Online matching with **graph neural networks**

Ellen Vitercik (Stanford)

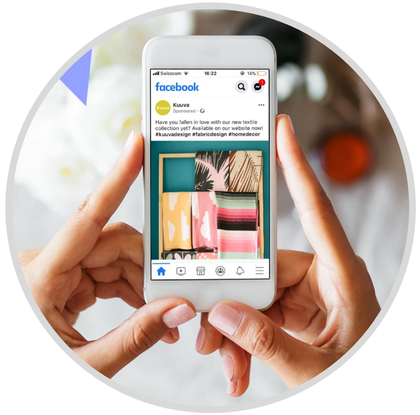
Machine learning for optimization

When designing algorithmic solutions to computational problems in practice:

- Often **ample data** about application domains

Routing

Rideshare



Digital advertising

Scheduling

- Data can potentially be harnessed to **optimize algorithmic performance**

Why machine learning?

Problems that a shipping company solves to route its trucks will change daily
…but not drastically

Demand and traffic will vary
…but road network will remain the same

Likely **underlying structure** that can be uncovered with the help of ML

Goal: use structure to optimize algorithm performance on future problems

Runtime, solution quality, …

Rich history

Data-driven approaches to algorithm selection

[e.g., Lobjois, Lemaître, ‘98; Gomes, Selman, ‘01; Leyton-Brown et al., 2009, Kadioglu et al., 2010, Sandholm, 2013, …]

ML for algorithm configuration

[e.g., Hutter et al. ‘09, ‘10, ‘11, Ansótegui et al. ‘09, Sandholm, 2013]

Integration of modern ML models into discrete solvers

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

Data-driven robust optimization

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

Theoretical guarantees

[e.g., book chapters by Balcan, ‘20; Mitzenmacher, Vassilvitskii, ‘20; …]

Key challenges

ML approaches to discrete optimization **rarely work out of the box**:
must be **aligned** to the algorithmic task at hand

Major challenges:

- **Architecture:** seq2seq, GNNs, transformers, hyperparameters, …?
- **Supervision:** how to supervise on (NP-hard) algorithmic problems?
- **Robustness:** can we provide any guarantees?

Outline

1. Introduction
2. **Graph neural networks (GNNs) for online matching** [ICML'24]

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., JACM'07]

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

Our contributions

GNNs for **Online Bayesian Bipartite Matching (OBBM)**

Theoretical: justify the suitability of GNNs for OBBM

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

- Supervise with actions of **optimal online agorithm** OPT_{on}
- Outperforms baselines on synthetic/semi-synthetic graphs

Key challenge: sheer complexity of OPT_{on}!

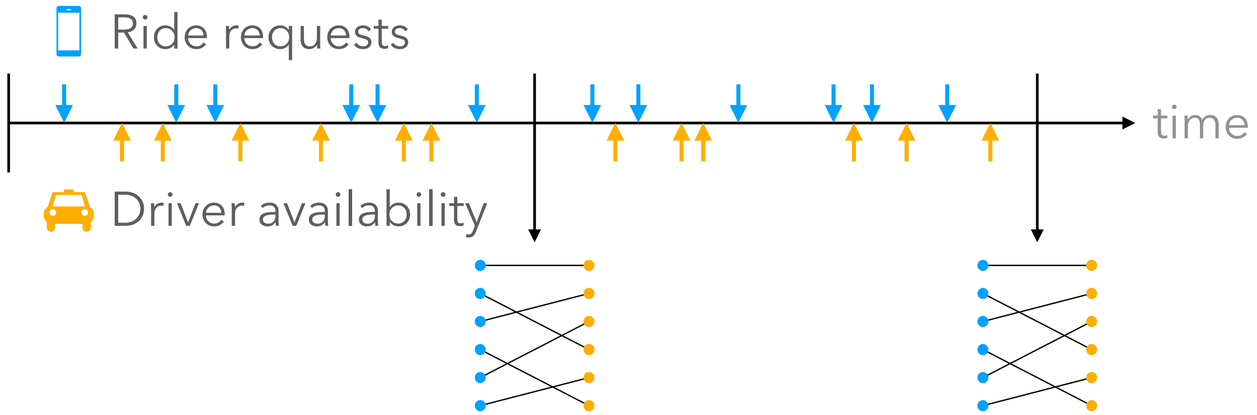
NP-hard to provide an α-factor approximation to OPT_{on} (for some constant α)

