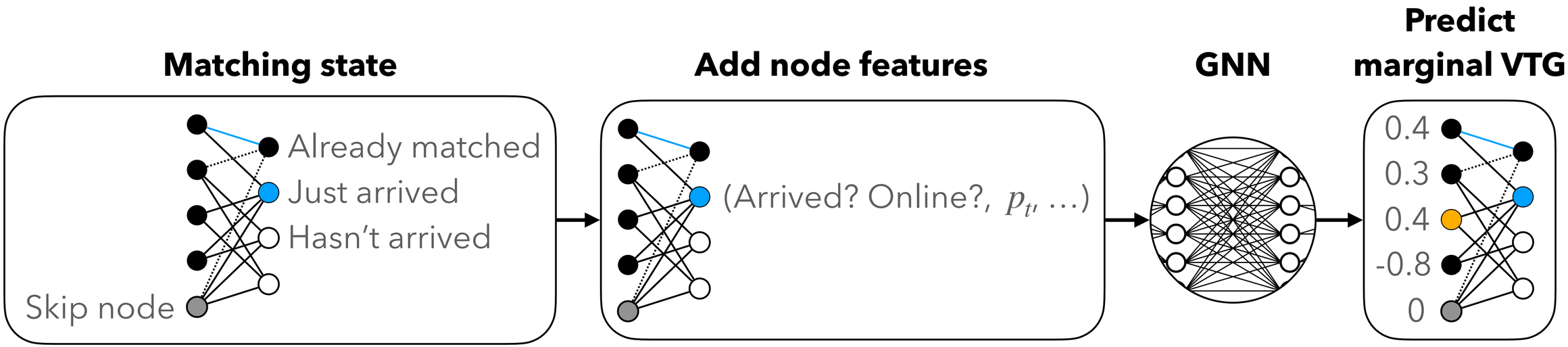
1. Aggregate neighbors' embeddings $\overrightarrow{m}_v^{(k)} = \max_{u \in N(v)} \left\{ \text{MLP}\left(\vec{h}_u^{(k-1)}, w_{vu}\right) \right\}$
2. Update node embedding $\vec{h}_v^{(k)} = \text{MLP}\left(\vec{h}_v^{(k-1)} + \overrightarrow{m}_v^{(k)}\right)$



Use $\vec{h}_v^{(L)}$ to make predictions about node v

MAGNOLIA

Matching **A**lgorithms via **GN**Ns for **O**n**li**ne Value-to-go **A**pproximation



Supervise with marginal VTG $\left(w_{tu} + V_G\left(S\backslash\{u\}, t+1\right)\right) - V_G(S, t+1)$

Value of matching with node u — Value of skipping

Graph datasets

Random graph families

- Erdős–Rényi graphs with U(0,1) edge weights
- Barabási–Albert graphs
- Random geometric graphs

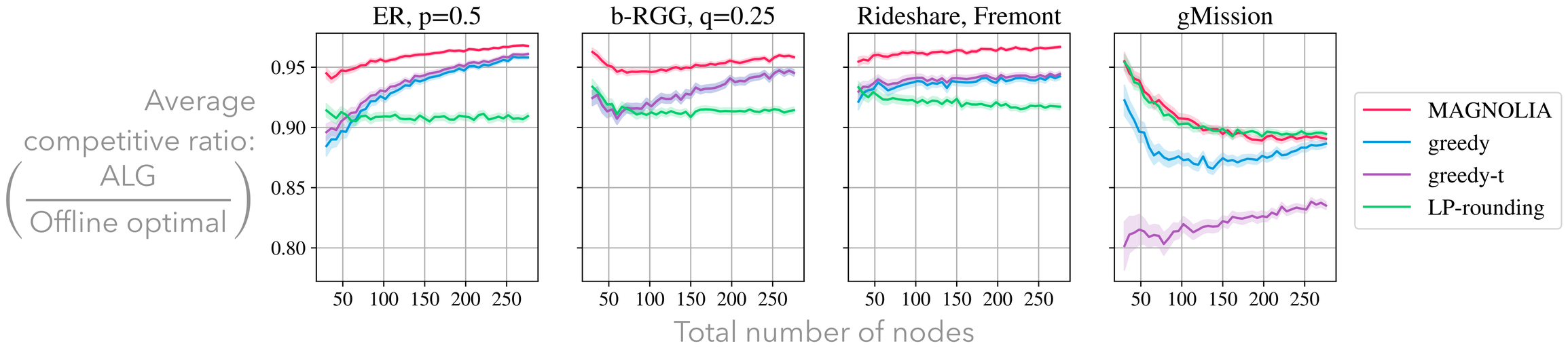
} **Training data:** mix of these, 6x10 nodes

Semi-synthetic graphs

- gMission: crowdsourcing data; workers → tasks
  [Chen et al., VLDB'14]
- OSMnx: library for for analyzing street networks
  [Boeing, Comput. Environ. Urban Syst. '17]