[Papadimitriou et al., EC'21]

Related work: RL for rideshare

Xu et al., KDD'18; Qin et al., INFORMS J. Appl. Anal. '20; Azagirre et al., INFORMS J. Appl. Anal. '24; …

- Open trip requests and available drivers are batched within time windows
- Matched at predefined discrete review times



- In contrast, we aim to match as soon as request arrives
  E.g., Adwords [Mehta et al., JACM'07]
- Future work: possible to combine techniques?

Figure inspired by Qin et al., Transp. Res. Part C Emerg. Technol. '22

Related work: ML for online matching

Focus on purely **theoretical** [e.g., Antoniadis et al., ICML'20, NeurIPS'20]

Adversarial learning to find good worst-case alg [e.g., Kong et al., ICLR'18]

Most related: Almorani et al. [TMLR'22]

- **RL** approach for unknown i.i.d. setting (optimal online algorithm unknown)
- In OBBM, can use supervision from OPT_{on} and **train on much smaller graphs**
- Learns a **tailored matching policy** per graph configuration, unlike us

Li et al. [ICML'23] bridge the gap between **worst-case/average-case**:

- **Dynamically switch** between expert and ML predictions
- Future work: combine with our approach for **robustness** guarantees?

Outline

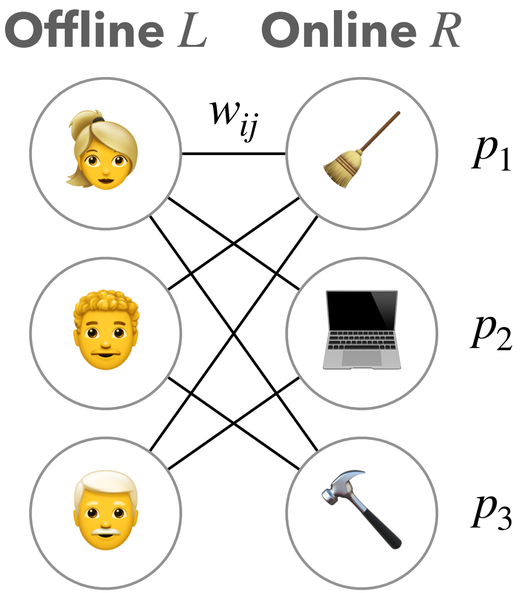
1. Introduction
2. Graph neural networks (GNNs) for online matching [ICML'24]
    i. **Background**
        a. **Online Bayesian bipartite matching**
        b. GNNs
    ii. GNN setup
    iii. Experiments
    iv. Theoretical guarantees

Online Bayesian Bipartite Matching (OBBM)

E.g., Krengel, Sucheston, Probab. Banach Spaces ’78; Papadimitriou et al., EC’21; Braverman et al., EC’22;…

- Bipartite graph $G = (L, R, E)$ with offline nodes L, online nodes R
  - E.g., represent **crowdworkers** and **tasks**
- Edge weights w_{ij} for each $(i, j) \in E \subseteq L \times R$
  - E.g., 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



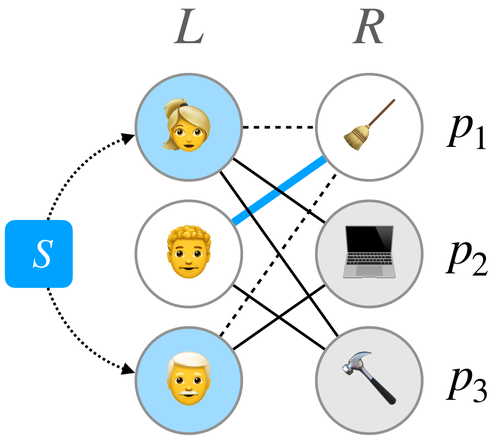
Online optimal algorithm and value-to-go