Experiments: Size generalization

Performance generalizes to graphs up to 20x larger than training graphs



- greedy: selects highest weight available edge
- greedy-t: selects greedy edge if weight exceeds threshold, otherwise skips
- LP-rounding: pen-and-paper 0.632-approximation to optimal online algorithm

  Papadimitriou et al. [EC'21], **Braverman et al. [EC'22]**, Naor et al. ['23]

Outline

1. Introduction
2. GNNs for online matching [Hayderi, Saberi, **V**, Wikum, ICML'24]
    i. ML setup and experiments
    ii. Theoretical guarantees
3. Calibration for online decision-making [Shen, **V**, Wikum, ICML'25]
4. Future directions

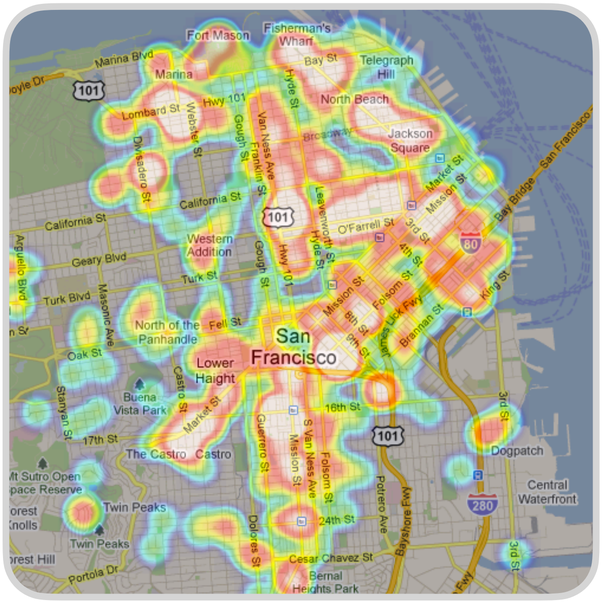
Why are GNNs well-suited for online matching?

Most practical instances are “well-structured”

We show this rigorously for **random geometric graphs**

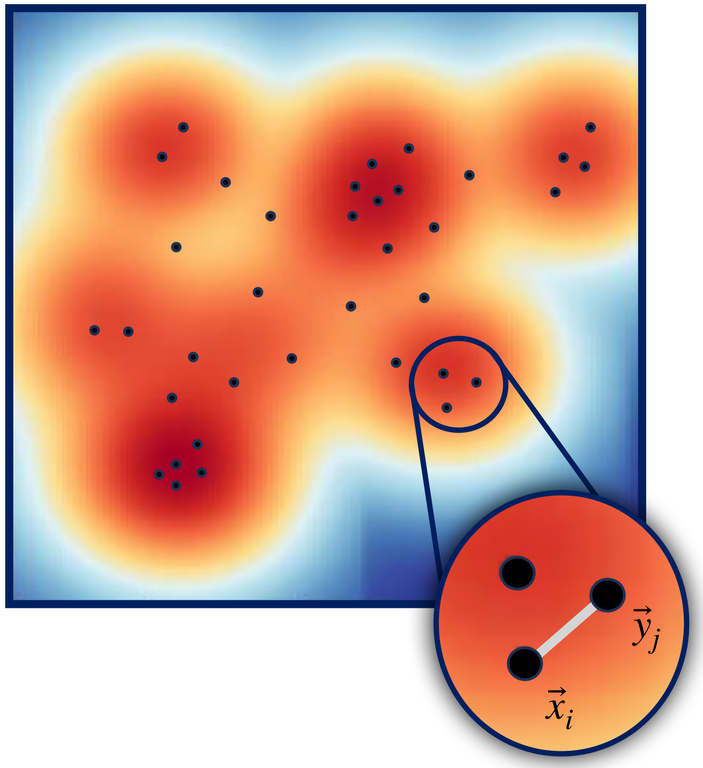
- *We prove that under these families,* ***VTG is nearly local***
- Applications:
  - Dynamic spatial matching [Kanoria ’23]
  - Opinion dynamics [Zhang et al. ’14]
  - Contagion in wireless networks [Nekovee ’07]

Already known that **GNNs compute local functions**



Uber demand heatmap

Random geometric graphs $\mathcal{G}(|L|, |R|, D, \Delta)$



Draw $\vec{x}_1, \ldots, \vec{x}_{|L|} \sim D$ and $\vec{y}_1, \ldots, \vec{y}_{|R|} \sim D$

In this talk, D = Uniform $\left([0,1]^d\right)$, generalized in paper

Connect nodes $i \in L, j \in R$ if $\left\| \vec{x}_i - \vec{y}_j \right\|_\infty \leq \Delta$

GNNs learn local graph functions

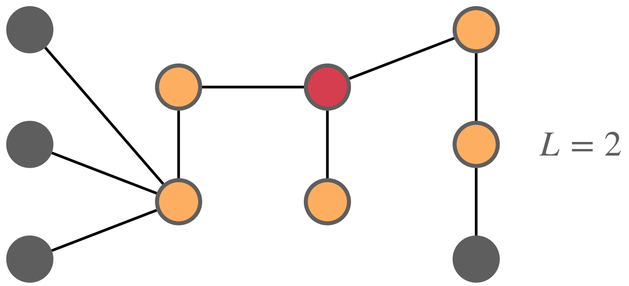
E.g., Tahmasebi et al., AISTATS'23

A function f over graphs is **r-local** if

$f(G)$ only relies on info in the **r-hop neighborhood** $N_r(v)$ of each node v

i.e., exist functions ϕ, ψ such that $f(G) = \phi\left(\left\{\psi\left(N_r(v)\right)\right\}_{v \in V}\right)$