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

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

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

- Expectation over arrivals $\{t, \ldots, |R|\}$
- Matchings restricted to S



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 S

$$V_G(S,t) = \underbrace{(1 - p_t)}_{\downarrow} \cdot \underbrace{V_G(S, t+1)}_{\downarrow} + p_t \cdot \max\left\{ V_G(S, t+1), \max_{u \in N_G(t) \cap S} \left\{ w_{tu} + V_G\left(S \backslash \{u\}, t+1\right) \right\} \right\}$$

If t doesn't arrive, don't get any value from node 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 S

$$V_G(S,t) = \left(1 - p_t\right) \cdot V_G(S, t+1) + p_t \cdot \max\left\{ V_G(S, t+1), \max_{u \in N_G(t) \cap S} \left\{ w_{tu} + V_G\left(S \backslash \{u\}, t+1\right) \right\} \right\}$$

If t does arrive

We can skip it, and get no value from node 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 S

$$V_G(S,t) = \left(1 - p_t\right) \cdot V_G(S, t+1) + p_t \cdot \max\left\{ V_G(S, t+1), \max_{u \in N_G(t) \cap S} \left\{ w_{tu} + V_G\left(S \backslash \{u\}, t+1\right) \right\} \right\}$$

Or match to some unmatched neighbor $u \in N_G(t) \cap S$, get value w_{tu}, and remove u from unmatched set S

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 S

$$V_G(S,t) = \left(1 - p_t\right) \cdot V_G(S, t+1) + p_t \cdot \max \left\{ V_G(S, t+1), \max_{u \in N_G(t) \cap S} \left\{ w_{tu} + V_G\left(S \backslash \{u\}, t+1\right) \right\} \right\}$$

If node t arrives, don't match if $V_G(S, t+1)$ > $\max_{u \in N_G(t) \cap S} \left\{ w_{tu} + V_G\left(S \backslash \{u\}, t+1\right) \right\}$, else match to maximizing u

Outline

1. Introduction
2. Graph neural networks (GNNs) for online matching [ICML'24]
    i. Background
        a. Online Bayesian bipartite matching
        b. GNNs
    ii. GNN setup
    iii. Experiments
    iv. Theoretical guarantees

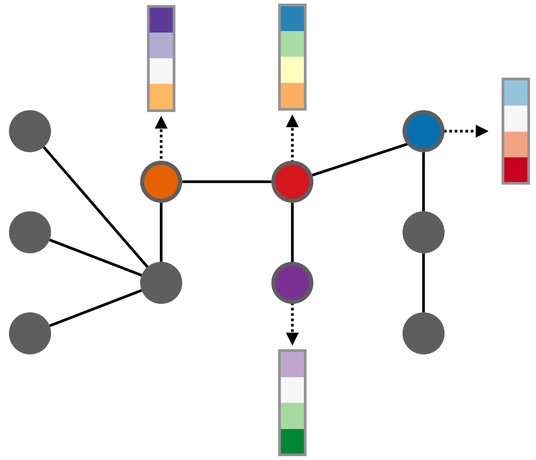
GNN architecture

GENConv [Li et al. '20]

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 $\vec{h}_v^{(k)} = \text{MLP}\left(\vec{h}_v^{(k-1)} + \overrightarrow{m}_v^{(k)}\right)$