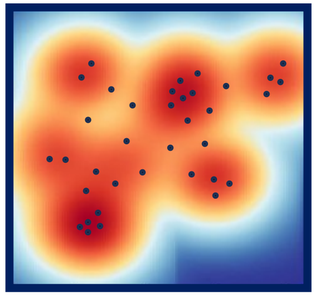
Observation: GNNs with L layers compute L-local functions



Value-to-go is locally approximable

Function f is (r, ϵ, δ)-**locally approximable** over a graph distribution $\mathcal{G}$ if:
there exists an r-local function h such that $\left|f(G) - h(G)\right| \leq \epsilon f(G)$
with probability $1 - \delta$ over $G \sim \mathcal{G}$

Main theorem (informal): For $\epsilon, \delta \in (0,1)$ and $\Delta = O\left((|L| + |R|)^{-1/d}\right)$,
VTG is $(\log(|L| + |R|), \epsilon, \delta)$-**locally approximable** over $\mathcal{G}(|L|, |R|, D, \Delta)$

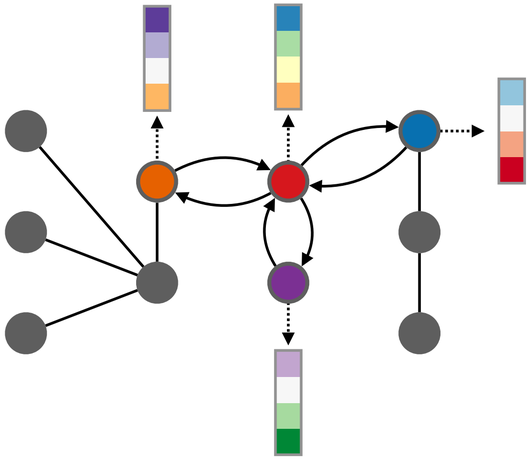


Proof intuition:

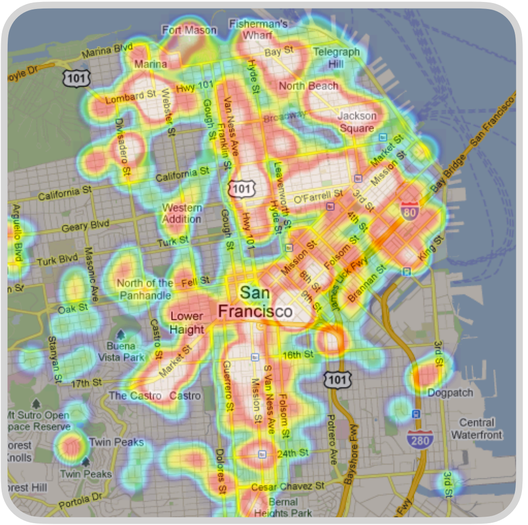
- Graphs can be split into "clusters" with few inter-cluster edges
- Matching within clusters is nearly optimal

Correspondance between GNNs and VTG

GNNs compute “local” functions



Bipartite random geometric graphs:
VTG approximable by local function



Uber demand heatmap

Outline

1. Introduction
2. **GNNs for online matching** [Hayderi, Saberi, **V**, Wikum, ICML'24]
3. Calibration for online decision-making [Shen, **V**, Wikum, ICML'25]
4. Conclusion and future directions

Major challenges:

- ✅ Architecture
- ✅ Integration
- ✅ Training
- ✅ Robustness

Alexandre Hayderi

Amin Saberi

Anders Wikum

Outline

1. Introduction
2. GNNs for online matching [Hayderi, Saberi, **V**, Wikum, ICML'24]
3. **Calibration for online decision-making** [Shen, **V**, Wikum, ICML'25]
4. Conclusion and future directions

Major challenges:

Architecture

✅ Integration

Training

✅ Robustness

Judy Hanwen Shen

Anders Wikum

Decision-making under uncertainty

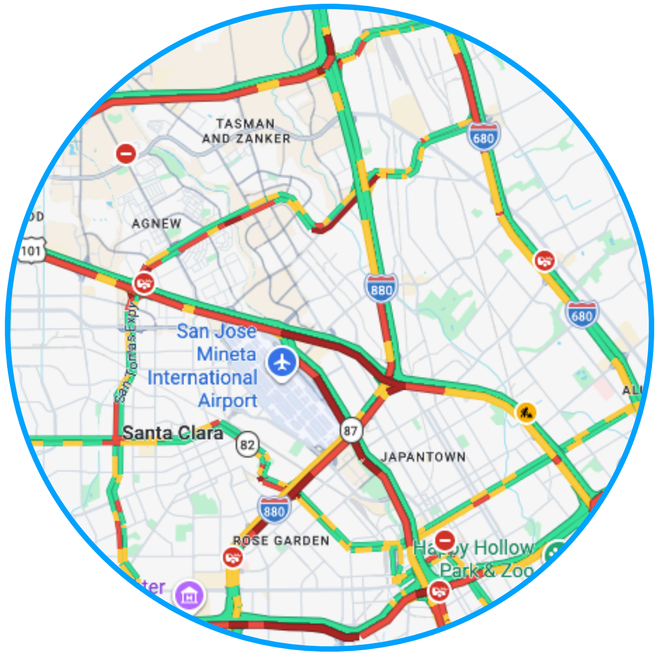
In practice, many aspects of inputs are **unknown** a priori

- 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**

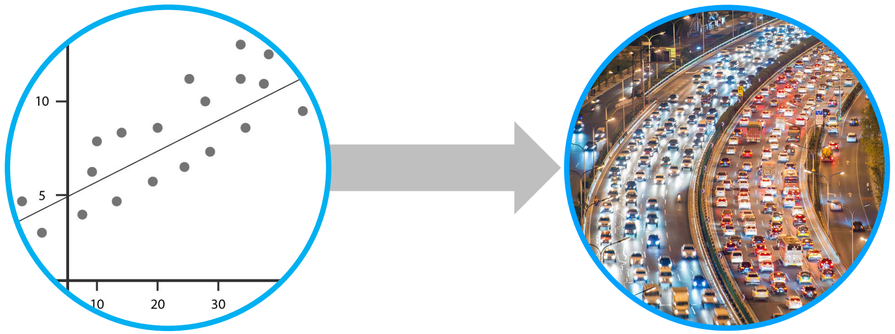
  [e.g., book chapter by Mitzenmacher, Vassilvitskii, ‘20]



Algorithms and prediction uncertainty

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

Don’t blindly trust predictions



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

Additional related work

Decision-making under uncertainty + machine learning + theory

Algorithms with predictions

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

This talk: most in this style

Calibration for decision-making

[e.g., Foster, Vohra, GEB’97; Zhao et al., NeurIPS’21; Kleinberg et al., COLT’23; Roth, Shi, EC’23; Noarov et al., ICML’25; Kiyani et al., ICLR’26]

Data-driven robust optimization

[e.g., Esfahani, Kuhn, ’17; Bertsimas et al., ’18; …]

Decision-focused learning

[e.g., survey by Mandi et al., JAIR’24]

Additional related work

Algorithms with predictions

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

Calibration for decision-making

[e.g., Foster, Vohra GEB’97; Zhao et al., NeurIPS’21; Kleinberg et al., COLT’23; Roth, Shi, EC’23; Noarov et al., ICML’25; Kiyani et al., ICLR’26]

Data-driven robust optimization

[e.g., Esfahani, Kuhn, ’17; Bertsimas et al., ’18; …]

Decision-focused learning

[e.g., survey by Mandi et al., JAIR’24]

Ongoing work: better unification of these styles

Additional related work

Algorithms with predictions

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

- Khodak et al. [NeurIPS’22]: Learning prediction reliability online
  - Similar in motivation, but not through lens of statistical UQ

- Sun et al. [ICML’24]: Algorithms with conformal ML predictions
  - We show calibration has key advantages over conformal methods
  - Especially helpful when predictions have high variance

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

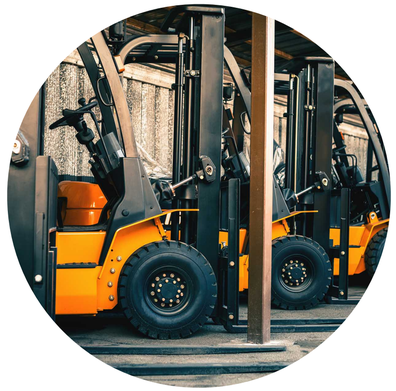
Outline

1. Introduction
2. GNNs for online matching [Hayderi, Saberi, **V**, Wikum, ICML'24]
3. Calibration for online decision-making [Shen, **V**, Wikum, ICML'25]
    i. **Rent-or-buy**
    ii. Job scheduling
4. Conclusion and future directions

Online rent-or-buy

- Models buy-vs-rent under unknown horizon
- Balances upfront cost against ongoing rental expenses

Cloud:
on-demand vs
reserved instances



Equipment:
lease machinery vs
purchase outright

Mobility:
transit passes,
bike share

Ski rental problem

Prototypical online rent-or-buy decision-making problem

- Skier will ski for some **unknown** number of days $Z \in \mathbb{R}_+$
- Each day, decide to rent skis for \$1 or buy for one-time cost \$$b$
- **Goal**: Minimize total skiing cost
- Worst-case "breakeven" strategy:
  Rent for b days, and if still want to ski, buy [Karlin et al. '01]

$$\text{Competitive ratio (CR)} := \frac{\text{ALG}}{\text{OPT}} = \frac{\text{amount algorithm pays}}{\min\{Z, b\}} \leq 2$$