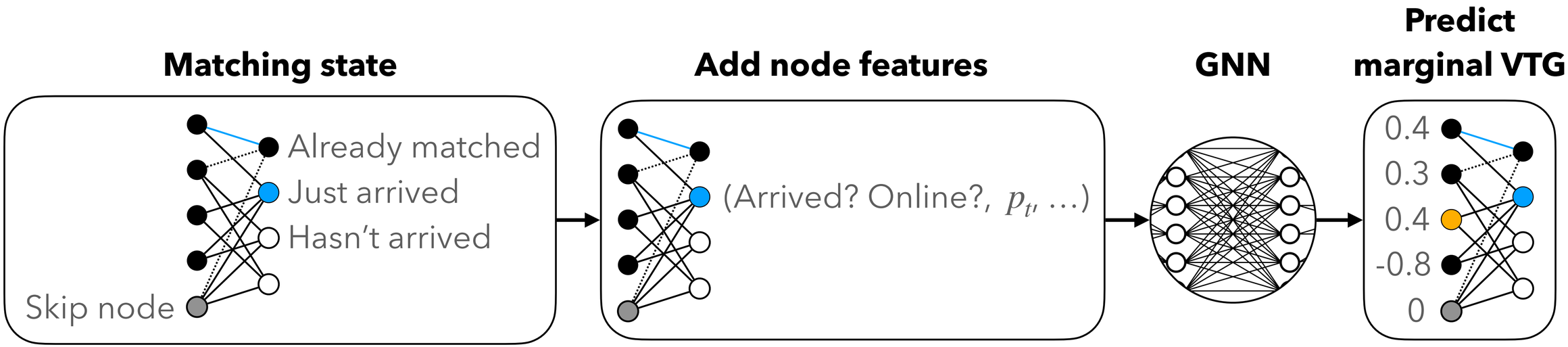
Use $\vec{h}_v^{(L)}$ to make predictions about node v

Outline

1. Introduction
2. Graph neural networks (GNNs) for online matching [ICML'24]
    i. Background
    ii. GNN setup
    iii. Theoretical guarantees
    iv. Experiments

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

Outline

1. Introduction
2. Graph neural networks (GNNs) for online matching [ICML'24]
    i. Background
    ii. GNN setup
    iii. Theoretical guarantees
    iv. Experiments

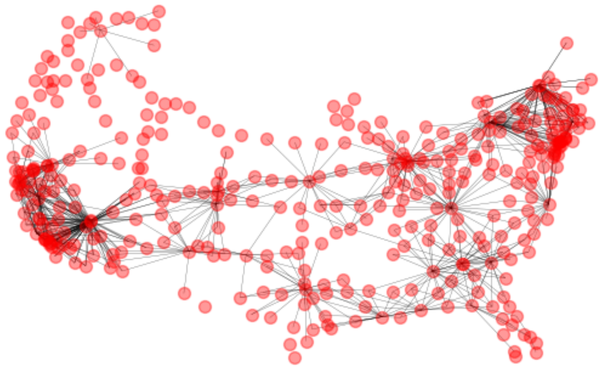
Why are GNNs well-suited for online matching?

Most practical instances are “well-structured”

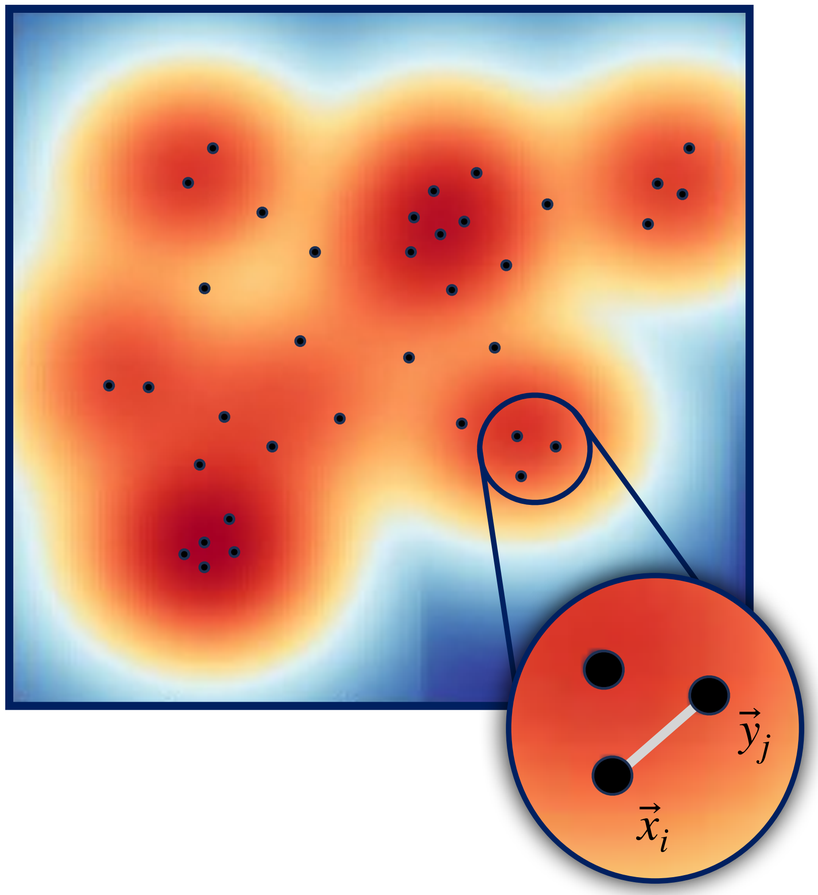
We show this rigorously for **random geometric graphs**

We prove that under these families, ***VTG is nearly local***

Already known that **GNNs compute local functions**



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 $\left([0,1]^d\right)$

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

Applications:

- Dynamic spatial matching
  [Kanoria '23]
- Opinion dynamics
  [Zhang et al. '14]
- Contagion in wireless networks
  [Nekovee '07]

GNNs learn local graph functions

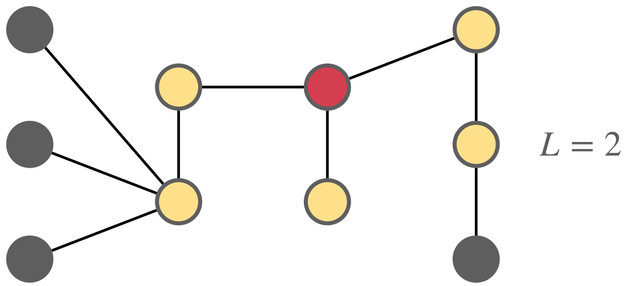
E.g., Tahmasebi et al., AISTATS'23

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 $f(G) = \phi\left(\psi\left(N_r(v)\right)_{v \in V}\right)$

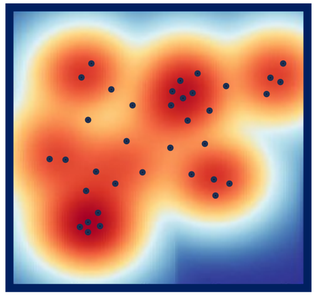
Observation: GNNs with L layers compute L-local functions



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

Outline

1. Introduction
2. Graph neural networks (GNNs) for online matching [ICML'24]
    i. Background
    ii. GNN setup
    iii. Theoretical guarantees
    iv. Experiments