Ski rental with predictions: Prior work

Algorithm with prediction of # skiing days Z [Kumar et al. NeurIPS'18]

- Uses "trust" parameter $\lambda \in [0,1]$
  - $\lambda = 0$: fully trust predictor; $\lambda = 1$: don't trust predictor at all
- **Algorithm:** If prediction $\geq b$: buy on day $\lceil \lambda b \rceil$. Else, buy on day $\lceil b/\lambda \rceil$.
- **Consistency guarantee:** Perfect prediction yields CR $\leq 1 + \lambda$
  - If you knew to set $\lambda = 0$, recover optimal CR $= 1$
- **Robustness guarantee:** Any prediction yields CR $\leq 1 + \frac{1}{\lambda}$
  - If you knew to set $\lambda = 1$, recover best worst-case CR $= 2$
- **Our goal:** leverage calibration to encode trust/uncertainty

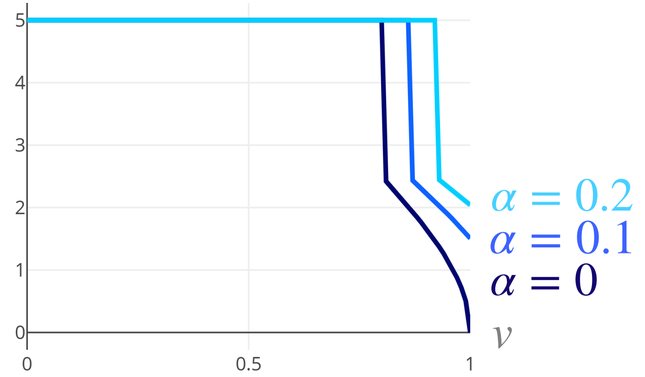
Algorithm with calibrated prediction

- $\mathcal{X} =$ skier features
- Predictor $f(X)$ of target $Y = 1(Z > b)$

  Max calibration error $\alpha = \max_{v \in R(f)} \left| v - \mathbb{P}[Y = 1 \mid f(X) = v] \right|$
- **Algorithm:** given prediction $f(X) = v$ rent for $k(v)$ days

$$k(v) = \begin{cases} b & \text{if } v \leq \dfrac{4 + 3\alpha}{5} \\ b\sqrt{\dfrac{1 - v + \alpha}{v + \alpha}} & \text{else} \end{cases}$$

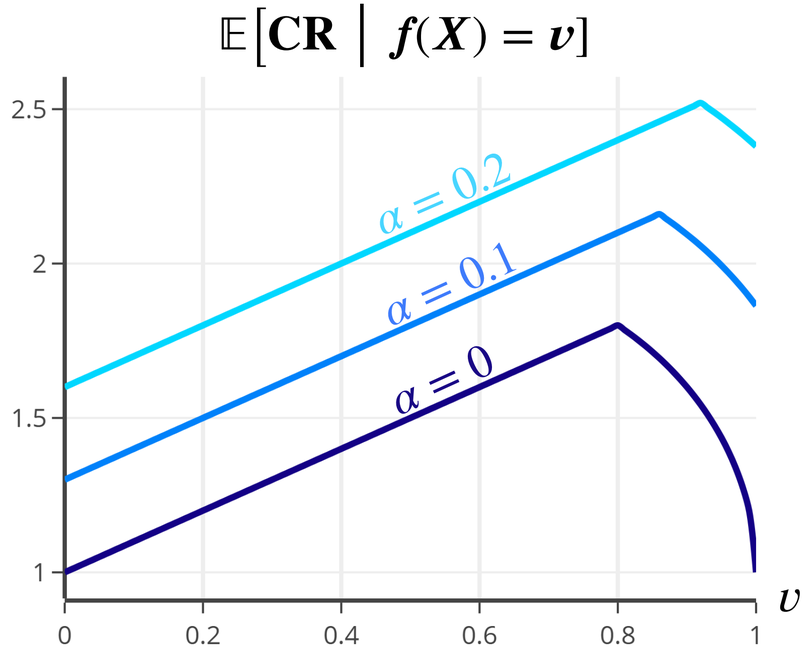
Days rented $k(v)$, $(b = 5)$



Ski rental: Main results

Prediction-wise bound:

$$\mathbb{E}\left[\mathrm{CR} \mid f(X) = v\right] \leq 1 + 2\alpha + \min\left(v + \alpha,\ 2\sqrt{(v+\alpha)(1-v+\alpha)}\right)$$



Ski rental: Main results

Prediction-wise bound:

$$\mathbb{E}\left[\mathrm{CR} \mid f(X) = v\right] \leq 1 + 2\alpha + \min\left(v + \alpha,\ 2\sqrt{(v+\alpha)(1-v+\alpha)}\right)$$

Lower bound: $\forall v$, exists distribution & calibrated predictor s.t.

$$\mathbb{E}\left[\mathrm{CR} \mid f(X) = v\right] \geq 1 + \min\left(v,\ 2\sqrt{v(1-v)}\right)$$