Graph datasets

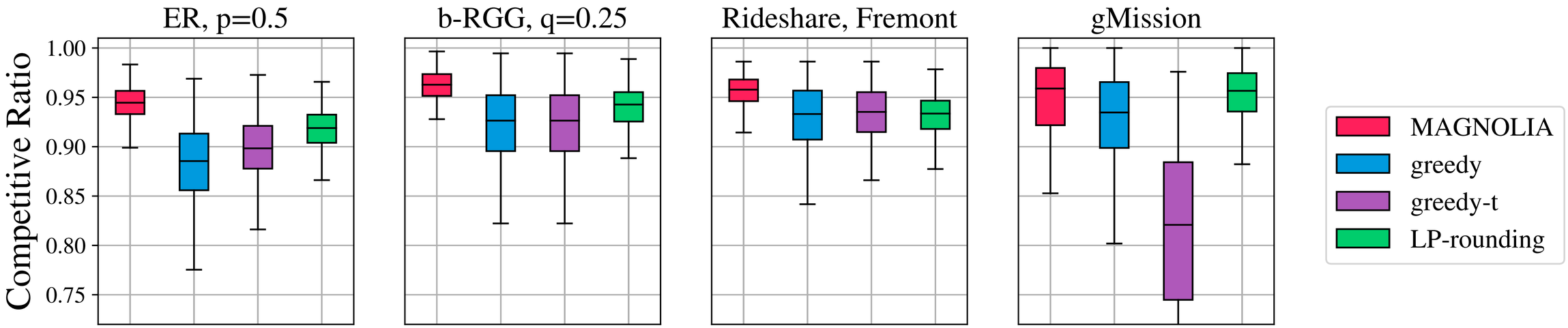
Random graph families

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

Semi-synthetic graphs

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

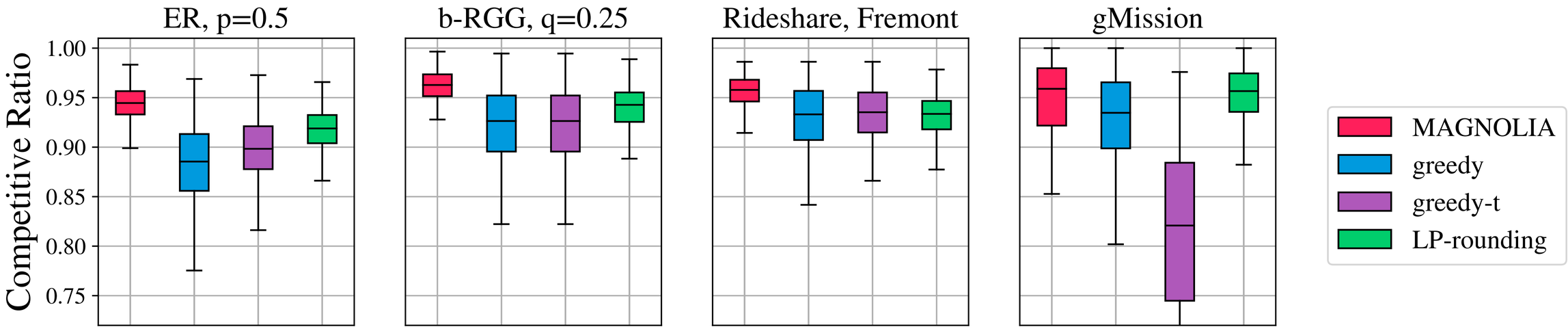
Experiments



Competitive ratio: $\frac{\text{ALG}}{\text{Offline optimal}}$

Trained on graphs of size (6x10), competitive ratios shown for graphs of size (10x30)