Global bound:

$$\mathbb{E}\left[\mathrm{CR}\right] \leq 1 + 3\alpha + \min\left(\mathbb{P}[Z > b],\ 2\sqrt{\mathrm{MSE}(f) + 3\alpha}\right)$$

✅ Lower MSE and calibration error lead to near-optimal CR

Outline

1. Introduction
2. GNNs for online matching [Hayderi, Saberi, **V**, Wikum, ICML'24]
3. Calibration for online decision-making [Shen, **V**, Wikum, ICML'25]
    i. Rent-or-buy
    ii. Job scheduling
4. Conclusion and future directions

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

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

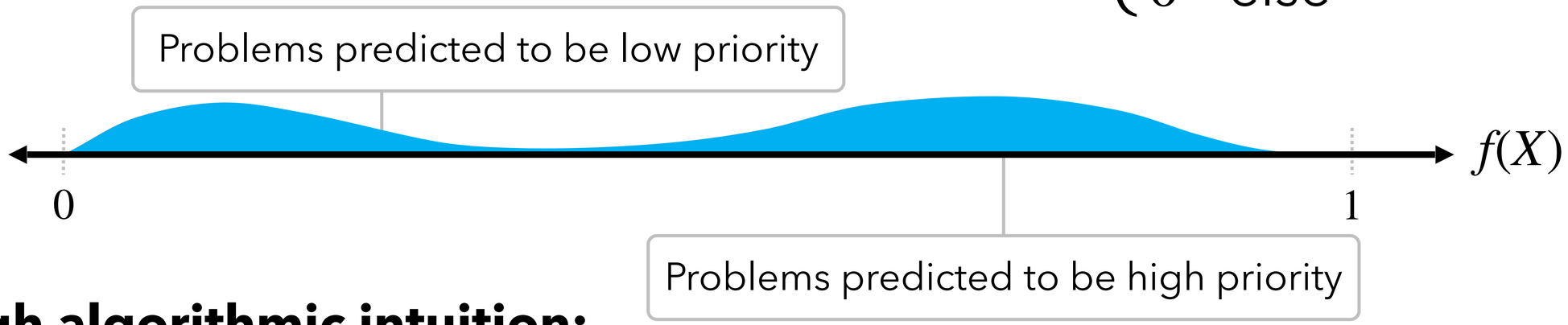
Cost per unit delay, with $\omega_1 > \omega_0 > 0$

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**

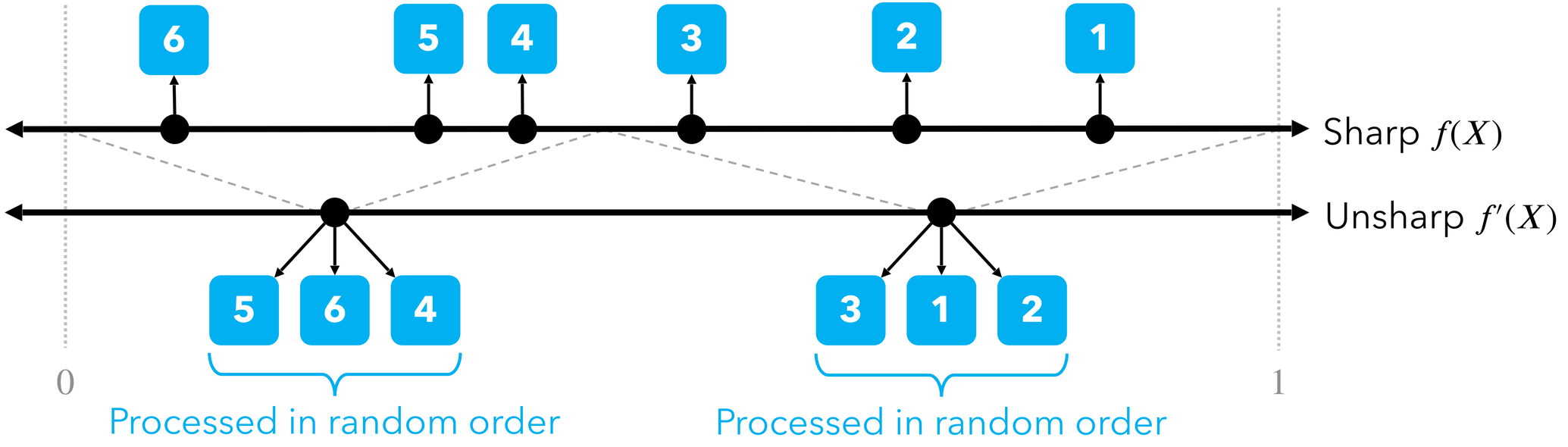
Weighted sum of completion times compared to optimal in hindsight