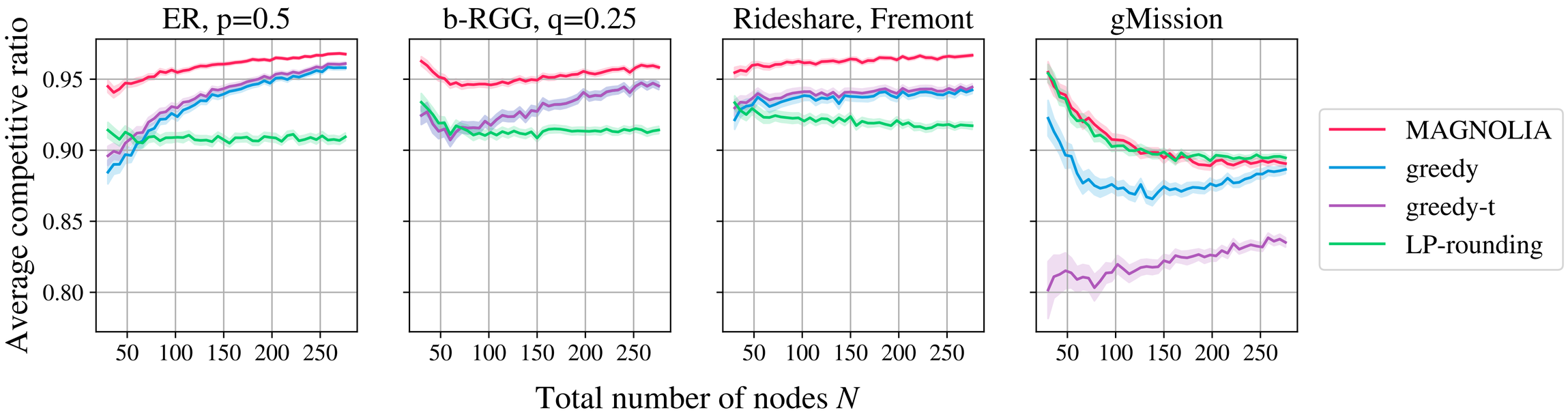
Experiments



- 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

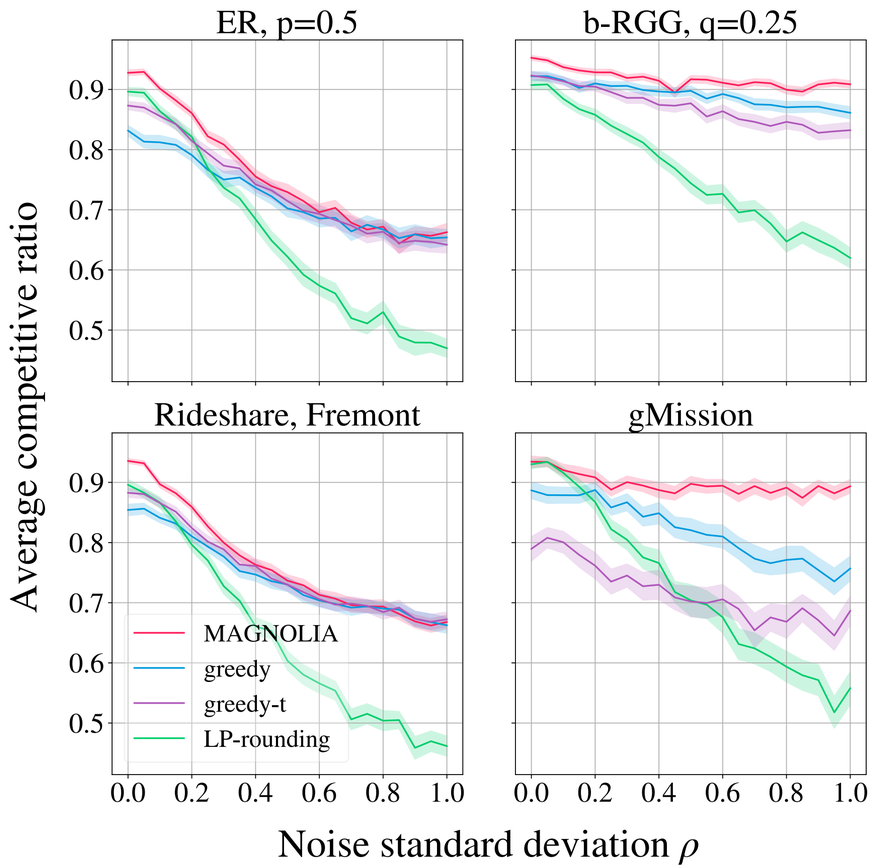
Papadimitriou et al. [EC'21], **Braverman et al. [EC'22]**, Naor et al. ['23]

Experiments: Size generalization



Performance generalizes to graphs up to 20x larger than training graphs (6x10)

Experiments: Noise



Weights, arrival probabilities may be unknown
Instead, only **noisy estimates** from data

Add $N\left(0, \rho^2\right)$ noise independently to edge weights and arrival probabilities

MAGNOLIA **more robust** than baselines

Conclusions and future directions

Graph neural networks (GNNs) for online matching

- NP-hard problem, applications in online advertising, crowdsourcing, …
- Supervise using (exponential time) online optimal policy
  - Train on small graphs, test on large graphs
- Theoretical guarantees to understand why GNNs are suitable for this task

Future directions

- **More unknowns:** graph structure, edge weights, arrival probabilities, …
- **Other problems:** locality analysis might apply to other problems

Online matching with **graph neural networks**

Ellen Vitercik (Stanford)