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

Importance of predictor sharpness

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

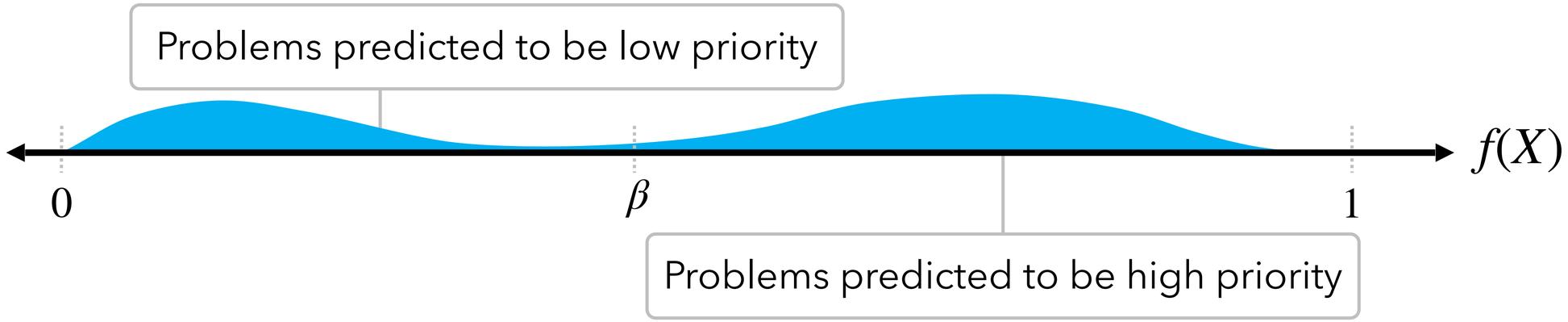
- Low priority job (partially) processed before high priority job
- **Sharp predictors** lead to **fewer interchanges**



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

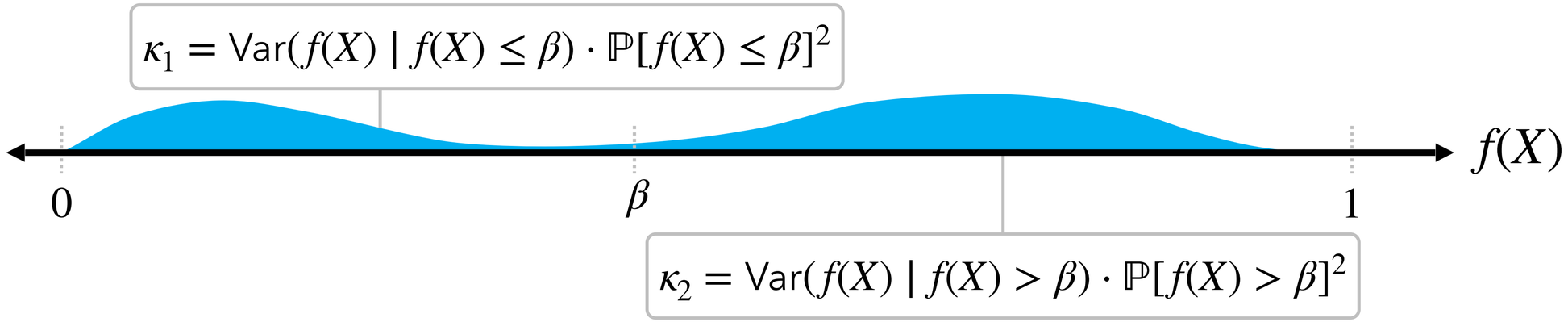
Importance of predictor sharpness

More formally



Importance of predictor sharpness

More formally



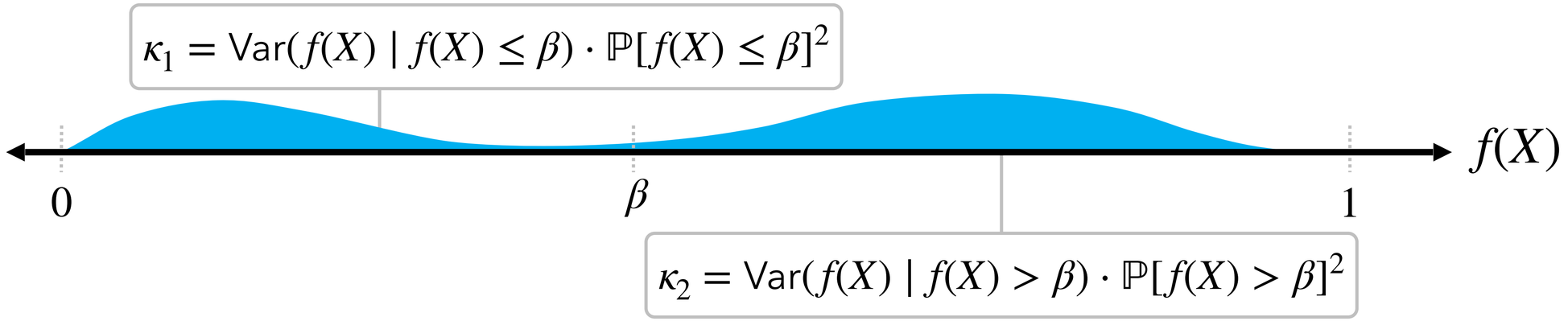
Thm: Let $\mathbb{P}[f(X) > \beta \mid Y = 0] = \epsilon_0, \quad \mathbb{P}[f(X) \leq \beta \mid Y = 1] = \epsilon_1$

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

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

Importance of predictor sharpness

More formally



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

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

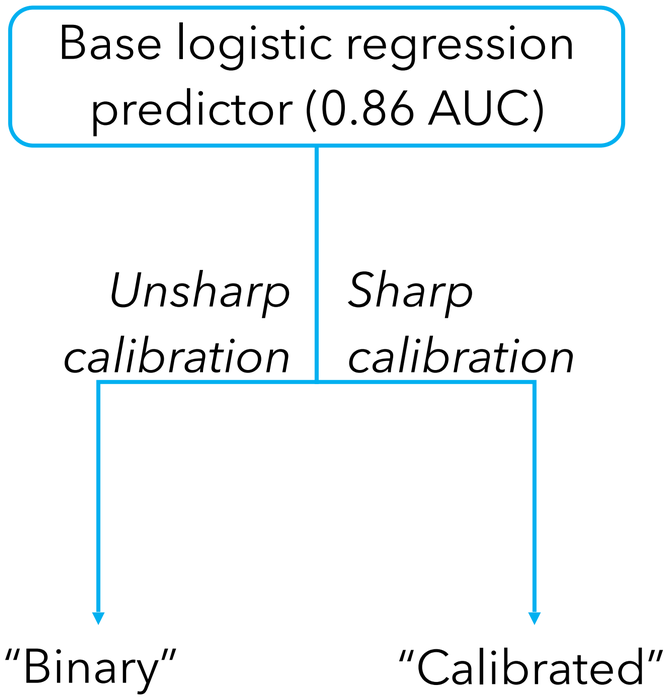
(Eventual) corollary: bound on algorithm's regret (see paper)

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

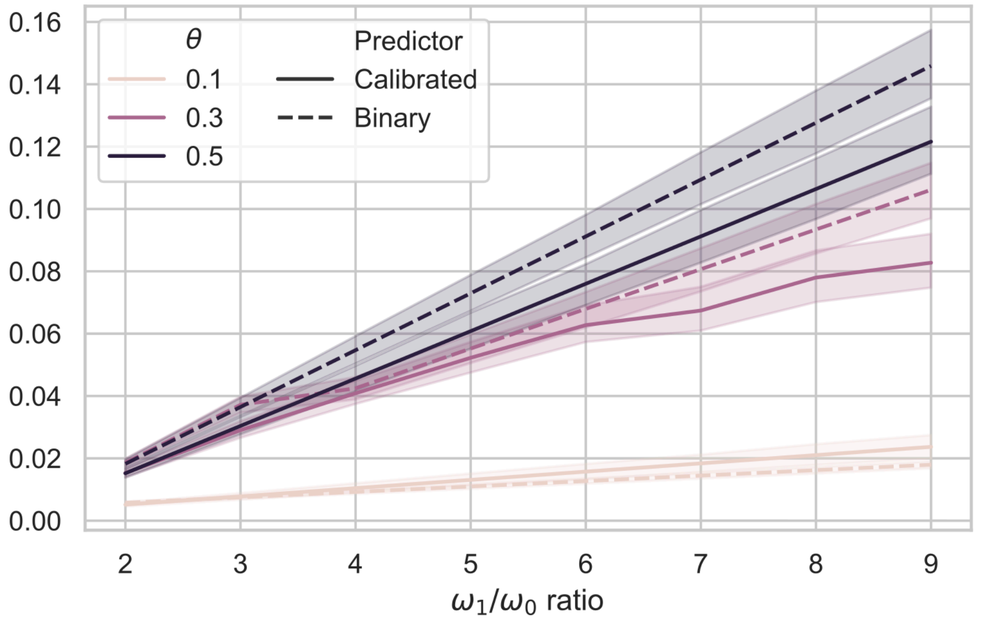
Experiments: Sepsis triage

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

Key challenges in ML for optimization

ML/AI methods applied to discrete optimization **rarely work out of the box**:
structure of the algorithmic task should be accounted for

Architecture:

🚕 Understand which algorithmic tasks are well-suited for GNNs

Integration:

🚕 Greedy algorithm for online matching

🛹 Classic algorithms for rent-or-buy, scheduling

Supervision:

🚕 Leverage GNN size generalization properties for online matching

Robustness:

🛹 Calibration: decide whether to follow worst-case alg or predictions

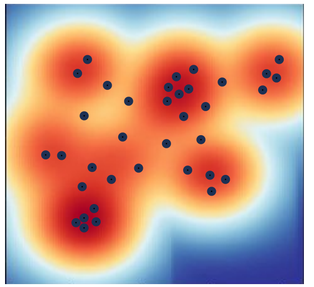
Ongoing directions: online matching

More unknowns:

- Graph structure, arrival probabilities, edge weights, …
- Requires a rich dataset where these can be learned from features

Other problems:

- Locality analysis likely applies to other combinatorial problems
- E.g., max-cut



Ongoing directions

Further open the ML black box of algorithms with predictions

- Analyze how metrics like competitive ratio 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

Machine Learning for Discrete Optimization: Theoretical Foundations

Ellen Vitercik

Stanford University